

A Joint Translation Model With Integrated Reordering

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der
Universität Stuttgart zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von
Nadir Durrani
aus Lahore

Hauptberichter:	Professor Dr. Hinrich Schütze
Mitberichter:	Professor Dr. François Yvon
Mitberichter:	Dr. Alexander M. Fraser
Mitberichter:	PD Dr. Helmut Schmid

Tag der mündlichen Prüfung: 19. November 2012

Institut für maschinelle Sprachverarbeitung
der Universität Stuttgart

2012

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel und Literatur angefertigt habe.

Stuttgart, den 02.10.2012

Nadir Durrani



With the Name of Allah, the Most Gracious, the Most Merciful

.....

*Read in the Name of thy Lord who created,
created Man from a blood-clot.*

*Read, and thy Lord is the Most Generous,
who taught by the pen,
taught man that he knew not.*

(Al-Quran, 96:1-5)

*No man succeeds without a good woman behind him. Wife or mother,
if it is both, he is twice blessed indeed. [Harold MacMillan]*

I dedicate this to the two most loving women in this world ...

*My mother, who made me the man I am today, this wouldn't have been
possible without her love and prayers ...*

*and Iqra, my understanding and patient wife, for the encouragement and
support she provided in the last 4 years ...*

Acknowledgments

I would like to take this opportunity to thank all those who helped me make my dream come true.

- I express my deepest and utmost gratitude to my supervisors, Alex Fraser and Helmut Schmid. Without their guidance and support I would not have been able to finish my dissertation. One simply could not wish for more helpful and friendlier supervisors.
- I would like to give a special thanks to Alex Fraser, also for his help in getting me an internship at IBM and also for the funding support after my scholarship lapsed.
- I would like to thank Prof. Hinrich Schütze for accepting me to be PhD Student at the Institute of Natural Language Processing. His feedback to my work and ideas have led to a great improvement.
- I am grateful to Professor François Yvon¹ for his survey of this work.
- My special thanks to Higher Education Commission (HEC) of Pakistan for funding my dissertation.
- I would also like to thank “Sozialamt für Esslingen Am Neckar” for the “Kinder and Wohen Geld”. Without this financial support, it would have been extremely difficult for me to support my family in Germany.

¹LIMSI – The Computer Sciences Laboratory for Mechanics and Engineering Sciences

- I would like to thank my friend and colleague Hassan Sajjad for our discussions and exchange of ideas, for his moral support and words of encouragement during the difficult times. My path to success would have been very different and difficult without you.
- I thank Fabienne Braune for her feedback on my paper and this work.
- I thank and apologize my fellow colleagues and members of IMS for bearing with my processes on the servers.
- I want to thank to Professor. Sarmad Hussain who introduced me to the field of Natural Language Processing, and helped me in getting admission at the University of Stuttgart.
- I thank family members my parents, my in-laws and my siblings for their patience, love and encouragement. Though miles may lie between us, I have always felt your presence and support in my studies. Your prayers have led me to this point.
- Special thanks to my Chachu², Prof. Qaiser Durrani, who inspired me to pursue doctoral studies. For the last two decades, I have lived a dream to be like you. Thank you for all the good advice, for the guidance encouragement and support.
- A loving thanks to Ayan, my son, who was born nearly at the beginning of my PhD studies. I was advised by many not to have a kid during PhD but my son did not disturb my studies one bit. In fact his presence brought me joy and was a source of delight.
- Last but not the least, I am hugely indebted to Iqra, my loving wife. Her tolerance of my mood swings, working late nights and during weekends helped me focus on my work in peace. Her patience and support was in the end what made this dissertation possible.

²Chachu is the word used for father's younger brother in Urdu

Abstract

This dissertation aims at combining the benefits and to remedy the flaws of the two popular frameworks in statistical machine translation, namely Phrase-based MT and N-gram-based MT.

Phrase-based MT advanced the state-of-the art towards translating phrases³ than words. By memorizing phrases, phrasal MT, is able to learn local reorderings, and handling of other local dependencies such as insertions, deletions etc. Inter-phrasal reorderings are handled through the lexicalized reordering model, which remains the state-of-the-art model for reordering in phrase-based SMT till date. However, phrase-based MT has some drawbacks:

- Dependencies across phrases are not directly represented in the translation model
- Discontinuous phrases cannot be represented and used
- The reordering model is not designed to handle long range reorderings
- Search and modeling problems require the use of a hard reordering limit
- The presence of many different equivalent segmentations increases the search space

³Note that phrases in this thesis mean sequence of words, which may not be linguistic phrases

- Source word deletion and target word insertion outside phrases is not allowed during decoding

N-gram-based MT exists as an alternate to the more commonly used Phrase-based MT. Unlike Phrasal MT, N-gram-based MT uses minimal translation units called as tuples. Using minimal translation units, enables N-gram systems to avoid the spurious phrasal segmentation problem in the phrase-based MT. However, it also gives up the ability to memorize dependencies such as short reorderings that are local to the phrases. Reordering in N-gram MT is carried out by source linearization and POS-based rewrite rules. The search graph for decoding is constructed as a preprocessing step using these rules. N-gram-based MT has the following drawbacks:

- Only the pre-calculated orderings are hypothesized during decoding
- The N-gram model can not use lexical triggers
- Long distance reorderings can not be performed
- Unaligned target words can not be handled
- Using tuples presents a more difficult search problem than that in phrase-based SMT

In this dissertation, we present a novel machine translation model based on a joint probability model, which represents translation process as a linear sequence of *operations*. Our model like the N-gram model uses minimal translation units, but has the ability to memorize like the phrase-based model. Unlike the “N-gram” model, our operation sequence includes not only translation but also reordering operations. The strong coupling of reordering and translation into a single generative story provides a mechanism to better restrict the position to which a word or phrase can be moved, and is able to handle short and long distance reorderings effectively. This thesis remedies the problems in phrasal MT and N-gram-based MT by making the following contributions:

- We proposed a model that handles both local and long distance dependencies uniformly and effectively
- Our model is able to handle discontinuous source-side units
- During the decoding, we are able to remove the hard reordering constraint which is necessary in the phrase-based systems
- Like the phrase-based and unlike the N-gram model, our model exhibits the ability to memorize phrases
- In comparison to the N-gram-based model, our model performs search on all possible reorderings and has the ability to learn lexical triggers and apply them to unseen contexts

A secondary goal of this thesis is to challenge the belief that conditional probability models work better than the joint probability models in SMT and that the source-side context is less helpful in the translation process.

Zusammenfassung

Diese Dissertation setzt sich zum Ziel, die Vorteile der beiden populärsten Methoden in der statistische maschinellen Übersetzung, nämlich der phrasenbasierten Übersetzung und der N-Gramm-basierten Übersetzung zu kombinieren und ihre Nachteile zu vermeiden.

Phrasenbasierte maschinelle Übersetzung (PBMÜ) erzielte eine zuvor unerreichte Genauigkeit, indem ganze Wortfolgen (genannt Phrasen) statt einzelner Wörtern übersetzt werden. Durch Abspeicherung von phrasalen Übersetzungen ist PBMÜ in der Lage, lokale Umordnungen zu lernen wie auch die Behandlung anderer lokaler Abhängigkeiten, die bei Einfügungen und Löschungen auftreten. Phrasenübergreifende Umordnungen werden mit Hilfe des lexikalisierten Umordnungsmodelles behandelt, das im Bereich der PBMÜ nach wie vor Stand der Technik ist. PBMÜ hat jedoch einige Nachteile:

- Abhängigkeiten zwischen Phrasen werden nicht direkt im Modell repräsentiert.
- Diskontinuierliche Phrasen können nicht repräsentiert und verwendet werden.
- Das Umordnungsmodell ist ungeeignet für die Behandlung von Umordnungen über weite Distanzen.
- Probleme bei der Modellierung und Suche erfordern eine harte maximale Grenze für die Umordnungsdistanz.
- Die Existenz vieler verschiedener äquivalenter Segmentierungen lässt den Suchraum anwachsen.

- Die Löschung von Wörtern im Ausgangssatz und die Einfügung von Wörtern in der Übersetzung ist nur innerhalb von (größeren) Phrasen erlaubt.

N-Gramm-basierte maschinelle Übersetzung (NGMÜ) stellt eine Alternative zu der weiter verbreiteten phrasenbasierten Übersetzung dar. Im Gegensatz zur PBMÜ verwendet die NGMÜ minimale Übersetzungseinheiten, die “Tupel” genannt werden. Dadurch vermeidet die NGMÜ das Problem der PBMÜ mit vermeintlichen Ambiguitäten in der Segmentierung. Allerdings verliert es damit auch die Fähigkeit, lokale Abhängigkeiten bei Umordnungen über kurze Distanzen mit Hilfe der Phrasen abzuspeichern. Die Umordnung erfolgt in der NGMÜ mit Wortart-basierten Umordnungsregeln. Der Suchgraph für die Übersetzung wird mit Hilfe dieser Regeln bereits in einem Vorverarbeitungsschritt aufgebaut. NGMÜ hat folgende Nachteile:

- Der Übersetzer erlaubt nur die vorausberechneten Umordnungen.
- NGMÜ kann keine lexikalischen Trigger für Umordnungen berücksichtigen.
- Nicht alignierte Wörter in der Übersetzung sind nicht möglich.
- Die Verwendung von Tupeln vergrößert das Suchproblem im Vergleich zur PBMÜ.

In dieser Dissertation stelle ich ein neuartiges Modell für die maschinelle Übersetzung vor, das die gemeinsame Wahrscheinlichkeit von Quell- und Zielsatz beschreibt und den Übersetzungsprozess als eine lineare Folge von Operationen repräsentiert. Wie die NGMÜ verwendet auch dieses Modell minimale Übersetzungseinheiten, ist aber dennoch wie die PBMÜ in der Lage, kurze Übersetzungsteile abzuspeichern. Im Gegensatz zur NGMÜ hat das Modell nicht nur Operationen für Übersetzungen sondern auch für Umordnungen. Die starke Kopplung von Übersetzung und Umordnung in einem einzigen N-Gramm-Modell liefert einen Mechanismus, mit dem die Position, zu der ein

Wort oder eine Wortfolge verschoben wird, besser eingeschränkt werden kann, und mit dem Umordnungen über kurze wie lange Distanzen effektiver behandelt werden können. Diese Arbeit löst die Probleme in PBMÜ und NGMÜ durch die folgenden wissenschaftlichen Beiträge:

- Es wird ein Modell präsentiert, das Umordnungen über kurze wie lange Distanzen einheitlich und effektiv behandelt.
- Das Modell erlaubt diskontinuierliche Übersetzungseinheiten auf der quellsprachlichen Seite.
- Im Gegensatz zur PBMÜ erfordert der Übersetzer keine harte Begrenzung der maximalen Umordnungsdistanz.
- Wie die PBMÜ und im Gegensatz zur NGMÜ hat das Modell die Fähigkeit, Phrasen abzuspeichern.
- Im Unterschied zur NGMÜ wird der Suchraum nicht im voraus eingeschränkt, und es können lexikalische Trigger für die Umordnung gelernt und auf neue Kontexte angewendet werden.

Die Arbeit stellt die Meinung in Frage, dass bedingte Wahrscheinlichkeitsmodelle in der maschinelle Übersetzung generell besser funktionieren, und dass Kontextinformation auf der quellsprachlichen Seite weniger nützlich für den Übersetzungsprozess ist.

Contents

1	Introduction	21
1.1	Machine Translation Problem	21
1.2	Motivation: Problems in Statistical Machine Translation	23
1.2.1	Reordering	23
1.2.2	Modeling of Translation Units	25
1.2.3	Joint Probability Model	26
1.2.4	Secondary Goals	27
1.3	Thesis Organization	27
2	Phrase-based Statistical Machine Translation	29
2.1	Word-based Models	29
2.1.1	Noisy Channel Model	30
2.1.2	Generative Models	31
2.1.3	Summary of the IBM Word-based Models	33
2.1.4	Drawbacks of Word-based Translation	36
2.2	Phrase-based Models	38
2.2.1	Advantages of the Phrase-based Model	39
2.2.2	Translation Models	40
2.2.3	Lexicalized Reordering Models	49
2.2.4	Log-linear Model	57
2.2.5	Search	60
2.2.6	Decoding Complexity	62
2.2.7	Future Cost	65
2.3	Drawbacks of Phrase-based SMT	68
2.3.1	Handling of Non-local Dependencies	69
2.3.2	Modeling of Gappy Units	71
2.3.3	Weak Reordering Model	73
2.3.4	Hard Distortion Limit during Search	75
2.3.5	Spurious Phrasal Segmentation	77
2.3.6	Deletions and Insertions out of Phrases	79
2.4	Hierarchical Phrase-Based Translation	80

Contents

2.5	Discontinuous Phrase-based SMT	83
2.6	Chapter Summary	86
3	N-gram-based SMT	88
3.1	Translation Model	88
3.1.1	Tuple Extraction	89
3.1.2	Crossing Alignments – Embedded Words	90
3.1.3	Tuple Pruning	91
3.1.4	Unaligned Words	92
3.2	Reordering Model	95
3.2.1	Source Linearization and Tuple Unfolding	96
3.2.2	Rewrite Rules	98
3.2.3	Handling of Gappy Units	103
3.2.4	Factored Bilingual Units	108
3.3	Features Used in N-gram-based SMT	110
3.4	Search	111
3.4.1	Reordering	111
3.4.2	Searching the Word Graph	113
3.5	Comparison with Phrase-based SMT	114
3.5.1	Translation Model	114
3.5.2	Reordering Framework	120
3.5.3	Decoding	122
3.5.4	Drawbacks of N-gram-based SMT	126
3.6	Chapter Summary	127
4	A Joint Sequence Translation Model with Integrated Reordering	129
4.1	Introduction	129
4.2	Generative Story	130
4.2.1	Basic Operations	131
4.2.2	Reordering Operations	131
4.2.3	Discontinuous Translation Units	136
4.2.4	Insertions and Deletions	137
4.2.5	Formal Definition of Operations	137
4.2.6	Conversion Algorithm	143
4.2.7	Algorithmic Complexity	147
4.3	Model	148
4.3.1	Search	148
4.3.2	Discussion	150
4.3.3	Discriminative Modeling	155

Contents

4.4	Word Alignments - Post-processing Heuristic	163
4.4.1	Removing Target-Side Discontinuities	163
4.4.2	Removing Unaligned Target Words	165
4.4.3	Algorithmic Complexity	168
4.5	Decoding	168
4.5.1	Tuple Extraction	169
4.5.2	Hypothesis Extension	174
4.5.3	Future Cost Estimation	175
4.5.4	Recombination and Pruning	180
4.5.5	Algorithmic Complexity	180
4.6	Training	182
4.7	Evaluation	183
4.7.1	Baseline Systems	184
4.7.2	Results	187
4.8	Chapter Summary	210
5	Conclusion	212
5.1	Contributions of this Work	212
5.1.1	Comparison with the Phrase-based System	212
5.1.2	Hard Reordering Limit	218
5.1.3	Comparison with the N-gram Model	220
5.2	Shortcomings and Future Work	223
5.2.1	Decoding	223
5.2.2	Future Cost Estimation	224
5.2.3	Reordering Constraints	226
5.2.4	Removing Target-Side Discontinuities	226
5.2.5	Unaligned Target-Side Words	228
5.2.6	Removing Deficiency and Derivational Ambiguities	229
5.2.7	Using POS-based and Lemma-based Operation Sequence Model	232
5.2.8	Additional Features	233
	Bibliography	235

List of Tables

2.1	Distortion Model for Phrase-based SMT	42
2.2	Extracted Phrases	46
2.3	Orientation Model for Phrase-based SMT – Current Phrase Pair $\langle F_{a(j)}, E_j \rangle$, Previous Phrase Pair $\langle F_{a(j-1)}, E_{j-1} \rangle$	51
2.4	Forward and Backward Orientation Models for Phrase-based SMT	56
2.5	Hypothetical Phrase Table	70
2.6	Hypothetical Phrase Table	71
2.7	Learned Parameters for Phrase-based SMT	75
2.8	Hypothesized Translations with the Orientations of <i>würden</i> and <i>abstimmen</i> as shown in Table 2.7	76
2.9	Discontinuous Phrases in Figure 2.25 – X = Place holder for 1 or more words	84
3.1	Reordering Rules	112
3.2	Extracted Phrases and Tuples	117
4.1	Step-wise Generation of Example 4.8. The arrow indicates po- sition j	147
4.2	All Possible Discontinuous Phrases in a 4 Word Sentence	170
4.3	Comparison of Symmetrization Heuristics + post-processing heuris- tic (PP) – German-to-English	188
4.4	Comparison of Symmetrization Heuristics + post-processing heuris- tic (PP) – Spanish-to-English	189
4.5	Comparison of Symmetrization Heuristics + post-processing heuris- tic (PP) – French-to-English	189
4.6	Comparison of Our Systems – DE = German, ES = Spanish, FR = French	192
4.7	10-best Translation Options With & Without Gaps and using our Heuristic	193
4.8	MERT-tuned Feature Vector for \mathbf{OS}_{ag}	194
4.9	Comparison of German-to-English on (S)mall, (M)edium and (L)arge Scaled Data	195

List of Tables

4.10	Comparison of German-to-English on 5-Test Sets (Large Scaled Data)	195
4.11	Comparison of Spanish-to-English on (S)mall, (M)edium and (L)arge Scaled Data	198
4.12	Comparison of Spanish-to-English on 5-Test Sets (Large Scaled Data)	198
4.13	Statistics on Reordering Distances in German, Spanish and French calculated from Bilingual Training Corpus	199
4.14	Comparison of French-to-English on (S)mall, (M)edium and (L)arge Scaled Data	199
4.15	Comparison of French-to-English on 5-Test Sets (Large Scaled Data)	200
4.16	Comparison to Full N-gram System l = Lexical Reordering, l+p = Lexical Reordering + POS-based bilingual Model – German-to-English	201
4.17	Comparison to Full N-gram System l = Lexical Reordering, l+p = Lexical Reordering + POS-based bilingual Model – Spanish-to-English	201
4.18	Comparison to Full N-gram System l = Lexical Reordering, l+p = Lexical Reordering + POS-based bilingual Model – French-to-English	202
4.19	Experiments on Distortion Limit – German-to-English	202
4.20	Experiments on Distortion Limit – Spanish-to-English	203
4.21	Experiments on Distortion Limit – French-to-English	203
4.22	Contribution of Different Feature Functions	205
4.23	Removing Source Word Deletion Model	206
4.24	Removing Monolingual Language Model Features from Moses, NCode and Our System	207
4.25	Contribution of Different Feature Functions	208
4.26	Translation Quality of Our System and Moses with different Stack Sizes	209
4.27	Translation Quality of Our System Varying the Number of Translation Options	210
5.1	Total Number of Extracted Translation Units (Types) in Phrasal and Our System with both Continuous and Discontinuous Units	219

List of Figures

2.1	Generative Story – Model 3	35
2.2	Phrase-based Machine Translation	39
2.3	German–English Sentence Pair with a Phrasal Segmentation . .	42
2.4	Word Alignments (a) German-to-English, (b) English-to-German, (c) Union of (a) & (b)	44
2.5	German–English Sentence Pair with a Phrasal Segmentation . .	51
2.6	Lexicalized Reordering Model – Orientations (m)onotonic, (s)wap, (d)iscontinuous	52
2.7	Lexicalized Reordering Model (Koehn et al., 2005)	53
2.8	Lexicalized Reordering Model (Ohashi et al., 2005)	54
2.9	Lexicalized Reordering (Galley and Manning, 2008) – Backward Orientation Model	55
2.10	Lexicalized Reordering (Galley and Manning, 2008) – Forward Orientation Model	56
2.11	Learning Feature Weights	59
2.12	A Sample Hypothesis	61
2.13	Beam Search Decoding	62
2.14	Hypothesis Extension and Recombination	63
2.15	Future Cost Estimation for Larger Spans	67
2.16	Using Future Cost during Decoding	68
2.17	Handling of Local Dependencies – Dotted lines = Word Align- ments	70
2.18	Handling of Gaps – Dotted lines = Word Alignments – (a) Learned Phrase (b) Unseen Context	71
2.19	Lexicalized Reordering Model – (a) Word Alignments (b) Back- ward Orientation (c) Forward Orientation	74
2.20	Long Distance Reordering	75
2.21	Spurious Phrasal Segmentation – Training	78
2.22	Spurious Phrasal Segmentation – Recombined Hypotheses Rep- resented with Dotted Lines	79
2.23	German Word Deletion – Different Learned Phrases	81

List of Figures

2.24	Alignments Not Handled By Hiero	83
2.25	Handling of Local Dependencies – Dotted lines = Word Alignments	83
3.1	Tuple Generation	90
3.2	Unaligned English Words	93
3.3	Symmetrized Word Alignments with Union	95
3.4	Short Distance Reordering	96
3.5	Source Linearization - Two Steps	97
3.6	Source Linearization of Example in Figure 3.4	98
3.7	Search Graph Extension	99
3.8	Search Graph Extension - Rule Fail	101
3.9	Rewrite Rules - Phrasal Chunks	101
3.10	Syntax Enhanced Reordering	102
3.11	Extracted Rules – Different Levels of Generalization	104
3.12	Source-side Gaps	105
3.13	Target-side Gaps	106
3.14	Handling Target Gaps Using Split Tokens	107
3.15	Syntax Aware Split Rules	108
3.16	No Evidence of Reordering During Test	109
3.17	Lattice-based Search Graph	112
3.18	Abstract Translation Process	113
3.19	Training Example	116
3.20	Handling Local and Non-local Reorderings	116
3.21	Handling German-Side Discontinuities	118
3.22	Handling English-Side Discontinuities	119
3.23	Comparison of Search - Phrase-based verses N-gram-based SMT	123
3.24	Comparison of Search - Phrases vs. Tuple Placement on Stack	125
3.25	Long Distance Reordering	126
4.1	Reordering Example	131
4.2	Sub-ordinate German-English Clause Pair	133
4.3	Complex Reordering Pattern in German-English Translation	135
4.4	Discontinuous German-side Cept	136
4.5	Insertion and Deletion	137
4.6	An Example to Illustrate Jump Forward Operation	141
4.7	Jump Forward Scenarios	142
4.8	Sample Bilingual Sentence for the Dry-Run of the Conversion Algorithm – German Indexes Represent Word Positions and the English Indexes Represent Cept Positions	145

List of Abbreviations

4.9	Sub-ordinate German-English Clause Pair	152
4.10	Handling Dependencies (a) Local (b) Non-local	153
4.11	An Example to Demonstrate Gap Distance Penalty	159
4.12	An Example to Demonstrate Relation Between Reordering and Gap Distance Penalties	160
4.13	An Example to Demonstrate Gap Width Penalty	161
4.14	An Example to Demonstrate Lexical Probability Feature	162
4.15	Target Side Discontinuities	164
4.16	Multiple Target Side Discontinuities	164
4.17	Learning Attachment Counts from Word Alignments	167
4.18	An Example to Demonstrate the Cept Extraction Algorithm	170
4.19	Search Tree for Source-side Cept Extraction	172
4.20	A Sample Hypothesis	175
4.21	Comparison of Search - Phrases vs. Tuple Placement on Stack	181
4.22	Discontinuous Phrases Configurations (i) Cross-serial DTU (ii) Bonbon	186
4.23	Long Distance Gappy Cept	189
4.24	Target-Side Discontinuous Units Statistics	191
4.25	Target-Side unaligned Units – Statitics collected from 1000K Parallel Data	192
4.26	Example-1 from Test MT09 – Demonstrating the Better Re- ordering Mechanism	196
4.27	Example-2 from Test MT09 – Demonstrating Better Reordering Mechanism	196
4.28	Example-3 from Test MT08 – Demonstrating Ability to Mem- orize Phrases	197
4.29	Example-4 from Test MT08 – Where No Reordering Limit Helps	204
4.30	Example-5 from Test MT08 – Where No Reordering Limit Helps	204
5.1	Handling of Local Dependencies – Dotted lines = Word Align- ments	213
5.2	Handling of Gaps – Dotted lines = Word Alignments – (a) Learned Phrase (b) Unseen Context	214
5.3	Learned Pattern	215
5.4	Long Distance Reordering	223
5.5	Target Side Discontinuities	227
5.6	Target Side Discontinuities – Split Rules	228

1 Introduction

1.1 Machine Translation Problem

Machine translation has been one of the oldest studied problems in the area of artificial intelligence. Given a foreign language sentence, the task is to automatically translate it into English. The efforts in machine translation started with the motivation of code breaking during World War II. Because of this machine translation is historically viewed as a decipherment problem. Warren Weaver, one of the pioneering researchers in machine translation wrote in 1947:

“When I look at an article in Russian, I say: This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode” (Weaver, 1955)

The lack of computational resources required to model complex linguistic phenomenon lead to the publication of ALPAC report that stopped the funding for machine translation in US. The efforts to build an automatic machine translation system were subsequently stalled. However, the improvement in computing capacity and the growing demands for translations in the multilingual societies, re-established the field in late 70's. Different approaches to solve the problem have been taken ever since:

Rule-based MT: The focus of rule-based machine translation (Hutchins and Somers, 1992; Arnold et al., 1994) is to develop a set of linguistic rules to explain the translation from source to target. This approach requires a lot of human effort and linguistic resources that are expensive to come by.

Empirical Approaches to MT: Over the last two decades, Machine Translation has taken a stride from rule-based towards corpus-based approach. The shift was based on two arguments i) human resources are expensive and the effort to produce linguistic rules has to be repeated for each new language pair, ii) natural language is too complex to be captured with a set of rules. The idea of the corpus-based approach was to automatically learn these rules from a large corpus of translated data. With the idea of learning from data, two popular frameworks namely example-based machine (EBMT) and statistical machine translation (SMT) were born.

Example-based MT: The EBMT (Brown, 1996) framework is driven by the idea of translation by analogy. The training data is learned from the bilingual parallel data that contains sentences that are translation of each other. During test, the idea is to map the input sentence to the previously seen examples in the training data and make appropriate changes to their stored translation to produce the translation of the test sentence.

Statistical MT: Statistical Machine Translation like EBMT is trained on the bilingual training corpus. However, SMT learns statistical models from the parallel and monolingual data and decode the test sentence to maximize the statistical models learn during training. Last two decades have witnessed exciting amount of research in this area. Initial efforts were based on word-to-word translation, making a transition towards using phrases as unit of translation. Recent systems have incorporated linguistic information such as morphology and syntax inside of statistical machine translation.

Hybrid MT: More recently people have also started to combine the advantages of rule-based and example-based MT into hybrid systems. The success of statistical machine translation is based on abundantly available data. However, rule-based systems are better at handling exceptions, the statistics of which are sparsely available in the data. Rule-based system also translate out-of-domain data better than statistical machine translation.

1.2 Motivation: Problems in Statistical Machine Translation

The aim of this dissertation is to improve the state-of-the-art in different aspects of the statistical machine translation. We studied the drawbacks in the machinery of the two popular frameworks in SMT namely the phrase-based model and the N-gram model and proposed a model that combines the benefits of the two and rectifies their drawbacks. In this section we will briefly discuss the problems in these frameworks to motivate the work done in this dissertation.

1.2.1 Reordering

The most difficult problem in statistical machine translation is getting the word order right. Some language pairs such as Hindi-Urdu, Thai-Lao have monotonic word order; translation of these is therefore easier. On the contrary translation between German and English or Japanese and English language pairs is more difficult because of their different syntactic structures. This makes the translation task in these language pairs more difficult both in terms of modeling and computation. Firstly the system must learn complex reordering patterns and syntactic difference between the languages and build a model that is able to reward good word orders and penalize the bad ones. Secondly during translation we have to construct an output sentence. Trying out all possible word orders is computationally feasible and makes the search problem hard.

Both Phrase-based and N-gram models are poor at the long distance reorderings. Therefore over the years, people have used syntax-based machine translation system for the translation of German-English and Japanese-English language pairs.

Phrase-based MT: Phrase-based model is able to model intra-phrase reorderings well; however, non-local reorderings are not handled directly through translation model. The lexicalized reordering model plainly learns how a phrase-

pair was ordered with respect to its previous phrase without taking into account the dependencies with previously translated source and target words. The reordering model relies heavily on the language model to get the word order right. However, the language model can not justify long distance reorderings because of the dis-preference of the translation model for non-local movement. This problem is magnified during decoding when the reordering model and other feature components in the phrase-based system can not differentiate between good and bad hypotheses, resulting in both modeling and search errors. A hard reordering limit of 6 words has to be applied in order to reduce the search space. This consequently means that a movement beyond a window of 6 words is never hypothesized.

N-gram-based MT: The reordering model in the N-gram-based system also has several drawbacks. i) The reordering model is based on POS-based rewrite rules, therefore require additional linguistic resource i.e. POS-tagged data and a POS-tagger other than parallel data to build an N-gram-based SMT system. ii) Like the phrase-based MT it also applies a hard-reordering constraint in form of restriction on the length of rewrite rule. A rule beyond 7 POS-tags is filtered out during training due to data sparsity. iii) POS-based rules do not generalize well on the unseen patterns unless the test sentence occurs with the exact sequence of POS-tags as observed during training. iv) Another drawback of this reordering approach is that search is only performed on a small number of reorderings that are pre-calculated on the source side independently of the target side. Often, the evidence for the correct ordering is provided by the target-side language model (LM). In the N-gram approach, the LM only plays a role in selecting between the pre-calculated orderings.

Goal: In this dissertation we aim at improving the modeling of long distance reordering. Our secondary object is to remove the hard-reordering limit during decoding and search for all possible permutations.

1.2.2 Modeling of Translation Units

One main difference between Phrase-based and N-gram based MT is the difference in modeling of translation units. The unit of translation in phrase-based MT (PBSMT) is phrase, a sequence of consecutive words which may not necessarily be a linguistic constituent. Using phrases helps the PBSMT to memorize local dependencies. However, non-local dependencies are not modeled directly in the translation model. This creates a bias towards local dependencies, the evidence of which is found in the translation model. Secondly, using phrases also creates spurious ambiguity in phrasal segmentations which is undesirable from the perspective of both training and decoding. During decoding, the spurious phrasal segmentation causes different compositions of the same phrasal unit to exist and compete with each other. The same translation is hypothesized with different segmentations.

The unit of translation in N-gram MT is tuple which is a minimal translation unit composed of bilingual source and target cepts. The framework in N-gram model assumes that source and target cepts are generated monotonically. Because of this assumption, source-side has to be linearized. By linearizing the source, N-gram-based SMT throws away useful information about how a particular word is translated with respect to the previous word. This information is rather stored in form of rewrite rules. However, because POS tags are used instead of word forms, a rewrite rule might fail to retrieve the reordering observed during training. Using minimal units during decoding also results in more difficult search problem. A larger beam is required to produce the same translation as the phrase-based MT.

Goal: In this dissertation, we aim to rectify the drawbacks of the N-gram model by coupling reordering information into the tuple-based model, thus removing the need to linearize source and using POS-based rewrite rules. This however, prevents us from using the decoding framework used by the N-gram-based system, because we do not have POS-based rules to construct a search graph. Our search graph is built dynamically, just like in phrase-based MT.

And because we want to remove the hard reordering limit and consider all possible reorderings, this presents a very difficult search problem.

Secondly, although we use minimal translation units just like the N-gram model, we propose a model that has ability to memorize and use lexical triggers. We will therefore try to combine the benefits of both approaches and rectify their drawbacks.

1.2.3 Joint Probability Model

Most previous research has concluded that conditional probability models work better than their counterparts based on the joint probability model. There is a widely held belief that modeling of source-side dependencies do not help because of the data sparsity. The translation model in phrase-based MT makes a context independence assumption. The dependencies on source-side are only handled inside of phrases. Target-side dependencies are handled through monolingual language model.

Goal: One of the aims of this research was to study joint-modeling. Joint models are better than conditional model in theory because they take both source and target context when translating. We would also like to mention a piece of work, our Hindi-to-Urdu machine translation system (Durrani et al., 2010), that was carried out earlier¹, and was one of the motivations of this research. In this work we presented a novel approach to integrate transliteration into Hindi-to-Urdu statistical machine translation. We proposed two probabilistic models, based on conditional and joint probability formulations. In our results we found that the joint probability model performs as well as the conditional probability model.

The **translation model** $p(h_1^n | u_1^n)$ is approximated with a context-independent model:

¹but has been left out of this dissertation to keep the writing coherent

1 Introduction

$$p(h_1^n | u_1^n) = \prod_{i=1}^n p(h_i | u_i) = \prod_{i=1}^n \frac{p(h_i, u_i)}{p(u_i)} \quad (1.1)$$

The joint probability $p(h_i, u_i)$ of a Hindi and an Urdu word is estimated by interpolating a word-based model and a character-based model.

$$p(h_i, u_i) = \lambda p_w(h_i, u_i) + (1 - \lambda) p_c(h_i, u_i) \quad (1.2)$$

The model was built on very small amount of parallel data containing roughly 7000 training sentences. Because of the data sparsity we used a context independent model. A question raised from this work however was that whether using a context and modifying the Equation 1.2 to Equation 1.3 would be helpful. Seeking the answer for this question we followed this line of research to pursue joint modeling.

$$p(h_i, u_i) = \lambda p_w(h_i, u_i | h_{i-m+1}, u_{i-m+1} \dots h_{i-1}, u_{i-1}) + (1 - \lambda) p_c(h_i, u_i) \quad (1.3)$$

1.2.4 Secondary Goals

We also aimed at the modeling of discontinuous and unaligned translation units. Previous research has shown that modeling these phenomenon can help improve the translation quality. One of the aims of this dissertation is to study these artifacts of translation.

1.3 Thesis Organization

This thesis is organized into five chapters. The introductory chapter has been spent on giving a brief overview on machine translation in terms of history and different approaches. We stated some of the problems in phrase-based and N-gram SMT that we aim to solve. The next two chapters are devoted to the literature review of Phrase-based and N-gram based MT respectively. We discuss both the frameworks in terms of translation modeling, reordering and

1 Introduction

decoding. Followed by which we discuss the drawbacks of these techniques. Chapter 3 also gives a comparison of phrase-based and N-gram-based models. In Chapter 4 we present our model discussing different aspects. We present the new reordering framework showing how it can handle both short and long-range reorderings, as well as recursive reorderings with complex patterns. We discuss how our model is able to memorize phrases. We discuss how our operation sequence model is able to handle discontinuous and unaligned source word cepts. We also present different features that we use to help the translation and reordering decisions taken by the decoder during search. Finally we present our decoding framework. The end of this chapter gives a comprehensive evaluation of our model against the state-of-the-art phrase-based and N-gram-based systems. Chapter 5 concludes this thesis by presenting the contributions. It also gives discusses future lines of research extending the work carried out.

2 Phrase-based Statistical Machine Translation

In this chapter we give a comprehensive overview of the state-of-the-art phrase-based SMT in terms of translation modeling, reordering and decoding. We discuss the benefits and drawbacks of phrase-based SMT (PBSMT). The chapter begins by summarizing word-based models developed by IBM in the last decade of 20th century. The drawbacks of word-based translation motivate phrase-based models. Instead of using words, phrase-based models make a shift towards using phrases as a unit of translation. We discuss the main stream phrase translation model (used in Moses), which is built on top of word alignments and uses the noisy channel model. Alongside we also discuss phrase-based systems that are based on joint models. The chapter also discusses the problem of reordering in machine translation. Both distance-based and lexicalized reordering models are discussed. In the end we talk about the major drawbacks of phrase-based SMT.

2.1 Word-based Models

With the increasing availability of the digital text due to rapid proliferation of Internet, machine translation made a transition from rule-based methods to example-based and statistical machine translation (SMT). The first SMT models built on parallel data were based on word-to-word translation. Word-based models were first proposed by the IBM Candide project (Brown et al., 1990, 1993). Word-based models were used both for word alignments and transla-

tion. Although the models are now obsolete for translation, they are still used for the word alignment task and underpin some of the very key concepts such as generative models, expectation maximization and the noisy channel framework that are still used today. Before discussing the IBM models let us first discuss the noisy channel framework and generative models to lay important foundations which will be referred to through the course of this thesis.

2.1.1 Noisy Channel Model

The noisy channel model (NCM) is a very useful framework that has been successfully applied to many areas of problems inside of NLP (such as spell-checking, automatic speech recognition, machine translation etc.) as well as outside of NLP (such as data compression, OCR error correction etc). The theory roots back to the problem in information theory. Say a signal “e” is emitted by a transmitter. It goes through a channel and gets corrupted to noise “f” before getting received at the other end. The problem in information systems is to retrieve the original signal “e” given the noise “f”. In the NCM framework this problem can be factored into two models. One of these models describes the common patterns of how “e” gets scrambled into “f” and the second model defines how a normal signal “e” usually looks like. This formulation has been borrowed to solve problems in many areas including machine translation.

The NCM metaphor in machine translation can be applied as follows. When we see a foreign language “f”, we say that the person actually wanted to write an English sentence “e” but somehow ended up writing a foreign language sentence “f” on the paper. The problem is to decode the English sentence given the foreign sentence. In terms of the original problem the English sentence “e” is the signal emitted by the transmitter that gets scrambled and turns into noise (foreign sentence) “f”. To recover the most likely English translation we try to model two phenomenon. 1) If the sentence is grammatical – $p(e)$: the probability of the English sentence being generated. and 2) how an English sentence turns into a foreign language (say German) sentence “f” – $p(f|e)$ Translation

Model (Brown et al., 1990). In other words 1) models fluency (grammatical correctness and word choices) and 2) models adequacy (the output conveys the same meaning as the input sentence). Combining a language model and a translation model in this manner is called a **Noisy Channel Model**.

Having these two sub-models in hand the remaining problem is to search for an English string that is fluent and adequately represents the foreign sentence. We can use Bayes' Rule to get the most likely English sentence given a German sentence $p(e|f)$ as:

$$p(e|f) = \frac{p(e)p(f|e)}{p(f)}$$

We are interested in the English sentence e that maximizes Bayes' rule equation:

$$\begin{aligned} \operatorname{argmax}_e p(e|f) &= \operatorname{argmax}_e \frac{p(e)p(f|e)}{p(f)} \\ &= \operatorname{argmax}_e p(e)p(f|e) \end{aligned}$$

The $p(f)$ term in the denominator can be dropped because it remains constant across different English sentences.

2.1.2 Generative Models

In the previous section we defined the problem of machine translation. Given a foreign sentence “f” we want to find its translation “e”. We then split the problem of finding $\operatorname{argmax}_e p(e|f)$ into two components, namely **Language Model** and **Translation Model**. The reason why we do this, is that, it is much harder to estimate $p(e|f)$ directly. Therefore we decompose the problem using Bayes' Rule. The translation model assigns a high probability to an English sentence “e” if words in “f” are generally translations of words in “e”. For example for a German sentence: “meistens ist die sache mit einer

entschuldigung abgetan”, the translation model gives a high probability to both the English sentences “mostly the matter ends with an apology” and “mostly the apology with an matter ends” but low probability to “the houses were put on fire”. On the other hand the job of the language model is to give high probability to “mostly the matter ends with an apology” and low probability to “mostly the apology with an matter ends”. True, that the language model will also favor “the houses were put on fire”, but the translation model dis-prefers it because it is not a good translation. The overall decoding problem is to search for an English sentence that maximizes the product of the two probabilities. As can be seen, it is very convenient to decompose the problem into two smaller problems and then combine them to solve the bigger problem.

Although we have split the problem into the two sub problems of estimating $p(e)$ and $p(e|f)$ note that we can still not model these distributions for full sentences because most sentences appear only once or twice even in large corpora. Therefore $p(e)$ and $p(e|f)$ would give very high probability to the sentence “mostly the matter ends with an apology” but might assign a probability of zero to the sentence “mostly the matter ends with an extenuation” because the sentence was never observed during training. In order to overcome this problem we split these processes further into smaller steps of translating one or more words at a time instead of whole sentence and scoring these smaller steps through the probability models. This is helpful because we might have all or at least most of the English and foreign language words in our data but it is impossible to have all possible English and foreign sentences because there can be infinitely many.

One commonly used language model is a tri-gram model which considers a window of the last two words only, when generating the next word. According to this model the probability of the sentence “it is very cold” would be:

$$p(it | \langle s \rangle) * p(is | \langle s \rangle \ it) * p(very | it \ is) * p(cold | is \ very) * p(\langle e \rangle | very \ cold)$$

This method of breaking up the process of generating data into smaller steps

and modeling the smaller steps with probability distributions, and combining the steps in a coherent story – is called **Generative Modeling** (Koehn, 2010).

2.1.3 Summary of the IBM Word-based Models

In this section we will summarize the IBM models that are based on the word-based translation. We will give generative story of each of them followed by its drawbacks. The generative story can be viewed as a string rewriting process that helps us to learn model parameters. Although the problem is to translate from foreign language to English the generative story explains how an English string can be rewritten or scrambled (in terms of NCM metaphor) to a foreign sentence. At the end of this section we will discuss the overall drawbacks of word-based translation models and motivate phrase-based models that are still the state-of-the-art for many language pairs.

Model 1

Model 1 is a simple generative model. Given a German string F (we will use German as foreign language for the rest of the thesis), having length l , Model 1 generates a sentence E with the following stochastic process:

- Model 1 choose a string length m for the English sentence E
- For each English position i , a German word f_j at position j is selected and an English word e_i is generated with a probability through a translation probability $p(e_i|f_j)$.

Model 2

The drawback of Model 1 is that it does not prefer good word order over a bad permutation of words in an output string. The translation output “apology the ends mostly with matter an” is as much probable as “mostly the matter ends with an apology” according to Model 1. A better translation model needs to penalize bad reorderings through some parameter. Model 2 adds an explicit

alignment step to the generative story of the Model 1 to penalize unlikely permutations.

Model 2 has a two step generative process. The first step models how English word can be mapped to German words (same as Model 1) and the second step is the alignment step which models reordering. The alignment step models that i^{th} word in English sentence is mapped to the j^{th} word in the German sentence through a probability distribution $a(j | i k l)$. Which means that i^{th} word in the English sentence is mapped to the j^{th} word in the German sentence. The parameters k and l are the lengths of English and German sentences respectively.

Model 3

Model 1 and Model 2 do not explicitly model the generation of many-to-one mappings from English to German. There can be multiple English words that map to a single German word for example “inflation rate” mapping to “inflationrate”. Modeling of this phenomenon is very useful in translation of agglutinative languages such as German, Turkish and Finnish, that tend to create very long words with derivational morphemes.

Model 3 removes this drawback by adding a fertility model $p(n|f_j)$ to the generative story of Model 2. This model assigns fertility (n) to each German word i.e. the number of English words generated by the German word (f_j). The German words that are mapped to, by one English word get a fertility $n = 1$. Those that generate two English words get a fertility of $n = 2$ and so on. A zero fertility means that this German word is dropped i.e. it does not have a corresponding word on the English side.

Model 4

The drawback of Model 2 and Model 3 is their weak reordering model which is based on the absolute positions of words in German and English sentences. The parameterization is weak, and can not distinguish between good and bad alignments. Secondly learning absolute indexes irrespective of lexicalized word

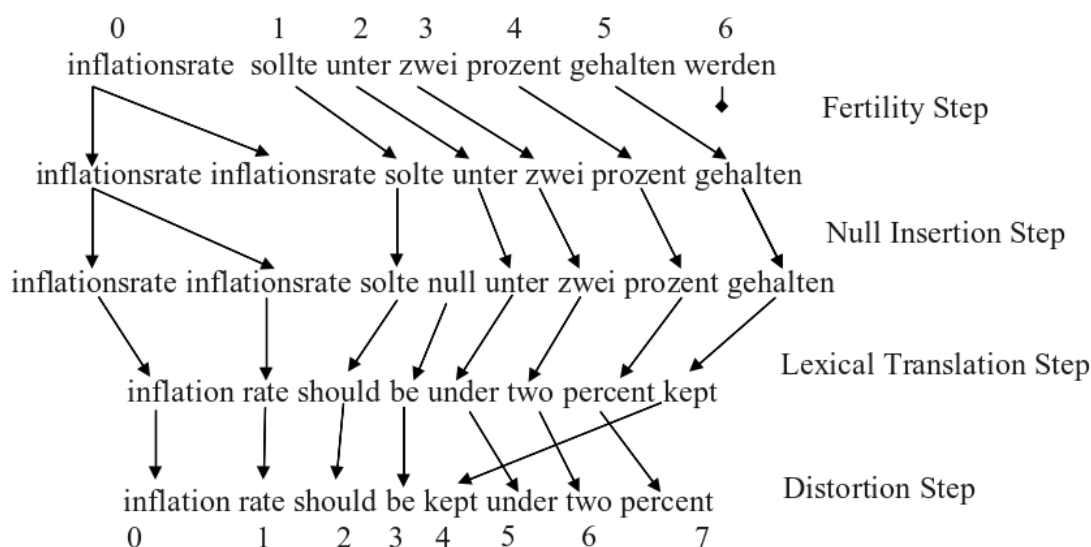


Figure 2.1: Generative Story – Model 3

forms is less helpful for generalizing over unseen test sentences. In order to remedy this problem, Model 4 proposes a relative distortion model. Model 4 learns the placement of a word relative to the placement of the last translated word.

The reordering model learned by Model 3 conditions movement only on German and English indexes and length of sentences. Model 4 introduces lexical forms in its distortion parameters by conditioning on current English (e_j) and previous German word (f_{i-1}). See Figure 2.1, for example. When placing “gehalten – kept”, the parameter learned by Model 3 is estimated as $p(5 | 4 8 7)$. This means that the 5th word in the German sentence maps to the 4th position in the English sentence. Model 4 learns $d_1(-3 | kept, prozent)$. The relative distortion is -3 which suggests the backward movement of the English word “kept” by jumping back 3 positions over the German words “under”, “two” and “percent”.

In order to overcome sparse estimates, word classes are used instead of lexicalized word forms. The parameters are then estimated as $p(d | \alpha(e_j), \beta(f_{i-1}))$,

where d is the relative distortion, $\alpha(e_j)$ and $\beta(f_{i-1})$ are the word class functions. Several different methods have been proposed in literature to define word class functions $\alpha(e_j)$ and $\beta(f_{i-1})$.

Model 5

One drawback of previous models is that they are deficient i.e. they assign probability mass to some events that can not occur in practice. For example we can pick an English word position i that has already been taken by another English word and Model 4 will still assign it a positive probability. In other words nothing prevents Model 4 from selecting the same target position j for every German word to be translated.

Model 5 eliminates the problem of deficiency by proposing an additional parameter that prevents it from placing words in the occupied slots. This is done by keeping track of the vacant positions. The movement is now conditioned on two additional parameters v_{max} and $v_{\odot_{i-1}}$ such as:

$$d_1(v_j \mid \alpha(e_j), v_{max}, v_{\odot_{i-1}})$$

where v_{max} defines the number of available slots in the English output at that point of translation. v_j defines the number of available slots for the interval $[1-j]$ and $v_{\odot_{i-1}}$ defines the available vacancies at the center of the previous cept covered. The conditioning on $\beta(f_{i-1})$, the word class of the previous German word is removed due to the sparsity issues.

2.1.4 Drawbacks of Word-based Translation

In the previous sections we summarized the word-based IBM Models. Word-based models, combined with symmetrization heuristics, show good performance in the word alignment problem, they are, however, not used for translation task anymore. In this section we will enumerate their drawbacks which help us to motivate phrase-based translation.

1. Word-based models do not allow many-to-one or many-to-many alignments. The English words can not be aligned to more than one German word. This restriction hinders the model to learn word correspondences like “habe gemacht – made” which aligns two source words with one target word. Many-to-one and many-to-many alignments turn out to be critical when translating to agglutinative languages such as German or Finnish. Word-based models fail to produce translation units such as “chocolate ice cream – schokoladeneis” (when translating from English to German).
2. Another major drawback of word-based models is their modeling of dependencies. Translation decisions are not conditioned on the previous translations or reorderings. Many words have different possible translations and various meanings in different contexts. For example the most typical translation of the German word “zum” is the preposition “to” as in “zum Bahnhof – to the train station”. But “zum” also translates to the word “for” as in the phrase “zum beispiel – for example” and gets deleted in phrase “zum mitnehmen – takeaway”.
3. The fertility model for generating multiple English words from a German word often breaks due to independence assumption. In the above example “zum” has fertilities one and zero respectively. Lets add another example here in which “zum” translates to “to the” in the phrase “von einem Haus **zum anderen** – from one house **to the** other”. In this case “zum” has a fertility two. Word-based model will mostly prefer the translation “zum – to” with a fertility $p(1|zum)$ because this is the most frequent translation and fertility for the word “zum”.
4. Word-based models have no ability to memorize idiomatic expressions such as “in den sauren apfel beißen – bite the bullet” which it translates as “the sour apple bite”.
5. The parameters learned by the reordering models do not generalize well during decoding. For example Model 4 learns the reordering parame-

ter $d_1(-3 | kept, prozent)$ according to which German word translates to “kept”, and is placed in a position, 2 slots before the target position for the word “prozent”. This can easily fail for a test sentence like “inflationrate sollte zwischen 2 bis 6 prozent gehalten werden – inflation rate should be kept between 2 to 6 percent”, which requires “kept” to be placed 4 slots before the target position of the word “prozent”. In a nutshell words can follow infinitely many permutations. A reordering model based on absolute or relative word positions remains weak when translating words with unseen patterns.

6. Insertion of English words is learned by introducing an imaginary token null, during word alignment process and by learning a probability distribution p_{ins} over all inserted English words. During decoding a decision is made at each translation step, with a probability p_{ins} , whether to insert an English word. Because IBM models do not learn which insertions are more probable than others there is no evidence other than the language model probability to score these. Another problem is that there can be a large number of word insertions observed during training and it is computationally challenging to try them all at each translation step.
7. Word deletion is modeled through the fertility model by learning parameters like $p(0|zum)$. Again whether to delete a German word or to translate it to some English word depends upon surrounding translations. For example the German pronoun “Sie” always gets dropped preceded by a verb such as “schreiben Sie - write”. In other contexts it will translate into “you”.

2.2 Phrase-based Models

Phrase-based translation overcomes the problems of word-based model by making a shift towards using phrases rather than words as a unit of translation (Och and Ney, 2004; Koehn et al., 2003). It provides a simpler yet powerful

translation mechanism by learning larger chunks of translation called phrases which are not necessarily linguistic constituents.

Given a bilingual sentence pair, we segment the German and English sentences into phrases such that words in phrases are a contiguous sequence of strings. Each German phrase aligns to exactly one English phrase and vice versa. Finally the phrases are reordered. Phrasal reorderings are modeled through a distance-based ordering model. A possible segmentation of the sentence pair “inflationrate sollte unter zwei prozent gehalten werden – inflation rate should be kept under two percent”, is shown in Figure 2.2.

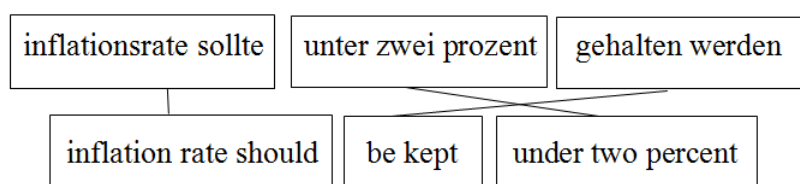


Figure 2.2: Phrase-based Machine Translation

2.2.1 Advantages of the Phrase-based Model

Now we discuss how using phrases as a unit of translation overcomes the problems enumerated in the previous section.

1. Phrase-based models can learn many-to-many and many-to-one mappings that are necessary to produce translations such as “gehalten werden – be kept” and “habe gemacht – made”. This also helps to get rid of the fertility parameters.
2. Memorizing larger translation units such as “zum beispiel – for example” and “zum kaffeetrinken – to drink coffee” enables phrase-based model to learn local dependencies.
3. Insertions and deletions inside of the phrases are handled well. For example by memorizing a phrase such as “will nicht schlafen – does not want

to sleep”, the model learns insertion of verbal auxiliary “does”. Similarly the model can learn deletion of German words by memorizing phrases such as “kommen Sie mit – come with me” having unaligned German pronoun “Sie”. Recall that “Sie” gets dropped in such constructions but is translated to “you” in other context. Learning insertion and deletions inside of phrases helps the phrase-based system to do away with the complex notion of null words and separate probability distributions for handling such cases.

4. The ability to memorize larger units also helps the phrase-based models to produce idiomatic expressions such as “andere Länder, andere Sitten – when in Rome, do as the Romans”.
5. Movement of words inside of the phrases are captured independently of the reordering model. This phenomenon is particularly useful for short distance reorderings such as flipping noun and adjective in French-to-English translation task such as “beauté noire – black beauty”.

To summarize, phrase-based models are simple and effective. They are a closer approximation of how humans translate. Using larger chunks as a unit of translation saves us the separate modeling of different aspects of translation such as fertilities, insertion and deletion. Many models for translation have been proposed to date but phrase-based SMT still remains the best performing statistical machine translation system for many language pairs.

2.2.2 Translation Models

In this section we discuss two probability models for learning phrase translation. One of these is based on the commonly used noisy channel model framework and the other is based on a joint probability formulation.

Conditional Probability Model

Recall Bayes' theorem from Section 2.1.1. Given a German sentence “f” we want to search for an English translation “e” that maximizes the following equation:

$$\operatorname{argmax}_e p(e|f) = \operatorname{argmax}_e p(e)p(f|e)$$

In this section we discuss how to estimate the translation model $p(f|e)$. Given a bilingual sentence pair $\langle f, e \rangle$, we choose a phrasal segmentation:

$$S_x = \{(F_1, F_2, \dots, F_n), (E_1, E_2, \dots, E_n)\}$$

such that every English phrase E_j aligns with exactly one German phrase F_i and vice versa. Each F_i and E_j contains one or more contiguous German and English words respectively. The translation probability of a sentence given the segmentation S_x is given as:

$$p(f|e) = \prod_{i=0}^n p(F_i|E_j) d(\alpha_i - \beta_{i-1} - 1)$$

where $p(F_i|E_j)$ is the phrase translation probability and $d(\alpha_i - \beta_{i-1} - 1)$ is the distance-based reordering model to penalize unjustified reorderings. This formulation can be defined as the basic phrase-based model.

The phrase-based model makes a phrasal independence assumption according to which all phrases are translated independently of each other. The probability of a phrase pair $\langle F_i, E_j \rangle$ is estimated as:

$$p(F_i|E_j) = \frac{\text{count}(F_i, E_j)}{\text{count}(E_j)}$$

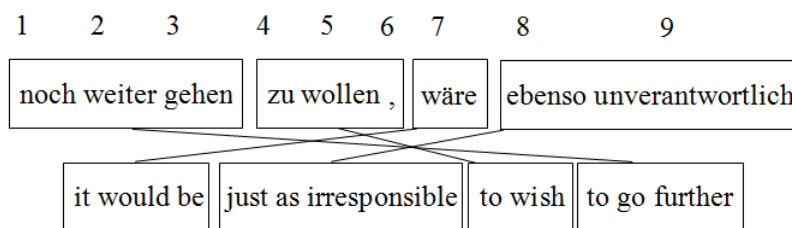


Figure 2.3: German–English Sentence Pair with a Phrasal Segmentation

Curr. Phrase F_i	Prev. Phrase F_{i-1}	$\alpha(i)$	$\beta(i)$	$\alpha_i - \beta_{i-1} - 1$
wäre	–	7	0	+6
ebenso unverantwortlich	wäre	8	7	0
zu wollen ,	ebenso unverantwortlich	4	9	-6
noch weiter gehen	zu wollen ,	1	6	-6

Table 2.1: Distortion Model for Phrase-based SMT

The quality of translation depends upon how good the probability estimates for $p(F_i|E_j)$ are. For smaller corpora containing a few thousand sentences, the statistics might be too sparse for the phrase-based system to produce high quality translation. It can nevertheless fall back to word-based translation since the minimal phrasal unit will have at least one German and English word pair.

Reordering Model The term $d(\alpha_i - \beta_{i-1} - 1)$ is the distance-based reordering model as used in phrase-based model. α_i denotes the index of the first word of the German phrase F_i and β_{i-1} represents the index of the last word of the German phrase F_{i-1} . F_{i-1} represents the previously translated German phrase. Consider the German-English sentence pair with a phrasal segmentation and alignment as shown in Figure 2.3. Table 2.1 gives the reordering distance for each phrase F_i .

A reordering distance of 0 means monotonic reordering i.e. the current phrase was translated monotonically with respect to the previous phrase. A reordering distance of +6 means that we jump ahead 6 words in a sequence to translate

a German phrase. Similarly -6 means jumping back over 6 words from the current position to cover a German phrase. See Table 2.1 for the examples of each of these. Instead of using a probability distribution, a decaying cost function $d(x) = \gamma^{|\alpha_i - \beta_{i-1} - 1|}$ is used where $\gamma \in [0, 1]$ (Marcu and Wong, 2002; Koehn et al., 2003). This function penalizes longer distance reorderings.

Phrase Extraction Method Different methods to extract phrases from a parallel corpus have been proposed in the literature. Some of these are based on extracting phrases from the word alignments (Tillmann, 2003; Zhang et al., 2003; Zhao and Vogel, 2005). While others extract phrases directly from the sentenced-aligned corpora, using pattern mining (Shin et al., 1996), matrix factorization (Goutte et al., 2004) and based on EM training (Marcu and Wong, 2002).

Symmetrization The phrase extraction method (Koehn et al., 2003) used in Moses¹ – a state-of-the-art phrased-based machine translation system is based on the word alignments. Given the sentenced-aligned parallel data the first step is to obtain word alignments. This is done by running GIZA++ (Och, 2000), a toolkit for word alignments, that uses IBM Models or any other word alignment method e.g. Liang et al. (2006); Fraser and Marcu (2007).

The IBM models do not align English words to more than one German word, this problem needs to be rectified. This is done by symmetrization. GIZA++ is run twice, first using German as source and English as target, then flipping the direction. Given the sentence pair:

noch weiter gehen zu wollen , wäre ebenso unverantwortlich

it would be just as irresponsible to wish to go further

GIZA++ learns German-to-English and English-to-German alignments as shown in Figure 2.4.

¹We will refer to Moses (<http://www.statmt.org/moses>) as the standard phrase-based model in this thesis

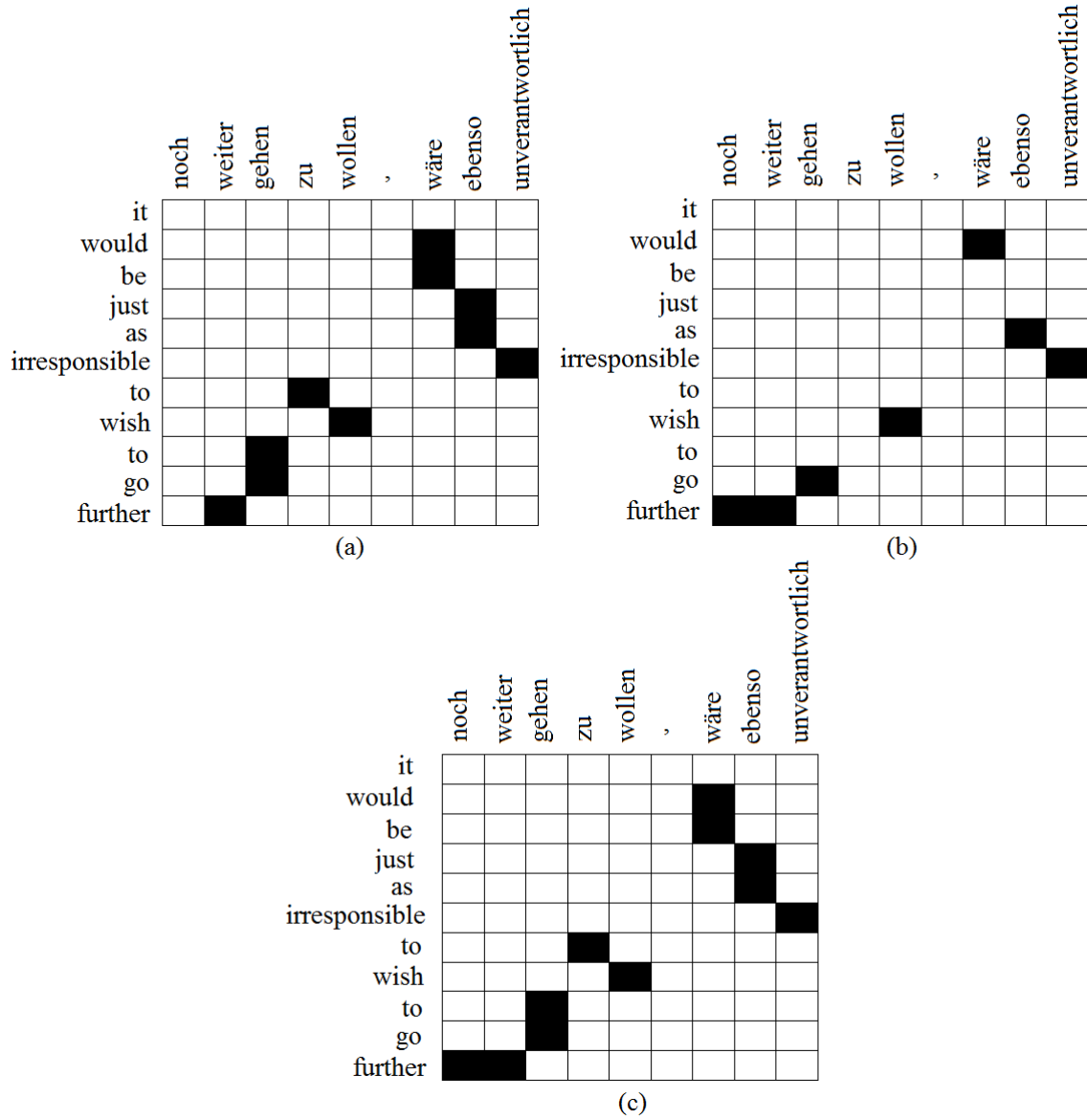


Figure 2.4: Word Alignments (a) German-to-English, (b) English-to-German, (c) Union of (a) & (b)

A number of heuristics have been proposed for symmetrization namely, intersection, union, grow etc. The best method depends upon the translation task and whether precision is more desirable or recall. Intersection produces high precision alignments whereas union produces high recall alignments. Sparse alignments tend to work better for Chinese-to-English translation, therefore intersection is used for symmetrization. Union gives better performance for Arabic-to-English task (Fraser and Marcu, 2007) because recall is more important in this case. For most other language pairs, grow-diag-final-and gives best performance. Symmetrization based on union for the given example is shown in Figure 2.4(c).

Extraction Now that we have many-to-many and many-to-one alignments along with one-to-one and one-to-many alignments, we can extract all phrase pairs that are consistent with the word alignments. A phrase pair $\langle F_i, E_j \rangle$ is said to be consistent with a word alignment A if the words in F_i are only aligned with words in E_j and vice versa and both F_i and E_j are continuous. Unaligned German and English words can be a part of a phrase F_i and E_j as long as they satisfy the constraint that words in a phrase can not be discontinuous. Given the word alignments shown in Figure 2.4(c), we extract the phrases shown in Table 2.2. Note phrases like “it would be – wäre”. Although “it” is not aligned to a word on the German side it can still be a part of this phrase because the English phrase is still a continuous sequence of words. On the other hand we can not extract phrase like “it further – noch weiter” because “it” and “further” are discontinuous with respect to each other.

Phrase-based model extracts both short and longer phrases. Longer phrases are helpful because they capture larger context and more dependencies. On the other hand shorter phrases are more frequent and generalize well to unseen sentences. Although it is computationally possible to extract phrases of unbounded length during training, results have shown that, due to data sparsity, phrases applied during decoding are fairly short (Callison-Burch et al., 2005; Zhang and Vogel, 2005). Therefore only phrases with 6 or less German words are extracted.

2 *Phrase-based Statistical Machine Translation*

English	German
it would be	wäre
it would be just as	wäre ebenso
it would be just as irresponsible	wäre ebenso unverantwortlich
it would be just as irresponsible	, wäre ebenso unverantwortlich
it would be just as irresponsible to wish	wollen , wäre ebenso unverantwortlich
it would be just as irresponsible to wish to	zu wollen , wäre ebenso unverantwortlich
it would be just as irresponsible to wish to go	gehen zu wollen , wäre ebenso unverantwortlich
it would be just as irresponsible to wish to go further	noch weiter gehen zu wollen , wäre ebenso unverantwortlich
would be	wäre
would be just as	wäre ebenso
would be just as irresponsible	wäre ebenso unverantwortlich
would be just as irresponsible	, wäre ebenso unverantwortlich
would be just as irresponsible to wish	wollen , wäre ebenso unverantwortlich
would be just as irresponsible to wish to	zu wollen , wäre ebenso unverantwortlich
would be just as irresponsible to wish to go	gehen zu wollen , wäre ebenso unverantwortlich
would be just as irresponsible to wish to go further	noch weiter gehen zu wollen , wäre ebenso unverantwortlich
just as	ebenso
just as irresponsible	ebenso unverantwortlich
irresponsible	unverantwortlich
to wish	zu wollen
to wish	zu wollen ,
to wish to go	gehen zu wollen
to wish to go	gehen zu wollen ,
to wish to go further	noch weiter gehen zu wollen
to wish to go further	noch weiter gehen zu wollen ,
to	zu
to go	gehen
to go further	noch weiter gehen
to go	gehen
to go further	noch weiter gehen
further	noch weiter

Table 2.2: Extracted Phrases

Joint Probability Model

The models that we have discussed so far, tell a story about how to map German sentences to English sentences. Joint models instead try to capture how German and English sentences can be simultaneously generated. Joint models thus capitalize on mutual information unlike conditional model which does not model anything about the distribution of German. The two main pieces of work on joint models for phrase-based machine translation are based on EM training of phrase models (Marcu and Wong, 2002) and phrase-based unigram model (Tillmann and Xia, 2003) building on word alignments like the standard phrase-based model discussed in the previous section. Let us briefly discuss each of these:

Joint Probability Phrase-based Model with EM Training Most phrase-based models build their phrase inventory on top of word alignments. The model proposed in Marcu and Wong (2002), instead estimates phrase tables directly from the sentence aligned parallel corpus using EM training, just as done in word-based IBM models. The key difference is that the generative story explains how German and English are jointly generated, rather than how English gets converted into German. Secondly the restriction of at most one German word aligned with any English word is removed.

German and English phrases are generated in form of **concepts**. Each concept c_i consists of a phrase pair $\langle F_{c_i}, E_{c_i} \rangle$ that are hypothesized as translation of each other. A bag of concepts which completely covers the sentence pair $\langle f, e \rangle$ is denoted by a segmentation $S_x = c_1, c_2 \dots, c_n$. Each concept c_i is generated with a probability $p(F_{c_i}, E_{c_i})$. The probability of a segmentation S_x is given as:

$$p(S_x) = \prod_i^n p(F_{c_i}, E_{c_i})$$

There can be many possible phrasal segmentations for a sentence pair $\langle f, e \rangle$. Let S be the set of all possible segmentations for a sentence pair. The overall translation probability is calculated by summing over them.

$$p(e, f) = \sum_{S_x \in S} p(S_x)$$

$$p(e, f) = \sum_{S_x \in S} \prod_i^n p(F_{c_i}, E_{c_i})$$

Marcu and Wong (2002) also present an extension of this model which adds a distortion parameter to penalize unlikely phrase alignments. The distortion model takes into account absolute word positions just like IBM Model 3.

The generative story in word-based models restricts the number of German and English words generated in a step to be 1. Removing this restriction in the current model, exponentially increases the number of alignments that can be hypothesized between the two sentences making the training process intractable. The complexity of the phrase alignment problem is NP-complete (DeNero and Klein, 2008). This problem has been addressed in Birch et al. (2006) by constraining the phrase alignments with the help of word alignments obtained from IBM model. However, for the larger data sets their performance is behind the standard phrase-based models.

Joint Probability Phrase-based Model based on Word Alignments The model proposed in (Tillmann and Xia, 2003) is also based on joint probability formulation but extracts phrases from the output of an HMM word alignment model (Vogel et al., 1996). The HMM viterbi training is carried out twice by training in both translation directions. The alignments are then symmetrized just as discussed in the standard phrase-based model using one of the proposed heuristics.

Previous research has shown that the joint probability phrase-based model tends to work better for smaller data sets (Birch et al., 2006).

2.2.3 Lexicalized Reordering Models

Reordering is one phenomenon that makes the translation of some language pairs harder than others in statistical machine translation. For example translation between Hindi and Urdu or between Thai and Lao is straightforward because these language pairs have the same word order. On the contrary translation between German and English or Japanese and English language pairs is more difficult because of their different syntactic structures.

IBM Models: The reordering mechanism in IBM models was designed for the translation of language pairs like French–English, Spanish–English etc. that have relatively similar syntactic structure and exhibit mostly short distance movements such as swapping of noun and an adjective in French-English task. The overall idea was that the translation model is mainly responsible for the adequacy, i.e. how well the hypothesized English sentence represents the words in the given German sentence. The fluency of the output sentence is judged by the language model. Reordering models based on absolute word positions were added inside the translation models to take some burden from the language model and filter out some unlikely word permutations. However, the commonly used language model in statistical machine translation does not span beyond a window of 3–5 words. This window might be too small for judging the overall fluency of the English sentence. Therefore it is unfair to expect the language model to take all the responsibility for the word order.

Model 4, for example learns the parameter $d_1(-3 \mid \textit{kept}, \textit{prozent})$ according to which German word translates to “kept”, and is placed in a position, 2 slots before the target position for the word “prozent”. This can not generalize to the test sentence “inflationrate sollte zwischen 2 bis 6 prozent gehalten werden – inflation rate should be kept between 2 to 6 percent”, which requires “kept” to be placed 4 slots before the target position of the word “prozent”. The conditioning of the position index (relative or absolute) on the words “kept” and “prozent”, results in sparse estimates. Model 4, therefore uses word classes

to overcome this problem. This however, results in a poor reordering model which is not discriminative enough.

Lexicalized Reordering: As the field has evolved over the last decade or so, research has gone in the direction of learning lexicalized models. Rather than using reordering distances with conditioning on word classes, three broader orientation classes namely monotonic, swap and discontinuous are defined. We now learn how a phrase pair was translated with respect to the previously translated phrases. The orientation model of Tillman (2004) and Tillmann and Zhang (2005) defines the orientation of a phrase pair $\langle F_i, E_j \rangle$ as follows:

1. Monotonic if $F_{a(j-1)}$ and $F_{a(j)}$ ² are adjacent and are in the same order as E_{j-1} and E_j . For example orientation of the phrase “ebenso unverantwortlich – just as irresponsible” is monotonic with respect to the phrase “wäre – it would be” (See Figure 2.5).
2. Swap if $F_{a(j-1)}$ and $F_{a(j)}$ are adjacent but are in an opposite order as E_j and E_{j-1} . For example orientation of the phrase “noch weiter gehen – to go further” is swap with respect to the phrase “zu wollen , – to wish”.
3. Discontinuous if $F_{a(j-1)}$ and $F_{a(j)}$ are not adjacent to each other. For example orientation of the phrase “zu wollen , – to wish” is discontinuous with respect to its previously translated German phrase “ebenso unverantwortlich – just as irresponsible”.

See Figure 2.6 and Table 2.3 for illustration of the example shown in Figure 2.5.

The lexicalized reordering model used in standard phrase-based SMT uses the same orientations (Koehn et al., 2005) however, with a slight difference. Although the orientations are being learned for a phrase pair, the decision of

²The mapping function $a(j)$ aligns the English phrase E_j to the German phrase F_i as $F_i = F_{a(j)}$

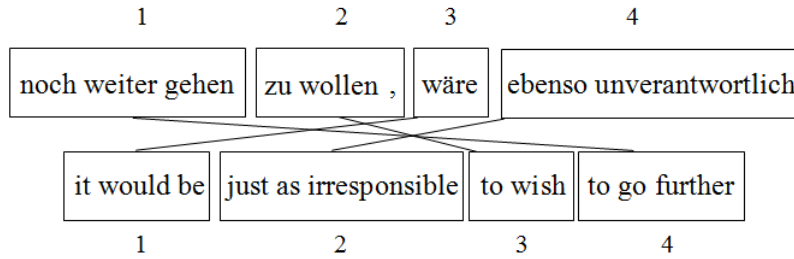


Figure 2.5: German–English Sentence Pair with a Phrasal Segmentation

Current	Previous	Orientation
wäre – it would be	–	(d)iscontinuous
ebenso unverantwortlich – just as irresponsible	wäre – it would be	(m)onotonic
zu wollen , – to wish	ebenso unverantwortlich – just as irresponsible	(d)iscontinuous
noch weiter gehen – to go further	wollen , – to wish	(d)iscontinuous

Table 2.3: Orientation Model for Phrase-based SMT – Current Phrase Pair $\langle F_{a(j)}, E_j \rangle$, Previous Phrase Pair $\langle F_{a(j-1)}, E_{j-1} \rangle$

which orientation to choose depends upon word alignments. This is because the phrasal segmentation is unknown at the training time.

For each extracted phrase pair we check whether there is an alignment point to its top left or top right in the alignment matrix. If there is an alignment point to its top left, the orientation is monotonic. If there is an alignment point to its top right, the orientation is swap. Otherwise the orientation is discontinuous. An illustration of this is shown in Figure 2.7. Phrases are marked with boxes (as previous) and alignment points in the matrix are shaded as gray.

Notice that the orientation for the phrase pair “noch weiter gehen – to go further” as defined by Tillman’s (Phrase-based) reordering model is swap where as it is discontinuous in Koehn’s (word-based) reordering model. Although word-based alignments are used during training in Koehn’s method, they also use phrasal alignments during decoding, because the phrasal segmentation is

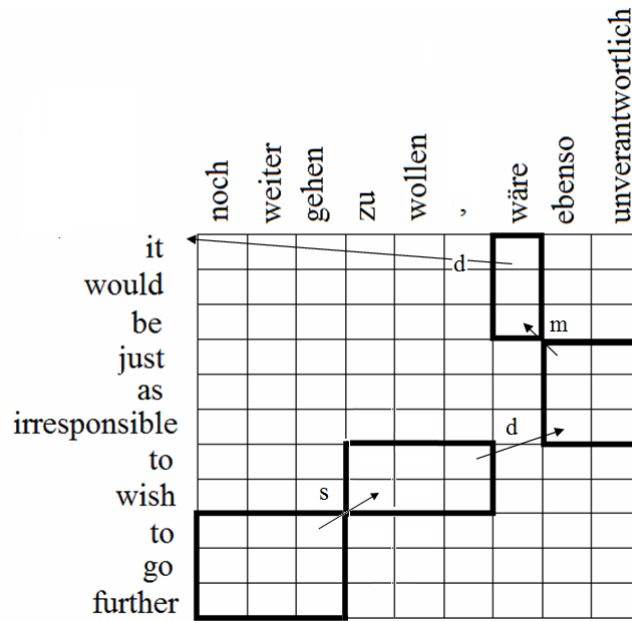


Figure 2.6: Lexicalized Reordering Model – Orientations (m)onotonic, (s)wap, (d)iscontinuous

known at that time. This however, results in a mismatch between training and decoding.

The reordering probability is calculated by conditioning on the German and English phrases under consideration as:

$$p_r(\textit{orientation} | F_{a(j)}, E_j) = \frac{\textit{count}(\textit{orientation}, F_{a(j)}, E_j)}{\textit{count}(F_{a(j)}, E_j)}$$

Similar models are proposed in Ohashi et al. (2005). They split the discontinuous orientation into two categories namely right discontinuity and left discontinuity. Different orientations according to these models are shown in Figure 2.8. Moreover, instead of conditioning on German and English phrases they condition on the more general POS tags for the previous German and English phrase pairs. Each phrase is represented by one tag. Different heuristics

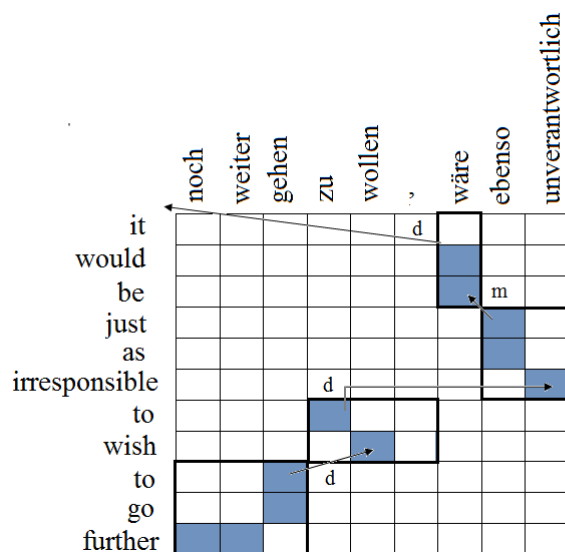


Figure 2.7: Lexicalized Reordering Model (Koehn et al., 2005)

for choosing POS tag of a phrase have been discussed. Nagata et al. (2006) conjecture that estimating the reordering model from the relative frequencies of German and English phrases result in very sparse statistics. In order to remedy this problem they propose two models based on N-best phrase alignments and grouping the phrases into 20 clusters.

The lexicalized reordering models (Tillman, 2004; Koehn et al., 2005) provide a substantial improvement over distance-based linear reordering models by handling short distance non-local movements where two phrases get swapped. However, they remain weak on handling long distance reorderings which motivates hierarchical and syntax-based machine translation. Coming back to the example shown in Figure 2.3, the clause “it would be just as irresponsible” should swap with the subordinate clause “to wish to go further”. The idea of recursively merging adjacent phrases into bigger blocks and deciding orientation of a phrase based on the bigger block is proposed in Galley and Manning (2008). The orientation of a phrase is decided with respect to the block of previously translated phrases by looking at the top-left and top-right corners

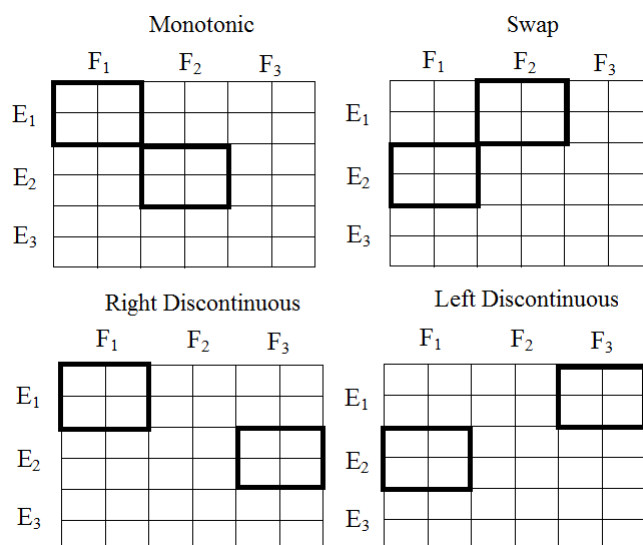


Figure 2.8: Lexicalized Reordering Model (Ohashi et al., 2005)

a phrase in alignment grid just as before. The only difference is that instead of looking at alignment points (like Koehn) or the previous phrase (like Tillman) this method looks at a block of merged phrases. Every step tries to merge the current phrase with a block of previously translated phrases if they are adjacent (monotonic or swap). See Figure 2.9 for illustration. Blocks containing more than one phrase are marked with a dotted line. The orientation of a phrase is decided with respect to the block. As a result of this modification, the orientation of the phrase “zu wollen , – to wish” is swap (rather than discontinuous as in previous models). The orientation is defined with respect to the block formed by combining the adjacent phrases “wäre – it would be” and “ebenso unverantwortlich – just as irresponsible”.

In their advanced model Galley and Manning (2008) also propose to split the category “discontinuous” into right and left discontinuities. Moreover they also learn forward orientations along with the backward orientations. While learning a forward orientation model we consider the next German phrase (or block of phrases) that are yet to be translated. Instead of looking at the top-

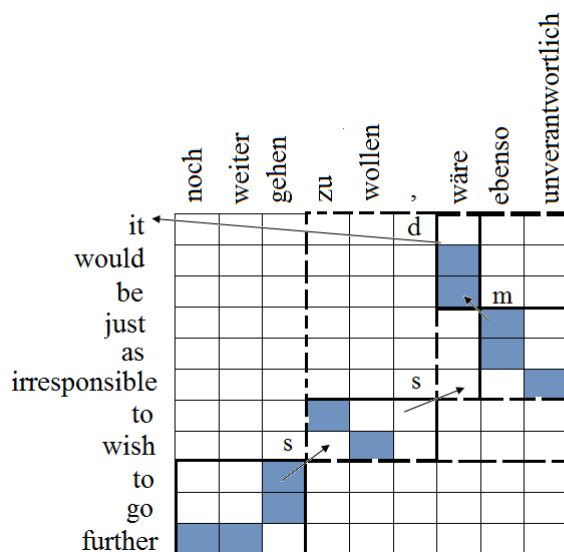


Figure 2.9: Lexicalized Reordering (Galley and Manning, 2008) – Backward Orientation Model

right and top-left corners of a phrase we now look at the bottom right and bottom left corners. If there is a block (of one or more phrases) at bottom right corner, the forward orientation will be monotonic. If there is a block at the bottom left corner, the forward orientation will be swap. Otherwise it will be discontinuous. See Figure 2.10 for illustration. Finding forward orientation during training is trivial because we know the German and English strings with some phrasal segmentation in advance. But during decoding we can only accurately estimate orientation for the previous German phrases (backward orientation), because we do not know ahead of time how the remaining part of the German sentence will be translated and what segmentation will be chosen. To overcome this problem approximations are done during decoding. Table 2.4 shows forward and backward orientation models as learned by Koehn (word-based), Tillman (phrase-based) and Galley’s (hierarchical) reordering models. The hierarchical reordering model gives a statistically significant improvement over word-based and phrase-based models for Chinese-to-English and Arabic-

to-English translation tasks (Galley and Manning, 2008).

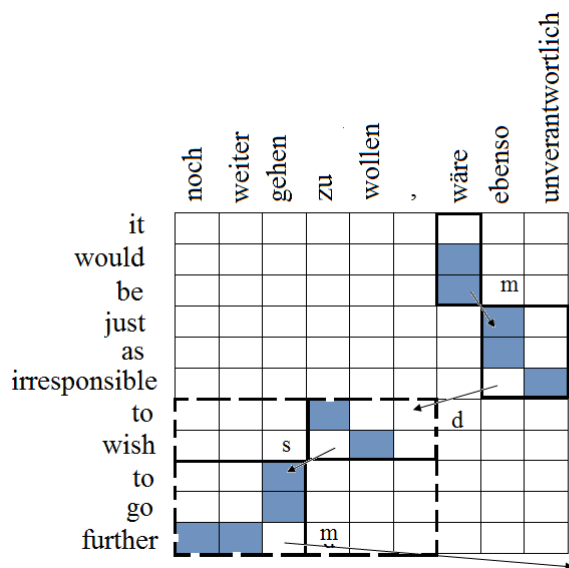


Figure 2.10: Lexicalized Reordering (Galley and Manning, 2008) – Forward Orientation Model

Phrase Pair $\langle F_{a(j)}, E_j \rangle$	Word-based		Phrase-based		hierarchical	
	Back	Fwd	Back	Fwd	Back	Fwd
wäre – it would be	d	m	d	m	d	m
ebenso unverantwortlich – just as irresponsible	m	d	m	d	m	d
zu wollen , – to wish	d	d	d	s	s	s
noch weiter gehen – to go further	d	d	s	d	s	m

Table 2.4: Forward and Backward Orientation Models for Phrase-based SMT

A lexicalized reordering model has also been used for SMT models other than the phrase-based system. Crego and Yvon (2010) used lexicalized reordering with tuple-based N-gram SMT (Mariño et al., 2006) (see the next chapter for details of N-gram-based SMT). They merge the monotonic and the swap categories to form a joint category known as consecutive along with using right and left discontinuous categories.

2.2.4 Log-linear Model

Machine translation is a very difficult problem. Given the limited knowledge extracted from the parallel data it is impossible to model all the reasons why humans choose to translate one way or another. But we can add many knowledge sources that help to improve translation. For example when we pick a phrase for translation we can give additional probability to that phrase if it is a linguistic constituent. We can prefer long phrases over short phrases because longer phrases cover more dependencies or short phrases over long because their estimates are more reliable. We can add another language model which is based on a different genre of text. To add all kinds of different knowledge sources we shift from generative models to discriminative training.

The generative models already break the translation process into translation and language model components. The translation model can be further broken into lexical probabilities and a reordering model. In a discriminative model each of these model components is viewed as a feature. We can add any relevant and useful information inside of a translation process as a feature. Different features such as a target-side parse which might be difficult to integrate during decoding can be added as a post-processing feature for the re-ranking of n-best translations. The first application of discriminative training in statistical machine translation was proposed by Och and Ney (2004) which redefines $p(e|f)$ as:

$$\operatorname{argmax}_e p(e|f) = \operatorname{argmax}_e \left\{ \sum_{j=1}^J \lambda_j h_j(f, e) \right\}$$

where $h_j(f, e)$ is a feature function such as the log probability of the reordering or the language model etc. and J is the number of feature functions.

Some features are more reliable than others, for example monolingual data is available in much more abundantly than bilingual data. Therefore we can have very reliable statistics for the monolingual language model feature. This means that if the language model feature supports a translation hypothesis we should

give it a higher weight than any other feature with less reliable estimates. In order to achieve this, we assign a feature weight λ_j to each feature $h_j(f, e)$. Several algorithms have been proposed for the tuning of feature weights. More commonly used methods are city-block search (varying one parameter at a time) as used in MERT (minimum error rate training) (Och, 2003) Powell Search, and the Simplex algorithm (Nelder and Mead, 1965).

Optimization of Lambda Weights

The process of parameter optimization is iterative. We pick a set of sentences that from the same domain as that of the test set, and call it dev-set (development set). The next step is to run the decoder (discussed in the next section) on dev-set with some random initial feature weights. The decoder then produces a list of N-best translations for all the sentences in the dev-set. The list of N-best translation is input to the parameter optimization algorithm (Powell Search, Simplex etc.), which re-ranks the n-best list such that it maximizes on some evaluation metric³ with respect to the English part of the dev-set. The re-ranking of N-best translation list is done by adjusting the feature weights. The decoder is then run on the dev-set with the optimized feature weights to produce another set of N-best translations. Parameter optimization is now done on the N-best lists from all previous iterations. This process is repeated until the feature weights converge or for a specified number of iterations. See Figure 2.11 for an illustration.

Features used in Standard Phrase-based SMT

The features that are typically used in phrase-based systems(Koehn et al., 2005) are:

1. English-side Language Model

³Lots of metrics have been proposed for the automatic evaluation of statistical machine translation such as WER (word error rate), BLEU (Papineni et al., 2002), NIST (Dodington, 2002), METEOR (Banerjee and Lavie, 2005) etc

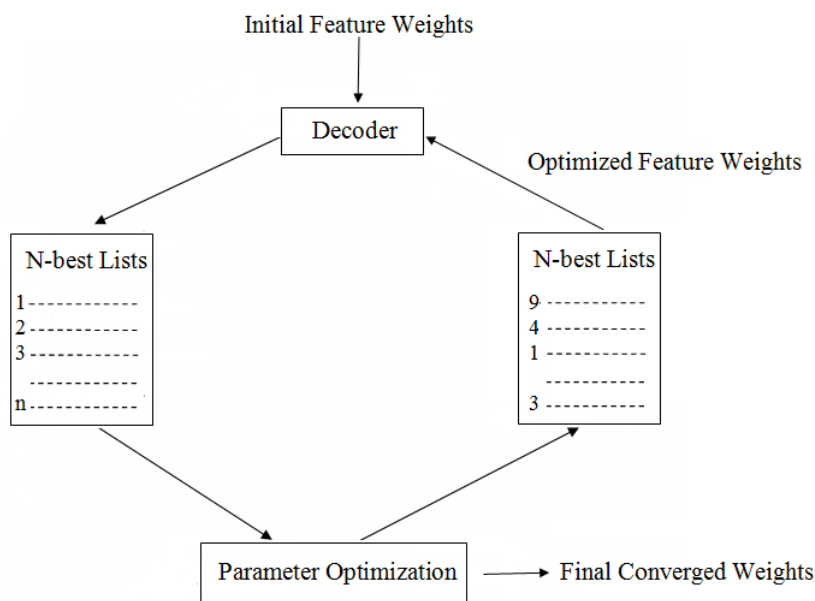


Figure 2.11: Learning Feature Weights

2. Phrase Translation Probability conditioned in both directions $p(F_i|E_j)$ and $p(E_j|F_i)$
3. Lexical Translation Probability – measures reliability of a phrase pair based on word-alignments. Word probabilities are also conditioned in both directions.
4. Word Penalty – counts the number of words produced in a translation output. When assigned a negative weight, it prevents the system from producing too short sentences.
5. Phrase Penalty – counts the number of phrases used to produce the output sentence. It controls the trade off between using shorter and longer phrases.
6. Linear Reordering Penalty – dis-prefers long distance reorderings.

7. Lexicalized Reordering Parameters - Discussed in the previous section. Each orientation class (monotonic, swap and discontinuous) is used as a separate feature. Forward and backward orientations constitute separate features. This results in 6 total features. Monotonic features for example can be defined as: $f_m = \sum^n \log p(o_i = M | \dots)$.

2.2.5 Search

After tuning the feature weights for each model component, the remaining problem in machine translation is to decode the German sentence and produce an English output which maximizes the model score. Searching for such a sentence is a hard problem, because there can be an exponential number of ways to translate a source sentence in terms of word reordering. Knight (1999) mapped the decoding problem into a Traveling Salesman Problem and showed that search in SMT, for the models that we have discussed so far, is an NP-complete problem. An approximate solution is obtained by putting constraints on the word order and by applying pruning strategies.

The state-of-the-art decoder described in Koehn et al. (2007) uses beam search to build up the translation from left to right (Tillmann and Ney, 2003; Koehn, 2004a). Given a German sentence, the first step is to extract all possible phrases with their possible English translations and feature values. The decoder then initializes a set of n stacks where n is the number of words in the German sentence. The stacks are arranged such that each stack i represents hypotheses that have already translated i many German words. Our goal is to find the best scoring hypothesis in stack n , that has translated all German words.

A hypothesis maintains several pieces of information such as, the German word(s) it translates, the English translation for those German words, a back pointer to the parent hypothesis, coverage vector (German words so far covered by this hypothesis and its predecessor hypotheses), cost for each component feature in the model etc. Figure 2.12 shows a sample hypothesis.

German : noch weiter gehen
English : to go further
Coverage Vector : 111000000
Features Values: Language Model : $p(E_j e_1 \dots e_m) = -0.15$ Translation Model : $p(E_j F_i) = -0.45$ Translation Model : $p(F_i F_j) = -1.30$ Translation Model (Lex) : $p(E_j F_i) = -0.33$ Translation Model (Lex) : $p(F_i E_i) = -0.76$ Word Penalty : -3 Phrase Penalty : -1 Distance Penalty : 0 Back Orientation : $p(\textit{monotonic} E_i, F_j) = -0.23$ Forward Orientation : $p(\textit{monotonic} E_i, F_j) = -1.35$
Cost: -8.57
Previous Hypothesis : Back Pointer
Future Cost Estimate: -23.34

Figure 2.12: A Sample Hypothesis

Hypothesis Extension

A hypothesis is extended by picking a translation phrase whose words are not covered yet, with its translation from the inventory of extracted phrases. All the feature values are computed with respect to the extending hypothesis. For example if we are extending the above hypothesis with the phrase “wäre – would be”, the tri-gram language model probability would be:

$$p_{tm} = p(\textit{would}|go\ further) * p(\textit{be}|further\ would)$$

Similarly the backward orientation probability $p(\textit{discontinuous}|wäre, would\ be)$ and distance penalty $d(wäre, zu)$ is computed. After calculating the final cost of a hypothesis it is placed in a stack i such that $i =$ number of German words covered by this hypothesis and its predecessors (all the parent hypotheses). In this case we add to stack 5 since the two hypotheses cumulatively cover 5

words “noch weiter gehen zu . . . wäre . . .”. This information is encoded in the coverage vector of each hypothesis. Figure 2.13⁴ abstracts the decoding process and Figure 2.14 shows a part of the stack space.

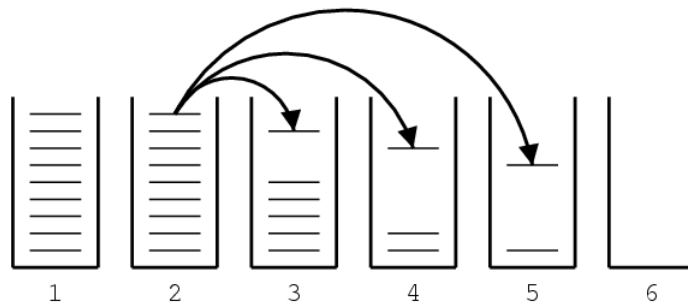


Figure 2.13: Beam Search Decoding

In order to get the final output, the best scoring translation is searched for, in the last stack. This hypothesis is then traversed to the starting hypothesis through back pointers to get the best scoring English sentence. An empty hyp is the starting hypothesis that has covered no German words as yet.

2.2.6 Decoding Complexity

At the beginning of this section we mentioned that decoding for SMT has been shown to be an NP-complete problem. In this section we will briefly go through how this is dealt with in practice.

Hypothesis Recombination

Hypothesis recombination takes benefit from the observation that hypotheses producing identical translations can be merged. This is particularly useful in phrase-based systems where different phrasal segmentations lead to the same output translation. See Figure 2.14. The hypothesis “wäre ebenso unverantwortlich – it would be just as irresponsible” also has an alternative representation where “wäre – it would be” and “ebenso unverantwortlich – just as

⁴This figure is borrowed from Koehn (2004a)

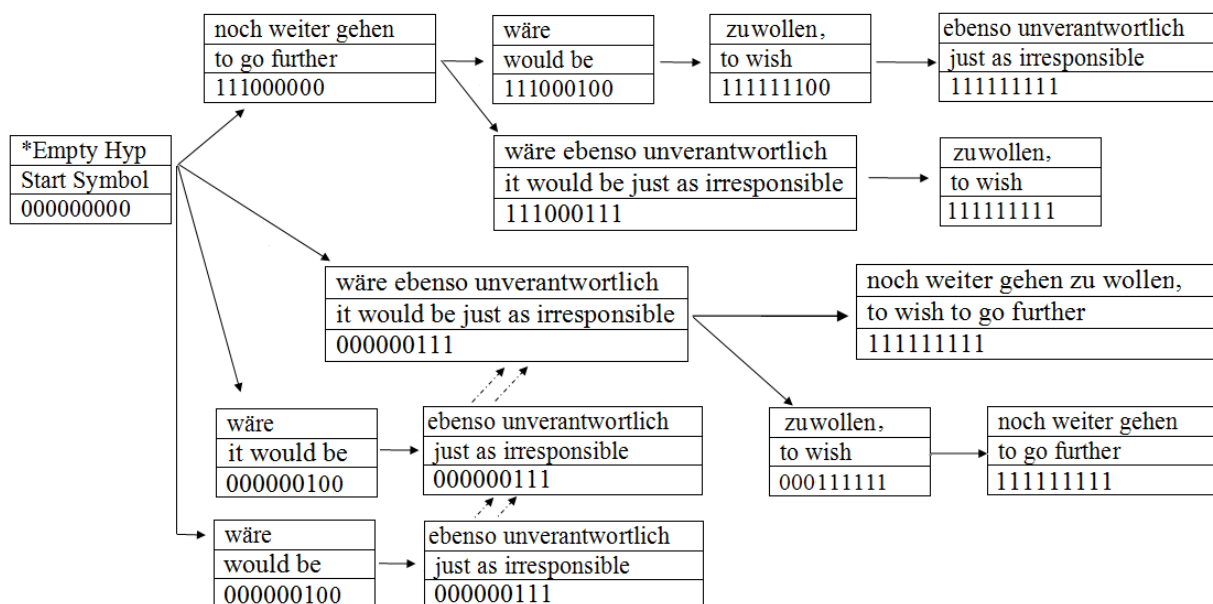


Figure 2.14: Hypothesis Extension and Recombination

irresponsible” are translated as two separate phrases in two hypotheses. The phrase-based system gives separate model scores to each of these representations. We can therefore safely drop the worse scoring hypothesis, or merge it into the better scoring hypothesis, if we are interested in N-best list of translations.

Recombination can also take place even if the output strings are different but identical in the language model context, coverage vector, reordering context (position of the last German word covered). See Figure 2.14 again. The worse scoring hypothesis “ebenso unverantwortlich – just as irresponsible” extending from “wäre – would be” could be merged into the hypothesis ebenso unverantwortlich – just as irresponsible” extending from “wäre – it would be”, because both hypotheses have the same coverage vector ($\{000000111\}$), lan-

guage model context (“would be”) and the position of the last German word translated (7). The argument is that the merged hypothesis can never lead to the best output translation according to the model. Any hypothesis leading from the merged hypothesis will have a better model score if continued from the better hypothesis and thus will be placed higher in the final stack.

Pruning

Hypothesis recombination makes the search more efficient. It contributes a lot in removing spurious ambiguities that arise due to phrasal segmentation, however it does not reduce the complexity of the search problem. To do that we resort to a less safe technique called pruning. The idea of pruning is to keep only the most promising hypotheses at any point in search and throw away the rest. Two commonly used pruning techniques are histogram pruning and threshold pruning. Hypotheses in stacks are kept in sorted order. Histogram pruning allows only the k -best hypotheses per stack. The threshold pruning method defines a beam threshold α . All the hypotheses worse than the best scoring hypothesis in a stack by a factor of α are thrown away. Both methods have their pros and cons and are used in practice. Limiting the stack space to retain k hypotheses per stack reduces the complexity of the problem from being exponential to polynomial (Koehn, 2004a, 2010). Pruning, unlike hypothesis recombination, is not a safe method. Because there is a chance that we may throw away some hypothesis, that is currently ranked below the k best hypotheses in the stack, but may produce the best translation in the end if not dropped. This is called a search error.

Reordering Limit

The decoding problem is further simplified by using a hard reordering limit on the reordering. Recall the linear reordering feature $d(\alpha_i - \beta_i - 1)$ that calculates the jump distance in terms of number of words skipped or jumped over when translating a phrase F_i . Apart from using this feature as a soft constraint, phrase-based system also use the reordering distance as a hard constraint.

This is done by disallowing movement beyond a window of 5-8 words (Koehn, 2010; Galley et al., 2009). This improves the decoding speed dramatically, because the number of translation options to be tried is significantly reduced. Using a hard constraint on reordering not only speeds up the decoding process but also improves the translation quality by reducing the number of search errors. The reordering model in the phrase-based system is unable to discount bad long-distance reorderings enough, causing good hypotheses to be dropped which results in search errors. Using a hard distortion limit might not be a bad idea for the translation of language pairs like French-English, Arabic-English and Chinese-English, etc. But this is undesirable for the translation of German and Japanese to English. We will come back to this problem later when we talk about the drawbacks of phrase-based systems.

2.2.7 Future Cost

When the pruning of hypotheses is done on the basis of model scores only, there is a problem. Some parts of a sentence are more difficult to translate than others. For example the translation of content words, such as “ebenso unverantwortlich – just as irresponsible”, is more expensive than translating frequent words, such as “zu wollen – to wish”. Because stacks are arranged based on the number of German words, that the hypotheses in it have translated so far, this leads to an unfair hypothesis comparison, resulting in search errors. The hypotheses that have skipped the difficult part of the sentence until now, will eventually need to translate it during later stages of decoding. In order to avoid this problem, an estimate of future cost is used along with the model score of the hypothesis, when comparing these for pruning. Future cost is an estimate of the expected cost required to translate the remaining part of the German sentence. A perfect future cost estimate will eliminate search errors altogether. However, computing the precise future cost is as complex as the decoding problem. Therefore, we resort to approximate estimates.

Estimation of Future Cost

In this section we describe one commonly used method for future cost estimation which is used in standard phrase-based SMT and in many other decoders that do left-to-right beam decoding.

The future cost is estimated in two steps. Step 1 estimates the cost of translating each extracted phrase. Step 2 uses a dynamic programming algorithm to estimate the future cost for bigger spans using estimates obtained from Step 1.

Step 1 – Future Cost for Phrases: The first step is to estimate the future cost for each extracted phrase. We will estimate the probability/cost of each feature component independent of context. Because of the phrasal independence assumption the probabilities of features like $p(E_i|F_j)$, $p(F_j|E_i)$ and their lexical probabilities can be estimated exactly. We can also compute the word and phrase penalties exactly. However, we do not have the language model context for the English part of the phrases, therefore we settle to unigram estimate for the first word of phrase, bigram estimate for the second word of a phrase and so on. The language model estimate for the phrase “noch weiter gehen zu – to go further” would be:

$$p_{lm} = p(to) * p(go|to) * p(further|to go)$$

The reordering cost is usually ignored in standard phrase-based decoding. An alternative approach is to assume that every phrase will be translated monotonically so the distance penalty would be zero and the lexicalized reordering model probability would be estimated as:

$$p_r = p(monotonic|noch weiter gehen zu, to go further)$$

Given all the feature values, the total cost of a phrase pair is calculated. A German phrase can have many possible English translations. We pick the

phrase pairs with the lowest cost. This process is repeated for all the extracted phrases.

Step 2 – Future Cost for Larger Spans: Loaded with the best estimate for covering a phrase, the next step is to estimate the cost for covering larger spans. Standard dynamic programming technique can be used here. The cost of a span $cost(i, k)$ is calculated as:

$$cost(i, k) = \min_{i \leq j \leq k} \{cost(i, j) + cost(j + 1, k)\}$$

At each step we break the problem $cost(i, k)$ into two sub-problems $cost(i, j)$ and $cost(j + 1, k)$ until we hit the base case i.e. a phrase spanning i, j . The future cost table is initially loaded with the initial costs extracted in Step 1. See Figure 2.15 for illustration and Koehn (2010) for details.

$$cost(1,8) = \min (\{ cost(1,1) + cost(2,8) \}, \{ cost(1,2) + cost(3,7) \}, \dots \\ \dots, \{ cost(1,7) + cost(8,8) \})$$

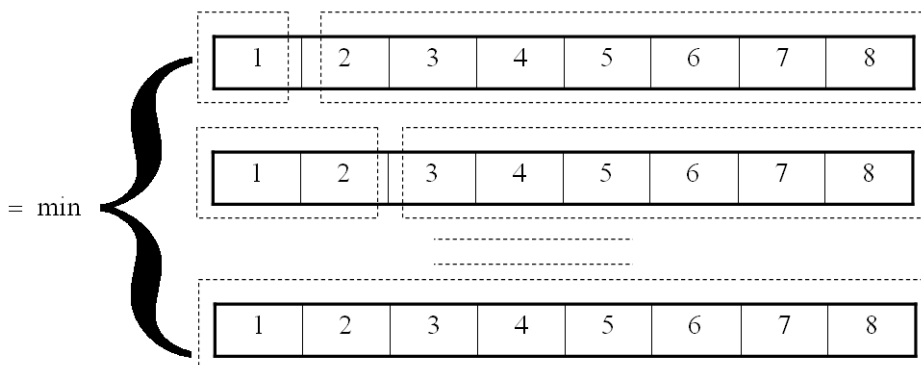


Figure 2.15: Future Cost Estimation for Larger Spans

Using Future Cost during Search: Once the cost for bigger spans is estimated we have a look up function $cost(i, j)$ that provides an estimate for translating German words in the span i to j . This estimate can be added to the actual cost of the hypothesis during decoding to make a fair comparison with other hypotheses that might be covering a different set of German words.

The future cost is computed during decoding by looking at the coverage vector of the hypothesis. Consecutive zeros represent span of uncovered words. All such spans are first detected. Then the cost function $cost(i, j)$ is used to calculate the cost of such spans. These are finally added to the actual hypothesis cost to get a final cost which is then used for the pruning. See Figure 2.16 for illustration. Hyp 1 and Hyp 2 both cover three German words but have different coverage. Hyp 1 has to translate German words 4 to 9 and Hyp 2 has to translate words 1 to 4 and words 8 and 9. The future cost estimate for the uncovered spans are added to the actual hypothesis cost. Pruning is then done based on total cost of the hypothesis.

Hyp 1	Hyp 2
...	...
Coverage: 111000000	Coverage: 000011100
Hyp Cost: X_1	Hyp Cost: X_2
FC = cost (4,9)	FC = cost (1,4) + cost (8,9)
Total Cost: $X_1 + FC$	Total Cost: $X_2 + FC$
....	...

Figure 2.16: Using Future Cost during Decoding

2.3 Drawbacks of Phrase-based SMT

Phrase-based SMT provides a powerful translation mechanism which learns local reorderings, translation of short idioms, and the insertion and deletion of words sensitive to local context. However, phrase-based machine translation also has some drawbacks.

1. Dependencies across phrases are not directly represented in the translation model.
2. Discontinuous phrases cannot be represented and used.
3. The reordering model is not designed to handle long range reorderings.
4. Search and modeling problems require the use of a hard reordering limit.

5. The presence of many different equivalent segmentations increases the search space.
6. Source word deletion and target word insertion outside phrases is not allowed during decoding.

In this section we will discuss these drawbacks to motivate the research conducted in this thesis. Some of the mentioned problems are inherent to using phrases as translation units, while others are due to lack of modeling of a specific phenomenon for which there is an ongoing research which tries to solve these problems.

2.3.1 Handling of Non-local Dependencies

Phrase-based SMT models dependencies between words and their translations inside of a phrase well. However, dependencies across phrase boundaries are largely ignored due to the strong phrasal independence assumption. Recall that the translation model is defined as:

$$p(f|e) = \prod_{i=0}^n p(F_i|E_j)d(\alpha_i - \beta_{i-1} - 1)$$

Consider the bilingual sentence pair shown in Figure 2.17. Reordering of the German word “abstimmen” is internal to the phrase-pair “gegen ihre kampagne abstimmen – vote against your campaign” and therefore represented by the translation model. Given a hypothetical phrase-table shown in Table 2.5, the translation model correctly translates the German sentence “die menschen würden gegen ihre kampagne abstimmen – people would vote against your campaign”. However, it fails to translate “die menschen würden gegen meine außenpolitik abstimmen” (see Table 2.5 for a gloss), which is translated as “people would against my foreign policy vote” unless the language model provides strong enough evidence for a different ordering.

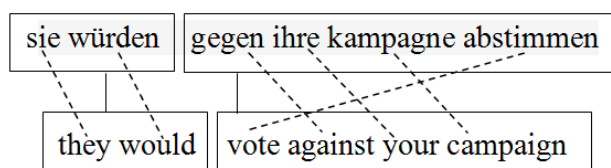


Figure 2.17: Handling of Local Dependencies – Dotted lines = Word Alignments

German	English
gegen ihre kampagne abstimmen	vote against your campaign
sie würden	they would
gegen	against
ihre	your
kampagne	campaign
abstimmen	vote
sie	they
würden	would
die menschen	people
außenpolitik	foreign policy
meine	my

Table 2.5: Hypothetical Phrase Table

Why? In the prior example, the translation model is able to capitalize on the phrase pair “gegen ihre kampagne abstimmen – vote against your campaign”, observed during training. The same can not be done for the latter example where data sparsity forces the translation model to fall back to word translation and deal with the reordering of the verb “abstimmen – vote” through other means. This phenomenon is fairly common in practice. Although the phrase-based system has the capacity to learn phrases of unrestricted length, previous research has shown that phrases used at test time are shorter than 3 words on average (Zhang and Vogel, 2005; Callison-Burch et al., 2005).

This problem has been recently addressed by Galley and Manning (2010)

by learning discontinuous source and target phrases. See section 2.5 for the details.

2.3.2 Modeling of Gappy Units

Another weakness of the traditional phrase-based system is that it can only capitalize on continuous phrases. If the discontinuity occurs inside of a phrase it can be learned and used during decoding. However, such dependencies can not be handled across phrases.

German	English
hat er ein buch gelesen	he read a book
hat	has
er	he
buch	book
gelesen	read
zeitung	newspaper
dann	then
märchenbuch	story book

Table 2.6: Hypothetical Phrase Table

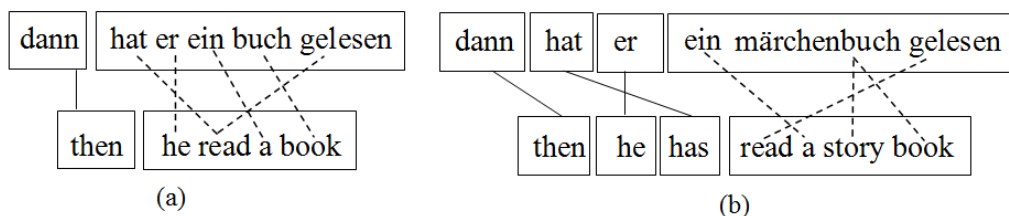


Figure 2.18: Handling of Gaps – Dotted lines = Word Alignments – (a) Learned Phrase (b) Unseen Context

Given the phrase inventory in Table 2.6, phrasal SMT is able to generate the example in Figure 2.18(a). The information “hat...gelesen – read” is internal to the phrase pair “hat er ein buch gelesen – he read a book”, and is therefore handled conveniently. On the other hand, the phrase table does not have the

entry “hat er eine märchenbuch gelesen – he read a story book”. Hence, there is no option but to translate “hat...gelesen” separately, translating “hat” to “has” (as in Figure 2.18(b)) which is a common translation for “hat” but wrong in the given context.

Other examples in German, where modeling of discontinuous translation units can be helpful are particle verbs. Such verbs can have separable prefixes that can move to the end of clause or sentence. An example of this is “(zu)machen – close” as in “machen sie bitte die tür zu – please close the door”. Instead of learning a discontinuous translation unit “machen...zu – close”, a phrase-based system aligns “machen–close” and learns deletion of “zu”. This is problematic because the connecting prefix can change the meaning of the verb. For example “(auf)machen” means “open”. On seeing a sentence like “machen sie bitte die tür auf”, the phrase-based system has no clue whether “machen” means “open” or “close” in the given context. Discontinuous units are also common in other languages such as “turn...off” in English as in “turn all electric devices off”.

An initial effort to overcome this problem for phrasal SMT was proposed in Simard et al. (2005). They introduce non-contiguous phrases having gaps on both German and English sides. For the example under discussion, a phrase-pair such as “hat $\diamond\diamond\diamond$ gelesen – read” is learned. Each \diamond sign is a placeholder for one skipped word. A problem with this approach is the fixed gap width. Each gap must span exactly one word. This creates an issue of data sparsity. The example will only help in cases where “hat” and “gelesen” are separated by exactly three words. The mechanism can successfully translate “dann hat er ein märchenbuch gelesen”, however fails when translating “er hat ein märchenbuch gelesen – he read a story book”, in which case “hat” and “gelesen” are separated by two words.

Recently Galley and Manning (2010) proposed using discontinuous phrases on both the German and English side. Their representation does not limit gaps to be of fixed size, hence making them more useful to generalize during test. See section 2.5 for details. Context-free hierarchical models (Chiang, 2007; Melamed, 2004) have rules like “hat er X gelesen – he read X” to handle such

cases. The discontinuities on German side are handled naturally because of source linearization in N-gram-based SMT (Crego et al., 2005c). Crego and Yvon (2009), in their N-gram system, use split rules to deal with target-side gaps and show a slight improvement on a Chinese-English translation task. Chapter 3 describes N-gram-based machine translation.

2.3.3 Weak Reordering Model

The lexicalized reordering models discussed in Section 2.2.3 give state-of-the-art performance for phrase-based machine translation. However, they are primarily designed to deal with short distance movement of phrases such as swapping two adjacent phrases. Long distance reordering often motivates parsing-based machine translation systems (Melamed, 2004; Chiang, 2005). Galley and Manning (2008) try to remedy this problem by defining the orientation based on larger blocks rather than previous phrases. However, parameters learned by this model remain weak on handling long distance reordering, and still heavily rely on the language model to select the right word order. Let us revisit the training example shown in Figure 2.17. The parameters learned from this example are shown in Figure 2.19.

sie würden gegen ihre kampagne abstimmen

they would vote against your campaign

The relevant parameters learned from the example are the orientations of the words “würden” and “abstimmen”, shown in Table 2.7, according to which “würden – would” is translated monotonically with respect to its previous phrase and its orientation is discontinuous with respect to the next phrase. The orientation of “abstimmen – vote” is discontinuous with respect to the previous and the next phrase.

Now consider applying these parameters to a sentence such as:

All the hypotheses shown in Table 2.8 are equivalent in terms of reordering parameters of the words “würden” and “abstimmen”. Which of these hypotheses gets selected as the best translation depends upon the feature values of the

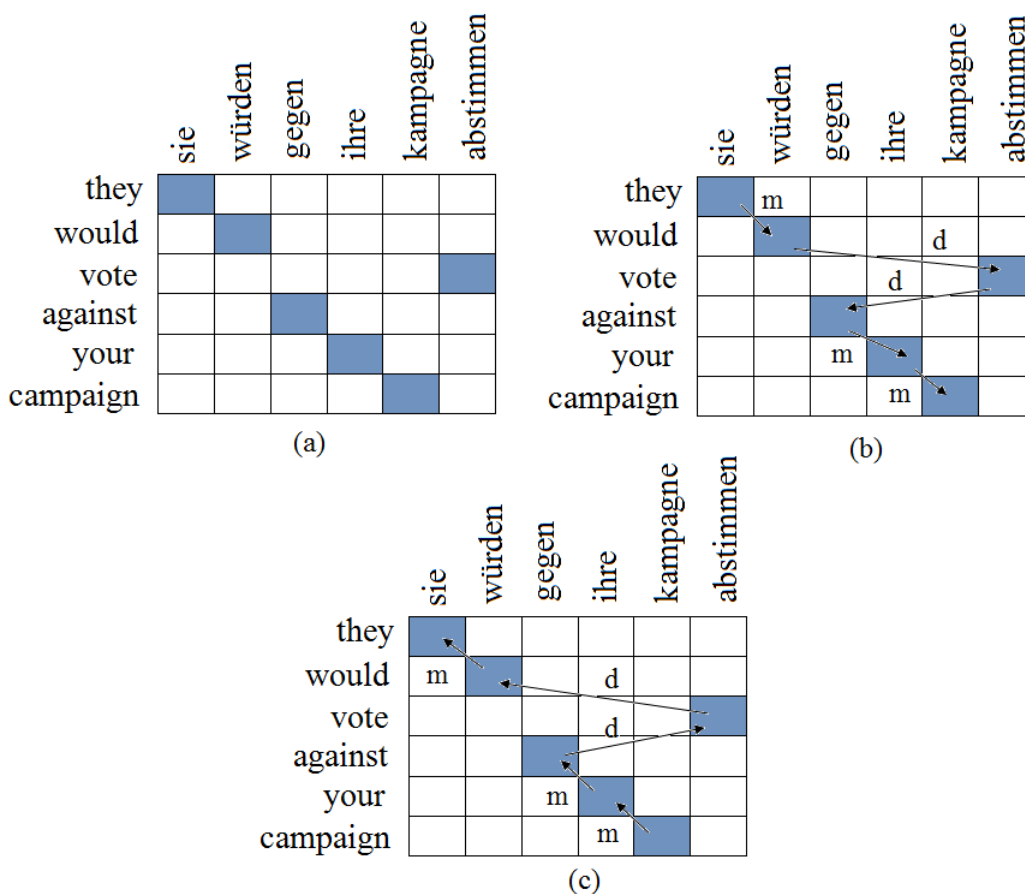


Figure 2.19: Lexicalized Reordering Model – (a) Word Alignments (b) Backward Orientation (c) Forward Orientation

other parameters such as orientation of other words in the sentence, language model score and reordering distance etc. The linear distortion parameter will penalize movement of “abstimmen”. The monolingual language model is no longer able to compensate for the dis preference of the linear distortion model for non-local reordering, in presence of other hypotheses such as “They would legalize in Canada ...” which might also get good bi- and trigram probabilities.

The principle drawback of this model is its weak connection between the

Phrase Pair $\langle F_i, E_j \rangle$	Backward Orientation	Forward Orientation
sie würden – they would	monotonic	discontinuous
abstimmen – against	discontinuous	discontinuous

Table 2.7: Learned Parameters for Phrase-based SMT

sie würden für die Legalisierung der Abtreibung in Kanada abstimmen

reordering of the words “würden – would” and “abstimmen – vote”. The model only learns how these phrases were translated with respect to their previous and next phrase, and makes independence assumptions over previously translated phrases. It does not take into account how previous words were translated and reordered. The model does not learn that reordering of “abstimmen” is highly probable after translation of “würden – would” in order to move the second part of the German complex verb to its correct position. The further to the right the word “abstimmen” is in the sentence the more difficult it is for the lexicalized reordering model to move it to the correct position.

74% würden gegen die Studiengebühren, 79% gegen die Praxisgebühr, und 84% gegen das Krankenhaus-Taggeld stimmen

74% would vote against the tuition fee, 79% against the clinical practice, and 84% against the hospital daily allowance

Figure 2.20: Long Distance Reordering

2.3.4 Hard Distortion Limit during Search

Phrase-based systems apply a hard distortion limit during search, restricting the reordering to a window of 5-8 words. This decreases the decoding time from being polynomial to linear and reduces the search complexity by throwing away all possible word permutations beyond a fixed range. Previous research (Koehn et al., 2005) has shown that a distortion limit beyond 8 words drops the

Hypotheses
They would vote to legalize abortion in Canada
They would abortion in Canada legalization vote
They would legalize the abortion vote in Canada
They would legalize vote abortion in Canada
They would legalize vote in Canada abortion
They would legalize in Canada vote abortion
They would in Canada vote legalize abortion
...

Table 2.8: Hypothesized Translations with the Orientations of *würden* and *abstimmen* as shown in Table 2.7

translation accuracy because of the search errors. The lexicalized reordering model is not good enough to filter out bad large-scale reorderings (Koehn, 2010).

The use of a hard reordering limit does not impact the translation between language pairs such as French-English, Chinese-English, Arabic-English where short distance movements suffice. However, the use of a hard limit is undesirable for German-English and Japanese-English translation, as these pairs have significantly different syntactic structures. German is a verb-final language as compared to English where the verb follows immediately after the subject. In longer German sentences the verb can be easily separated from the subject by more than 10-15 words. A hard reordering limit causes phrasal SMT to reject such hypotheses even before trying them out in the search process. See Figure 2.20 for example. To move the German clause-final verb “*stimmen* – vote” to its correct position behind the auxiliary “*would*”, it needs to jump over 15 German words. Such long-range reorderings motivate syntax-based approaches to machine translation.

Use of hard constraints is ultimately undesirable in machine translation and any other machine learning problem, where the goal is to learn all the parameters from the data. A strong reordering model should be able to discard bad

hypotheses and keep the good ones through soft constraints learned from the data. Green et al. (2010) recently addressed this problem through better estimation of the future cost of the linear distortion model. This enables them to achieve the same or better translation accuracy (on different test sets) than the baseline⁵ phrase-based system for a higher distortion limit of 15 words, for an Arabic-to-English translation task. They also propose a discriminative reordering model to predict the word movement during translation. The reordering distance $\alpha_i - \beta_{i-1} - 1$, as used in the linear distortion model, is discretized into nine classes, the parameters of which are learned from the bilingual training data.

2.3.5 Spurious Phrasal Segmentation

One of the inherent properties of phrase-based machine translation is its ambiguous segmentation of the source sentence into phrases. Given a bilingual sentence pair and its word alignment, a phrase-based system can learn a large number of source segmentations. See Figure 2.21 for a sample phrase extraction example. All phrases extracted for this example are exhaustively enumerated in Table 2.2.

While it is not a problem to extract and store any number of phrases during training, because of the now-days large computing power, spurious ambiguity created by arbitrary phrasal segmentation is undesirable from the perspective of modeling.

During decoding the spurious phrasal segmentation causes different compositions of the same phrasal unit to exist and compete with each other. The same translation is hypothesized with different segmentations. This is illustrated in Figure 2.22 where multiple segmentations of the phrase “inflation sollte unter zwei prozent” compete with each other. This problem is substantially reduced,

⁵However, their baseline system does not include lexicalized reordering model. It is unclear, whether the reported results are better or at least as good as the baseline system with lexicalized reordering and lower distortion limit of 5, which has been shown to be optimal for the Arabic-to-English translation task (Galley et al., 2009)

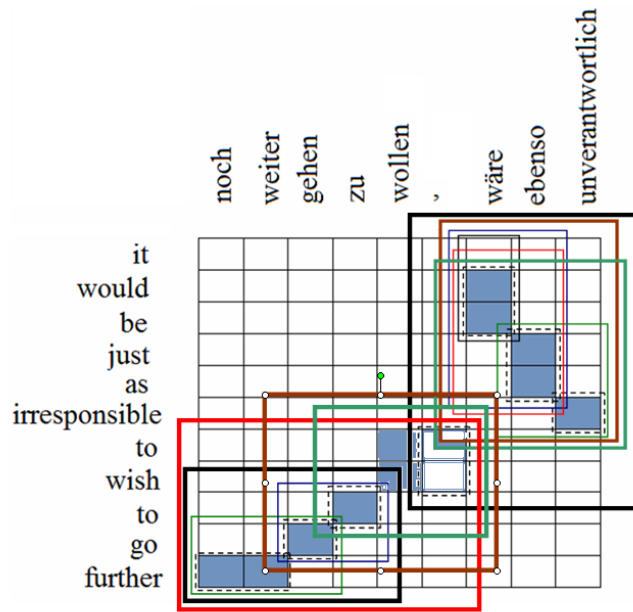


Figure 2.21: Spurious Phrasal Segmentation – Training

but not completely eliminated, with the help of hypothesis recombination, a technique to merge identical hypotheses (see Section 2.2.6). However, recombination can take place only after the hypothesis has been created and the values for all its feature components have been computed, thus making the decoding inefficient.

Phrasal segmentation is less of a problem for the phrase-based system in practice because of the hard reordering limit, applied during decoding and data sparsity, that forces the decoder to use only smaller phrases during test. However, the removal of the hard reordering limit which is ultimately desired, will cause a massive increase in the decoding time because of the multiple segmentations of the same phrasal unit being tried over and over again with different hypothesis extensions.

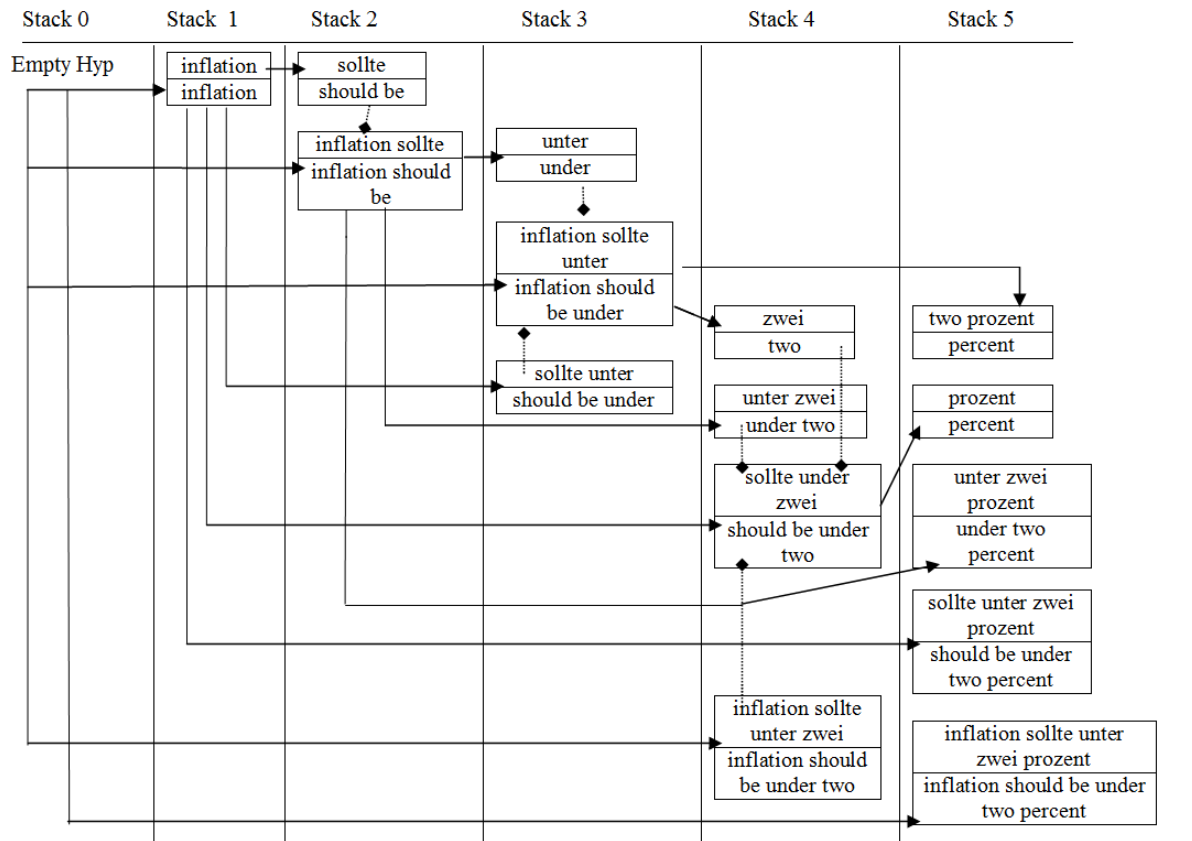


Figure 2.22: Spurious Phrasal Segmentation – Recombined Hypotheses Represented with Dotted Lines

2.3.6 Deletions and Insertions out of Phrases

Handling of spurious words is one of the desirable feature in statistical machine translation. Models for deleting German words (like the flavoring particle “ja”) and insertion of English words (like auxiliary verb “does”) can help improve translation.

Phrase-based systems handle deletion and insertion of words inside of phrases but do not allow these operations outside phrases. This means that words can

only be deleted during test if they appear in the same context as during training. Consider the example of short sentence “kommen Sie mit – come with me”. German pronoun “Sie – you” always gets deleted when followed by a verb. The phrase-based system learns the three phrases from this sentence shown in Figure 2.23. But none of these prove helpful when translating the test sentence “lesen Sie bitte mit”, which is translated as “please read you with me” instead of “please read with me”. The German word “Sie” translates to “you”, an English word that it usually translates to. Deletion of “Sie” could not be hypothesized because the phrase-based system does not learn and use phrases like “Sie – null”.

It is arguable, whether out of context arbitrary deletion of source (German) words, can be helpful. Li et al. (2008) showed an improvement of more than 1.5 BLEU points with a simple extension of the phrase-based system on a Chinese-English task. They introduce an imaginary token ϵ (null word) on the English side and learn a probability distribution $p(\epsilon|F_i)$, by counting phrase pairs such as $\langle F_i, \epsilon \rangle$ and calculating the MLE by dividing by counts of F_i . Further improvements were achieved by using more sophisticated models that involve using POS tags.

Spurious insertion of English words during decoding is a much more difficult problem than German word deletion. Notice that the problem of word deletion is only to justify whether any German word appearing in the test sentence can be dropped without translation. However, the English sentence is hidden at test time, and is actually what we construct during decoding. It is a non-trivial problem to identify which words to hypothesize as candidate for insertion during decoding. Hypothesizing all unaligned English words observed during training, increases the decoding complexity.

2.4 Hierarchical Phrase-Based Translation

Chiang (2007) extended the phrase-based translation model to include hierarchical phrases – phrases that contain sub-phrases to improve the inter-phrase

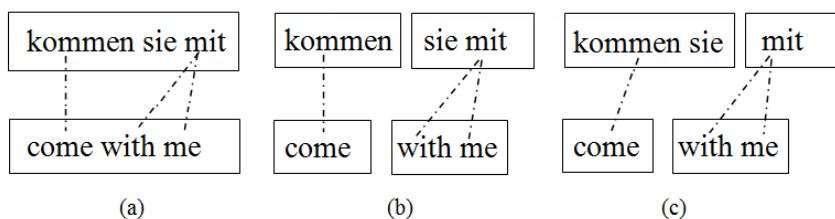


Figure 2.23: German Word Deletion – Different Learned Phrases

reordering. The main goal was to enable the idea of hierarchical structure (as previously advocated by Syntax-based MT, but not proven to be as effective as the phrase-based translation at that time), retaining key insights from phrase-based MT i.e. learning large chunks of translation and the capacity to memorize local dependencies.

The model is based on synchronous context-free grammar (SCFG) where the translations are represented as rewrite rules such as:

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle$$

where X is a non-terminal, γ and α are the source and target phrases, containing terminals (strings) and non-terminals, and \sim is one-to-one correspondence between the nonterminal occurrences in γ and α . The SCFG rules are extracted through a following sequence of steps:

- The word alignments are obtained by running GIZA++ in both directions
- Alignments are symmetrized
- Continuous phrases are extracted just as in the standard phrase-based system
- For each continuous phrase, remove any sub-phrases within that phrase and replace it with non-terminals

For example the phrase pair “gegen ihre kampagne abstimmen – vote against your campaign” in Figure 2.17, the following rules could be learned:

$$X \rightarrow \langle \text{gegen ihre kampagne } X_1, X_1 \text{ against your campaign} \rangle$$

$$X \rightarrow \langle \text{gegen } X_1 \text{ kampagne } X_2, X_1 \text{ against } X_2 \text{ campaign} \rangle$$

$$X \rightarrow \langle \text{gegen } X_1 \text{ kampagne } X_2, X_1 \text{ against } X_2 \text{ campaign} \rangle$$

Extracting rules in this manner can result in a very large rule vocabulary creating spurious ambiguity which is more severe in this case than traditional phrase-based system. The decoder can produce the same output sentence through different set of derivations having exactly same feature vector values. This also results in slower decoding and more search errors. To avoid this, the rule inventory is filtered using several constraints such as i) two non-terminals can not occur adjacently on the source-side, ii) rules with more than two non-terminals are removed iii) limit the rule size i.e. total number of terminals and non-terminals in a rule to be 6.

The hierarchical phrase-based translation, however, departs from the left-to-right stack-based decoding that produces English sentence linearly. Decoding is done through CKY (Cocke-Kasami-Younger) parser using cube-pruning (See Chiang (2007) for details).

By using rules with non-terminals the hierarchical phrase-based translation system (Hiero), can handle both local and non-local dependencies through the translation model and can also handle discontinuous units on source and target-sides. However, SCFG-based systems such as Hiero have been recently criticized for their inability to model certain types of alignments (Søgaard and Kuhn, 2009; Søgaard and Wu, 2009). Hiero, for example can not independently generate translation units, a, b, c, and d with the types of alignments shown in Figure 2.24 (Galley and Manning, 2010). Phrase-based system can handle (i) inside-out but unable to deal with (ii) cross-serial DTU and (iii) “Bonbon”.

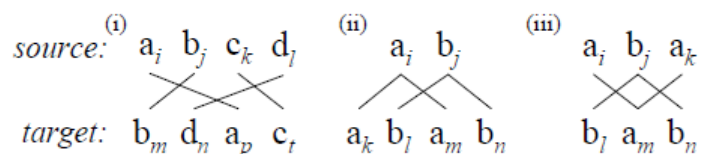


Figure 2.24: Alignments Not Handled By Hiero

2.5 Discontinuous Phrase-based SMT

In this section we discuss a recent advance in phrase-based machine translation that addresses the first two drawbacks of the PBSMT (i) handling of non-local dependencies and (ii) modeling of gappy units, discussed in the previous section, that often motivate parsing-based approach to machine translation. The principle weakness that triggers these problems is that traditional phrase-based machine translation only extracts continuous phrases during training. Because of this limitation the translation model can not learn long distance dependencies with context. Recall the example shown in Figure 2.25. The translation model can learn a large continuous phrase “gegen ihre kampagne abstimmen – vote against your campaign” which is less useful during test or a small, one word phrase “abstimmen – vote” which does not capture any translation context.

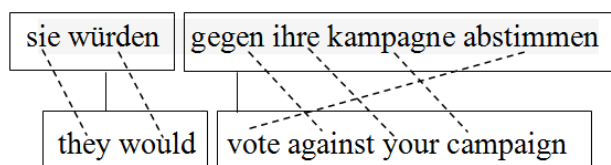


Figure 2.25: Handling of Local Dependencies – Dotted lines = Word Alignments

Galley and Manning (2010) has recently addressed this problem by removing the limitation of extracting only continuous phrases. All continuous and discontinuous phrases $\langle F_i, E_j \rangle$ are extracted that are consistent with the

given alignment. A phrase pair is said to be consistent with an alignment A if all German words in F_i have alignment points only with the words in English phrase E_j and vice versa. Any German or English word in $\langle F_i, E_j \rangle$ can be unaligned but there has to be at least one alignment point in the phrase. More formally as described in the paper:

$$\forall (x, y) \in A : x \in F_i \longleftrightarrow y \in E_j$$

All discontinuous phrases extracted for this example are shown in Table 2.9. The interesting ones are boldfaced.

English	German
kampagne abstimmen	vote X campaign
würden X abstimmen	would vote
sie würden gegen ihre kampagne	they would X against your campaign
würden gegen ihre kampagne	would X against your campaign
ihre kampagne abstimmen	vote X your campaign
würden gegen	would X against
sie würden gegen	they would X against
sie würden gegen X abstimmen	they would vote against
gegen X abstimmen	vote against
würden gegen X abstimmen	would vote against
sie würden X abstimmen	they would vote
sie würden gegen ihre	they would X against your
gegen ihre X abstimmen	vote against your
würden gegen ihre X abstimmen	would vote against your
sie würden gegen ihre X abstimmen	they would vote against your
würden gegen ihre	would X against your

Table 2.9: Discontinuous Phrases in Figure 2.25 – X = Place holder for 1 or more words

The decoding mechanism is based on the left-to-right decoding as used in the standard phrase-based system. However, in order to produce phrases with target-side discontinuities, the hypothesis extension is divided into two steps, grow and consolidate. The first part of such a phrase is extended through a grow operation, whereas all subsequent $n - 1$ parts of it are produced with $n - 1$

consolidate operations. Each consolidate hypothesis extension operation will simply generate the remaining isolated target-side phrases. For example when translating the phrase-pair “würden gegen ihre – would X against your”, the grow operation would produce “would” and isolate the second part “against your” to be generated later. The decoder will then cover the phrase-pair “abstimmen – vote” and then use the consolidate operation to produce the isolated phrase.

The new model learns discontinuous phrases such as “würden X abstimmen – would vote” capturing the non-local dependency which is helpful when translating the test sentence “die menschen würden gegen meine außenpolitik abstimmen”. Similarly learning phrases such as “gegen X abstimmen – vote against” forms a template that could generalize to “gegen meine außenpolitik abstimmen – vote against my foreign policies”. Useful templatic phrases can also be learned on the English side. For example “würden gegen ihre kampagne – would X against your campaign” would be useful when translating the test sentence “sie würden gegen ihre kampagne protestieren – they would protest against your campaign”. However, notice that the phrasal independence assumption is still a problem for the phrase-based system. The phrase-based system can either use the phrase “würden X abstimmen – would vote” or the phrase “gegen X abstimmen – vote against” during decoding.

Now we will come back to the second problem of modeling discontinuous word alignments such as “hat ... gelesen - read” as shown in Figure 2.18. A traditional phrase-based system does not learn this unit independently of the intervening context “er ein buch”. Instead it learns a phrasal unit “hat er ein buch gelesen – he has read a book” which is less useful during test. This problem is naturally solved in the new model that learns discontinuous phrases. From the given training sentence, the model is able to learn discontinuous phrases such as “hat X gelesen – read” which can be generalized to the test sentence “dann hat er eine märchenbuch gelesen”. Another useful phrase learned by the new model is “hat X ein X gelesen – read a”. The first gap can be filled with any pronoun and the second gap can be filled with any object noun.

Also notice that the discontinuous phrase-based system can handle the cross-serial DTU and “Bonbon” alignments shown in Figure 2.24 unlike hierarchical phrase-based system. Galley and Manning (2010) in their results on Chinese-to-English task show statistically significant improvements over Joshua (Li et al., 2009), a hierarchical phrase-based and Moses, the standard phrase-based system. Their results also show the Moses slightly outperform Joshua, although Joshua uses both continuous and discontinuous phrases.

2.6 Chapter Summary

In this chapter we gave a comprehensive overview of the state-of-the-art phrase-based system and its different components. We discussed the noisy channel metaphor which rests as a foundation for statistical machine translation. We gave a brief account on IBM models, starting with Model 1 which is based on simple word replacement. Model 2 adds an absolute reordering model. Model 3 adds a fertility model. Model 4 returns to the challenge of reordering and proposes relative reordering. Model 5 addresses the problem of deficiency. We discussed the drawbacks of word-based models to motivate the phrase-based model, showing benefits of using phrases over words as a unit of translation.

We mainly presented standard phrase-based SMT which is based on the noisy-channel framework, in terms of translation model and phrase extraction. We also briefly talked about different joint models for phrase-based SMT (i) that are based on concepts that directly extract phrases from sentence-aligned data through EM training or (ii) that are built on symmetrized word alignments.

We discussed the drawbacks of distance-based reordering, followed by an account of lexicalized reordering that considers word forms rather than indexes. We discussed several variants of the lexicalized reordering as proposed in the literature. We discussed that machine translation can be improved a lot by integrating different sources of knowledge inside and outside of the SMT mod-

els. To make this possible we have to shift from the generative paradigm to log-linear modeling.

We discussed beam search as used in phrase-based decoders. We talked about different aspects of search such as hypothesis extension, future cost estimation and decoding complexity. By applying a pruning and hard reordering limit, search complexity can be reduced from NP-complete to being linear.

At the end of the chapter we talked about some drawbacks of phrase-based machine translation. Traditional phrase-based systems do not represent non-local dependencies through the translation model. Modeling of gappy units and spurious ambiguities is ignored. The reordering model is weak and can not handle long distance reorderings effectively without relying on the language model. Hard reordering limit is applied during decoding to reduce search errors and make the decoding efficient. Removing the hard limit causes a performance drop. Traditional phrase-based SMT extracts non-minimal translation units during training which cause spurious ambiguities during training and decoding, allowing multiple representations of the same German and English strings, with different segmentations, to occur in the search space.

Lastly we discussed a major recent advance in phrase-based machine translation that removes the restriction of using continuous phrases and introduce discontinuous German and English phrases. This enables the new phrase-based model to learn non-local dependencies with context.

3 N-gram-based SMT

N-gram-based SMT exists as an alternative to the more commonly used phrase-based machine translation approach. While the two models have some common properties, they are substantially different in terms of translation units, reordering and search. In this chapter we will discuss the details of N-gram-based SMT, contrasting it with phrase-based machine translation. The model is discussed in terms of translation modeling, reordering framework and search strategy as used in the N-gram model. We discuss the useful properties of the N-gram model that overcome some of the drawbacks of phrase-based machine translation detailed in the previous chapter. We also mention the drawbacks of the N-gram model that do not exist in the phrase-based approach.

3.1 Translation Model

N-gram-based SMT (Banchs et al., 2005; Mariño et al., 2006) is an instance of a joint model that generates German and English strings together in bilingual translation units called tuples. Tuples are essentially phrases but are minimal translation units and can not be decomposed any further. The translation model is an n-gram model which defines the probability of a sequence of tuples as follows:

$$p(e, f, a) = \prod_{j=1}^J p(t_j | t_{j-m+1} \dots t_{j-1})$$

where each tuple $t_j = \langle F_j, E_j \rangle$ couples German and English strings, m indicates the amount of context used, a defines the alignment function between the bilingual sentence pair $\langle f, e \rangle$. Translation model is implemented as an

N-gram model of tuples using SRILM-Toolkit (Stolcke, 2002) with Kneser-Ney smoothing. A trigram model ($m = 2$) is found to be optimal in Mariño et al. (2006) for Spanish-to-English and English-to-Spanish tasks.

3.1.1 Tuple Extraction

Tuples are extracted from the alignments obtained by running Giza++ in both directions (from source-to-target and target-to-source) and then symmetrizing¹ the alignments. However, the procedure to extract tuples is not the same as applied in the standard phrase-based systems. Tuples are extracted with the following constraints:

1. A bilingual sentence is segmented in such a way that monotonic chunks are obtained i.e. in a tuple t_j , F_j aligns with E_j .
2. No words inside a tuple are aligned to words outside the tuple.
3. A tuple is a minimal unit satisfying the above two conditions.

A Tuple is essentially a phrase-pair with the difference that it is a minimal phrasal unit that can not be decomposed any further without violating the criteria of monotonicity. Consider the bilingual sentence pair:

meistens ist die sache mit einer entschuldigung abgetan

mostly null the matter ends with an apology

Figure 3.1 shows the generation of German and English strings in 5 tuples. Note that tuples are monotonically generated such that F_j (German words in tuple t_j) aligns with E_j (English words in tuple t_j). All words F_j are aligned only to the words inside E_j and vice versa. Lastly no tuple could be split any

¹Mariño et al. (2006) performed a comparison between different symmetrization techniques namely union, source-to-target and refined and found the “union” heuristic to give best results for both Spanish-to-English and English-to-Spanish translation.

further without violating the criteria of monotonicity. Notice that tuple t_5 , can not be decomposed because of the crossing alignment “abgetan – ends”. If we form a tuple $t_x = \langle abgetan, ends \rangle$ then German and English words are no longer generated in the same order. Because of the tuple extraction conditions, only one segmentation is possible for each bilingual sentence pair unlike in traditional PBSMT which extracts many phrasal units.

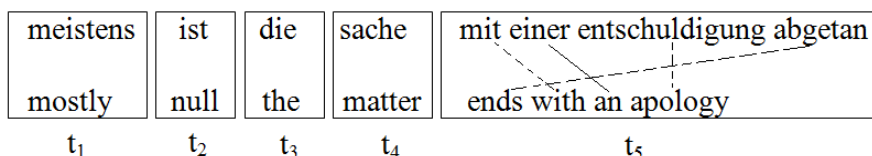


Figure 3.1: Tuple Generation

3.1.2 Crossing Alignments – Embedded Words

One of the main problems with the discussed formulation is tuples like t_5 in Figure 3.1 which result in a loss of smaller translation units that are embedded inside the tuple. Embedded pairs in tuple t_5 are “mit – with”, “einer – an”, “entschuldigung – apology” and “abgetan – ends”. If any of these German words appear in a test sentence, N-gram-based SMT would not be able to translate them unless they appear in the exact sequence as observed during training i.e. “mit einer entschuldigung abgetan” or they have been extracted as smaller tuples in other training sentences. This loss of information is clearly a drawback against phrasal SMT which extracts both smaller and larger translation units for translation. The problem is more aggravating for the translation of language pairs like German-English that involve long range reorderings, resulting in very long tuples that are less useful during test. Even in a language pair like French-English with short reorderings this can cause problem. Consider smaller translation tuples like “beauté noire – black beauty” that flip noun and adjective. If “noire – black” always occurred as a modifier to a noun, the N-gram model can not produce a translation for an unseen construction

“l’encre noire – black ink”. Another daunting problem is that the collected probability estimates are incorrect. Assume that out of 10 sentences where “noire – black” appears, in 5 it appears with an inverted construction like “X noire – black X”. The N-gram model is unable to collect the counts (noire , black) from all such sentences, resulting in incorrect estimates.

In order to handle this problem, a bilingual dictionary of embedded words is appended to the tuple corpus as a unigram model (Mariño et al., 2006) . The embedded tuples are given the same probability as assigned to the unknown tuples by the SRI-Toolkit. Using embedded words to enhance the bilingual tuple corpus is shown to be useful to alleviate the data sparsity problem (de Gispert and Mariño, 2004) when the training data is small. This issue is more properly addressed through linearization of the source sentence, which we will discuss later.

3.1.3 Tuple Pruning

Larger tuples such as “mit einer entschuldigung abgetan – ends with an apology” are less useful during decoding and their presence in the tuple corpus may increase the look-up time, hence causing a drop down in decoding speed. Dropping such tuples may not lead to a drop in the translation performance but may lead to an improvement in decoding speed. Tuple pruning was introduced with this perspective. A threshold value N is chosen and all tuples whose frequency fall below that number are replaced by an unknown tuple token $\langle unk \rangle$. For example the sentence pair shown in Figure 3.1 is represented as:

$\langle meistens-mostly \rangle \langle ist-null \rangle \langle die-the \rangle \langle sache-matter \rangle \langle unk \rangle$

As can be observed that unknown word token $\langle unk \rangle$ is learned with context. This can sometimes be useful in translation of out-of-vocabulary (OOV) words. Consider the following sentence pair:

3 N-gram-based SMT

Musharraf sagte , dass

Musharraf has said that

Suppose that the tuple $\langle \textit{Musharraf} - \textit{Musharraf} \rangle$ occurs below a certain threshold. The model replace it with the $\langle \textit{unk} \rangle$ token. The SRI-Toolkit instead learns a tri-gram such as:

$$\langle \textit{unk} \rangle \langle \textit{sagte} , -\textit{has said} \rangle \langle \textit{dass} - \textit{that} \rangle$$

This can be useful when translating “Gilani sagte, dass” where the model finds a tri-gram. Tuple pruning, however is mainly used as a tool to speed up the decoding process. A value of $N = 20$ and $N = 30$ were found to suffice for translation of Spanish-to-English and English-to-Spanish respectively

3.1.4 Unaligned Words

Unaligned words on the German side are handled by learning tuples such as $\langle \textit{ist} - \textit{null} \rangle$ in Figure 3.1. This enables N-gram-based SMT to learn a source word deletion model that can learn deletions in context by learning a tuple sequence such as:

$$\langle \textit{meistens} - \textit{mostly} \rangle \langle \textit{ist} - \textit{null} \rangle \langle \textit{die} - \textit{the} \rangle$$

A phrase-based system instead learns a phrase-pair such as “meistens ist die – mostly the”. Similar tuples can also be learned for the unaligned words on the English side. However, target-side spurious words can be problematic during decoding. Spurious words on the German side pose no problem during decoding because German sentence is given during decoding. On the contrary the English sentence is hidden and is actually what we are searching for. Hypothesizing all spurious English words observed during training increases the decoding time significantly (See Section 2.3.6 for discussion).

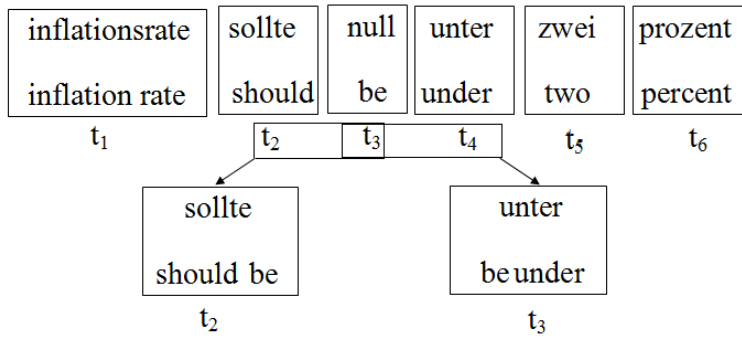


Figure 3.2: Unaligned English Words

A solution to handle this problem is to attach null-aligned English words to the right or left neighbor tuples based on some decision factor. Different strategies have been proposed to make the attachment decision. The simplest approach is to attach the unaligned English word to the next tuple (right tuple). A more principled choice is to attach left or right based on some lexical evidence. Lexical probabilities $p(f|e)$ obtained from IBM Model 1 (Brown et al., 1993) are used to decide whether to attach left or right (Crego et al., 2005b; Mariño et al., 2006). See Figure 3.2 for example. Tuple $t_3 < null - be >$ is required to be merged into tuple t_2 or tuple t_4 . This is done by comparing the product of the probabilities $p(sollte|should)$ and $p(sollte|be)$ with the product of the probabilities $p(unter|be)$ and $p(unter|under)$. The attachment with the better probability estimate is chosen as a merging tuple which in this case is $< sollte - should be >$. An even sophisticated strategy based on POS (Part-of-speech) entropy is proposed in Gispert and Mariño (2006). Forward (p_{POS}^f) and backward (p_{POS}^b) POS distributions are estimated as:

$$\begin{aligned}
 p_{POS}^f &= p(POS_{E_{i+1}} | E_{i-1}, E_i) \\
 &= \frac{\text{count}(E_{i-1}, E_i, POS_{E_{i+1}})}{\text{count}(E_{i-1}, E_i)}
 \end{aligned}$$

3 N-gram-based SMT

$$\begin{aligned}
 p_{POS}^b &= p(POS_{E_{i-1}} | E_i, E_{i+1}) \\
 &= \frac{\text{count}(POS_{E_{i-1}}, E_i, E_{i+1})}{\text{count}(E_i, E_{i+1})}
 \end{aligned}$$

where E_i is the source-null English word which is “be” in the example. E_{i-1} and E_{i+1} are the candidate English words to which E_i can attach. Forward and backward entropies are estimated as:

$$H_{POS}^x = - \sum_{POS} p_{POS}^x \log(p_{POS}^x)$$

where $x = f$ or b for forward or backward POS entropy models respectively. If $H_{POS}^f > H_{POS}^b$ we append the source-null word to the previous English word else we merge it with the following English word. The argument is that if $H_{POS}^f > H_{POS}^b$ then the word sequence E_{i-1}, E_i has been observed more commonly in different contexts than the word sequences E_i, E_{i+1} .

A comparison of translation accuracies using three strategies is done for the Spanish-to-English, English-to-Spanish and Arabic-to-English translation task. The POS entropy model outperforms the other two strategies with statistically significant results. However, when other features are used with the translation model in the log-linear framework the gains are smaller.

Gispert and Mariño (2006) also investigated the performance of the POS entropy model for target-null words (unaligned words on German side) and found that removing target-null words from the tuple corpus (by appending to right or left tuples) cause a drop in translation accuracy. This validates the hypothesis that handling of spurious words can be helpful in machine translation.

3.2 Reordering Model

Lack of reordering capability in the earlier N-gram-based SMT systems makes them less useful for translating language pairs with different syntactic structures. Let us revisit the example from last chapter. Given the bilingual sentence, and the word alignments shown in Figure 3.3, N-gram-based SMT extracts just one tuple containing the entire sentence pair. This occurs due to the crossing alignment “noch weiter – further” which aligns the first word of the German sentence to the last word of the English sentence. Such displacements are quite common when translating from German to English where the verb has to be reordered from the end of the sentence.

	noch	weiter	gehen	zu	wollen	,	wäre	ebenso	unverantwortlich
it									
would							■		
be							■		
just								■	
as								■	
irresponsible									■
to				■					
wish					■				
to			■						
go			■						
further	■								

Figure 3.3: Symmetrized Word Alignments with Union

Notice that the N-gram translation model can not learn any dependencies in such scenarios. The fall back approach of dealing with the embedded words (see Section 3.1.2), appends translation units as unigram tuples, to the tuple corpus and assigns all such units a uniform probability. Given the scenario, N-gram SMT can not achieve a better performance than word-based models. If the exact same sentence appears in test set, the decoder will not be able

to produce the output sentence as observed during the training. The pruning heuristic will prune out the only learned tuple because its frequency falls below the threshold limit. The decoder only has embedded word units in its unigram tuple inventory. The most likely output according to the translation model is a monotonic sequence of tuples: “further go to to wish it would be just as irresponsible”.

The ability to learn lexicalized reorderings is also desirable for the translation of language pairs that exhibit short distance movements. Due to data sparsity it is not possible to see all possible word sequences and word permutations. Consider a short training sentence with the tuple segmentation as shown in Figure 3.4. The reordering of “gegessen – eaten” is local to the tuple and is useful merely when the exact same tuple appears during test. However if the test sentence appears with a different noun such as “erdbeerkuchen – strawberry cake” as in “er hat einen erdbeerkuchen gegessen”. The translation model does not know how to reorder the verb “gegessen” anymore. In absence of any reordering mechanism the most likely translation would be “he has a strawberry cake eaten”. This is because the tuple sequences $\langle \text{hat} - \text{has} \rangle \langle \text{einen} - \text{a} \rangle$ and $\langle \text{einen} - \text{a} \rangle \langle \text{erdbeerkuchen} - \text{strawberry cake} \rangle$ get good tuple bigram probabilities. The language model also prefer the word sequence “has a” more than “has eaten”.

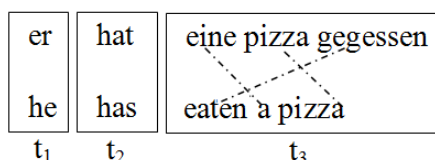


Figure 3.4: Short Distance Reordering

3.2.1 Source Linearization and Tuple Unfolding

A reordering framework based on linearization of the source-side (German) was proposed to address these problems (Crego et al., 2005c). The idea is to

3 N-gram-based SMT

change the order of the German sentence so that it appears in the same order as the English sentence. Linearization is a two step process:

1. All the words on the German side that are aligned to the same English word(s) form a group and all the words on the English side that are aligned to the same German word(s) form a group. The process is iteratively repeated unless no new groups are formed. Each group represents a tuple.
2. Tuples are output maintaining the English word order.

See Figures 3.5 for an illustration of the linearization process. The translation model is now estimated on the newly formed tuple corpus. The advantage of linearization is dual. First it unfolds the embedded tuples alleviating the data sparsity problem. Secondly the tuple-based translation model can now also be used to score different word permutations.

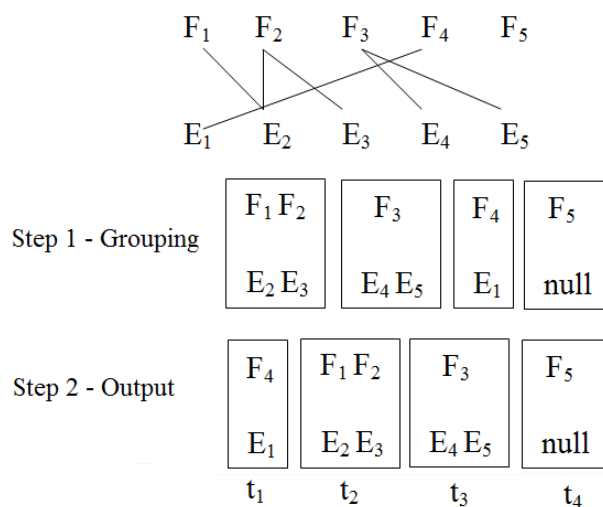


Figure 3.5: Source Linearization - Two Steps

Tuple unfolding eliminates the crossing-alignments and now there is no need for appending embedded words to the tuple corpus. See Figure 3.6 for the illustration of source linearization of the example shown in Figure 3.4. The ordering

of German words in tuple t_3 is changed so that it appears in the same order as its English counterpart. This enables the embedded words, initially blocked by the crossing-alignment, to form tuples of their own. Tuple t_3 splits into three separate tuples. The second advantage is that the n-gram estimates obtained from the tuple corpus now indirectly provide information on the reordering. After translating the tuples $\langle er - he \rangle$ and $\langle hat - has \rangle$, there is now a supporting evidence in the translation model for the decoder to hypothesize a tuple $\langle gegessen - eaten \rangle$. The idea of linearization of source has also been applied in other machine translation systems (Collins et al., 2005; Kanthak et al., 2005).

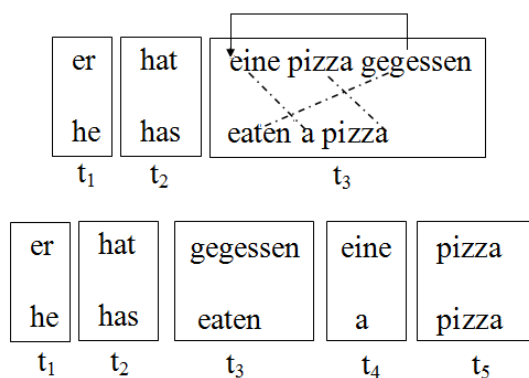


Figure 3.6: Source Linearization of Example in Figure 3.4

3.2.2 Rewrite Rules

Hypothesizing all possible reorderings during decoding is not only computationally expensive but also results in more search errors. Despite the hard reordering constraints, such as the distortion limit (reordering window of 5 words) and the reordering limit (allowing up to 3 jumps) (Crego et al., 2005c), there is still a difficult search problem causing a performance drop as compared to using regular tuples (without unfolding) and monotonic translation. Crego and Mariño (2006) has reported translation accuracy to drop by more than a

BLEU point for Spanish-to-English and English-to-Spanish tasks, when using tuple unfolding with reordered search.

Crego and Mariño (2006) proposed the use of rewrite rules to guide the search process, rather than trying all possible word permutations with brute-force. Rewrite rules are extracted during the linearization of a German sentence. A rewrite rule is composed of a left and right hand side. The left-hand side is formed of German words in a tuple t_i which is required to be unfolded. Let the German words in the tuple t_i have indexes $0, 1, \dots, n$. The right hand side of the rule is composed of the indexes of the German words in the tuple t_i after it is unfolded into smaller tuples. For example the tuple t_3 in Figure 3.6, forms a rewrite rule such as:

$$\text{eine pizza gegessen} \mapsto 2\ 0\ 1$$

This rule means that “gegessen” gets displaced to the beginning of the tuple while the ordering of the other two words remains the same with respect to each other. The idea of rewrite rules is to hypothesize only the reorderings that have been observed during training. The search graph in decoding is built on the base of rewrite rules. Each rewrite rule forms its own arc. Once all the rules have been applied, the graph is searched monotonically with different arcs. A graph for the given example is shown in Figure 3.7. The search graph is instantiated with the monotonic path, the input sentence “er hat eine pizza gegessen”. For the rewrite rule, observed during training and applicable to the test sentence, the search graph is extended with an arc “gegessen eine pizza”. As a result of this the search space is dramatically reduced.

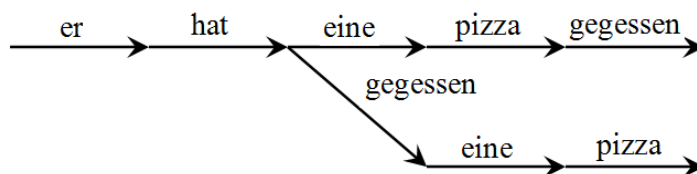


Figure 3.7: Search Graph Extension

POS-based Rewrite Rules

One major drawback of the rewrite rules based on word forms is that they can not generalize well to unseen data. Coming back to the example sentence “er hat einen erdbeerkuchen gegessen”. The learned rule is not applicable to this example because of a different article “einen” and a different noun “erdbeerkuchen”. While the brute force method will hypothesize reordering “gegessen” after translating “er” and “hat”, the arc-based search mechanism will rule out this possibility because it does not have a corresponding rule in its inventory.

In order to remedy this problem and improve the generalization power of the rewrite rules, POS tags are used instead of word forms. The above rule is now recorded² as:

$$\text{DT NN VBN} \mapsto 2\ 0\ 1$$

This enables the N-gram-based SMT to generalize to unseen text during decoding. However, POS-based rules are less accurate than their word-based counterparts. The word sequence “eine pizza gegessen” will most certainly translate as “eaten a pizza” or “ate a pizza” both satisfying the rule $\text{eine pizza gegessen} \mapsto 2\ 0\ 1$. However, it is easy to find a word sequence where the POS-based rewrite rule would fail. See Figure 3.8 for example. The search graph is extended for the word sequence “die Vereinbarung beendet” using the rule $\text{DT NN VBN} \mapsto 2\ 0\ 1$. According to this arc the best translation would be “if terminated the agreement is”. The right translation is achieved using the dotted arch which swaps “bendet” and “wird”.

In order to obtain an accurate search graph only reliable rules are used. Rules are filtered based on the rule probability. All rules falling below the cut-off threshold are dropped. Let X_1, X_2, \dots, X_n be the left hand side of the rule composed of word-forms or POS tags and j_1, j_2, \dots, j_n be the set of

²DT = Determiner, NN = Noun, VBN = Verb Past Participle

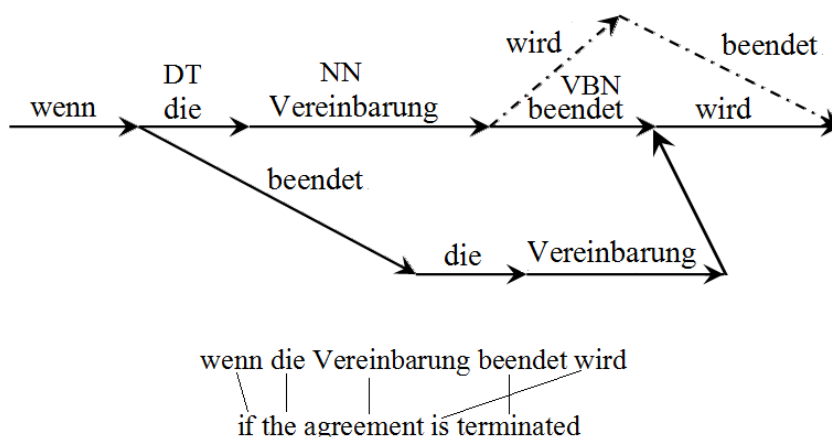


Figure 3.8: Search Graph Extension - Rule Fail

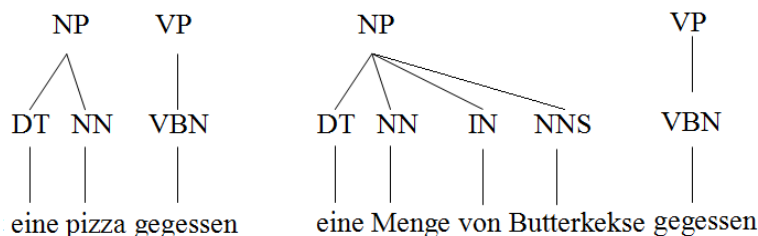


Figure 3.9: Rewrite Rules - Phrasal Chunks

relative indexes of a tuple t_i which is unfolded during training. The reordering probability (Crego and Mariño, 2006) is estimated as:

$$p(X_1, X_2, \dots, X_n \mapsto j_1, j_2, \dots, j_n) = \frac{N(X_1, X_2, \dots, X_n \mapsto j_1, j_2, \dots, j_n)}{N(X_1, X_2, \dots, X_n)}$$

Syntax Enhanced Rewrite Rules

Using POS tags still imposes a constraint on the order and number of tags to be the same as observed during training. Thus the learned rule $DT\ NN\ VBN \mapsto 2\ 0\ 1$ can generalize to “einen Erdbeerkuchen gegessen” but not to

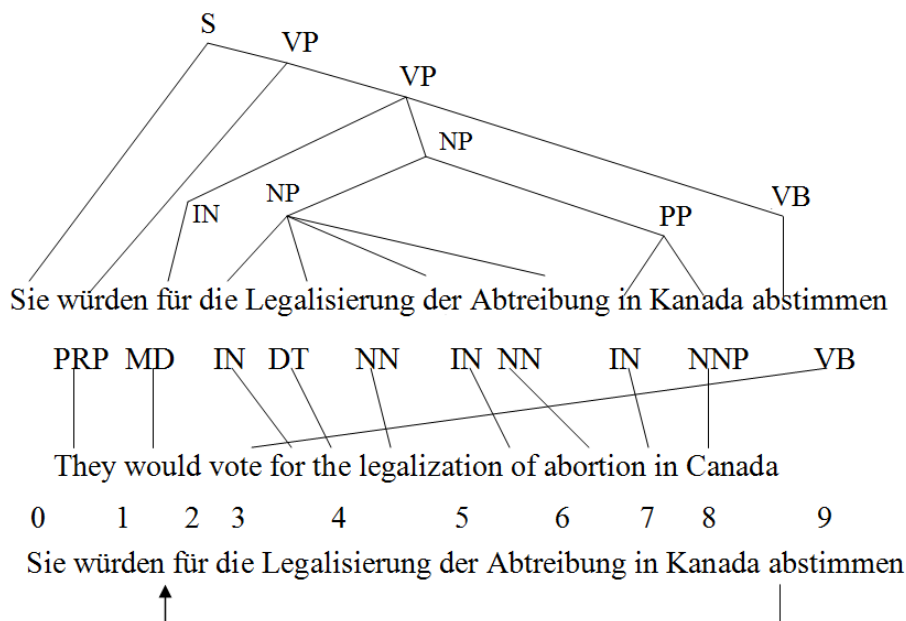


Figure 3.10: Syntax Enhanced Reordering

“eine Menge von Butterkekse gegessen”, because of the additional number of German words in the test sentence. Another rule such as:

$$DT\ NN\ IN\ NNS\ VBN \mapsto 4\ 0\ 1\ 2\ 3$$

is required to hypothesize this reordering. POS-based rules can be effective to handle very short distance reorderings that occur frequently, however long distance reorderings that involve jumping over many words would raise sparsity problems.

This problem can be solved by going to the next level of generalization by using chunks or dependency parse trees, to form linguistic rules. Notice the common pattern in the two examples in the Figure 3.9. At higher level the reordering is being carried out by swapping the verb and noun phrases. Therefore instead of learning rules based on POS tags, chunk level rules such

as $\text{NP VP} \mapsto 1\ 0$ can be learned.

The idea of using syntax trees (Crego and Mariño, 2007) to form rewrite rules, is motivated by the need to perform long distance reorderings which can not be captured by the POS tags because of data sparsity. The left hand side of the rule is now composed of the syntactic tags. Consider Figure 3.10 for example. In order to monotonize the German sentence the main verb “abstimmen” must be reordered by a jump from index 9 to index 2. A sub-tree spanning all the German words that are jumped over, is identified. In this case it is the second verb phrase spanning “für ... abstimmen”. The left hand side of the rule is composed of the syntactic category of each German word in the span along with its root node. Root node denotes the German word (“abstimmen” in the current example) which has to be moved in order to achieve linearization. The right hand-side specifies the relative indexes of the syntactic categories after unfolding. The extracted rule for this example is:

$$\text{IN DT NN DT NN IN NNP root} \mapsto 7\ 0\ 1\ 2\ 3\ 4\ 5\ 6$$

The surface rule is exactly the same as the POS-based rule. However, different levels of generalizations can be achieved by merging categories to form bigger categories. See Figure 3.11 for different rules learned from the linearization of “abstimmen”. Each rule provides a different level of abstraction. Learning generalized rules such as $\text{IN NP root} \mapsto 2\ 0\ 1$ are helpful for learning long distance reorderings. Crego and Mariño (2007) have shown significant improvements using syntax-enhanced rules over their POS-based counterparts for a Chinese-to-English translation task.

3.2.3 Handling of Gappy Units

Source Side Discontinuities

Linearization of the source-side (German), not only unfolds the embedded tuples and introduces a reordering framework but also handles source-side discontinuities inherently. Recall the procedure of unfolding. The first step

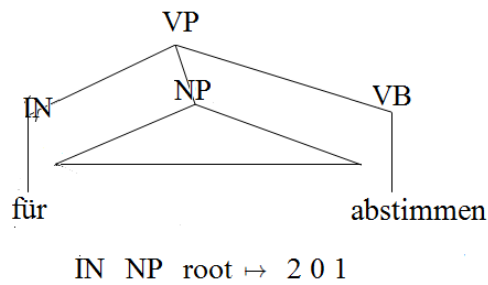
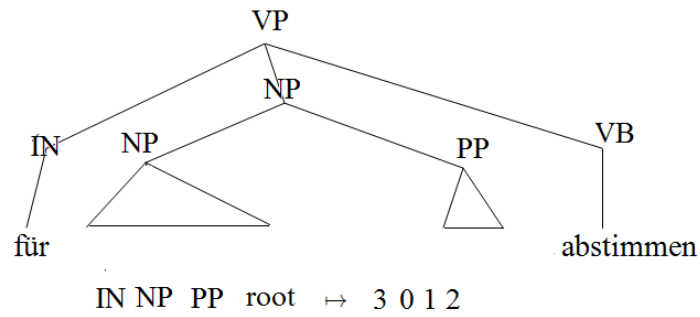
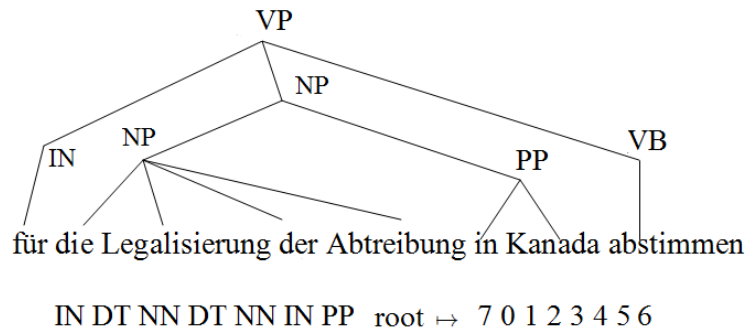


Figure 3.11: Extracted Rules – Different Levels of Generalization

3 N-gram-based SMT

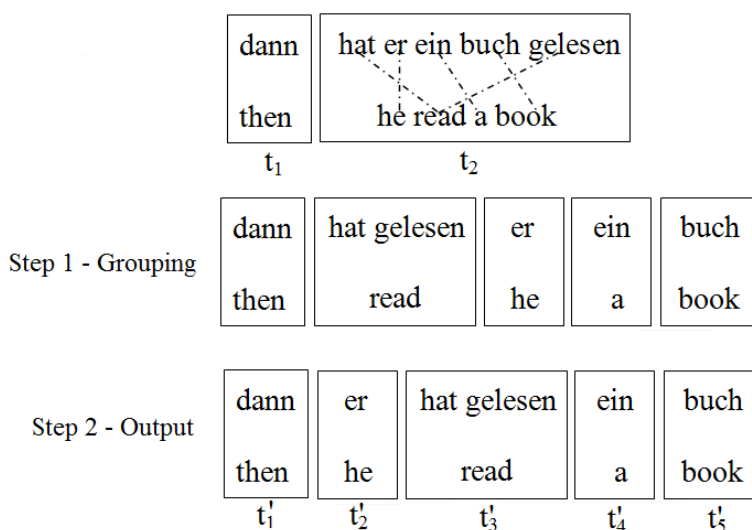


Figure 3.12: Source-side Gaps

groups the German and English words in tuples such that each group contains only German and English words that are aligned to each other and no words from other groups align to the words in this group. This step eliminates any German side discontinuity to produce a linear sequence. See Figure 3.12. Tuple t_2 contains the discontinuous translation unit “hat...gelesen – read”. Step-1 groups “hat” and “gelesen” together in a single group because they align to “read”. This eliminates the discontinuity in the final output. The rewrite rule learned along the unfolding procedure is:

$$\text{VBZ PRP DT NN VBN} \mapsto 0 \ 4 \ 1 \ 2 \ 3$$

This rule can be successfully used during decoding for translation of the test sentence “dann hat er eine märchenbuch gelesen”. But using a fixed number and order of POS tags causes the same generalization problem as before.

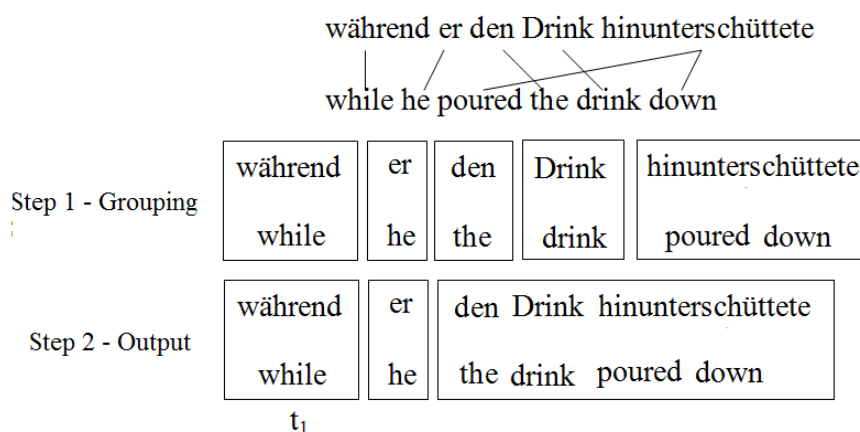


Figure 3.13: Target-side Gaps

Target Side Discontinuities

Target side discontinuities can not be enabled in the reordering framework so far discussed. The method of linearization applied to the German side can be used principally to unfold the English side and for enabling discontinuities. However, the generation of English side can then no longer be done in a left-to-right manner.

The so-far-discussed N-gram-based systems handle this by merging all the groups, that appear between the two English words, together to form a single group. See Figure 3.13 for illustration. Although the grouping stage bundles “hinunterschüttete – poured down” in a separate tuple but this tuple can not be output without violating the condition of maintaining the order of English words. The order of English words can be maintained by extracting only one tuple (See t_3 in Figure 3.13) which merges all the tuples that exist within.

This method of merging tuples can turn into a big drawback when the discontinuous English words are far apart. No useful tuple can be extracted. Crego and Yvon (2009) propose a method to handle this by splitting German words that are aligned to discontinuous English side words. Each German

3 N-gram-based SMT

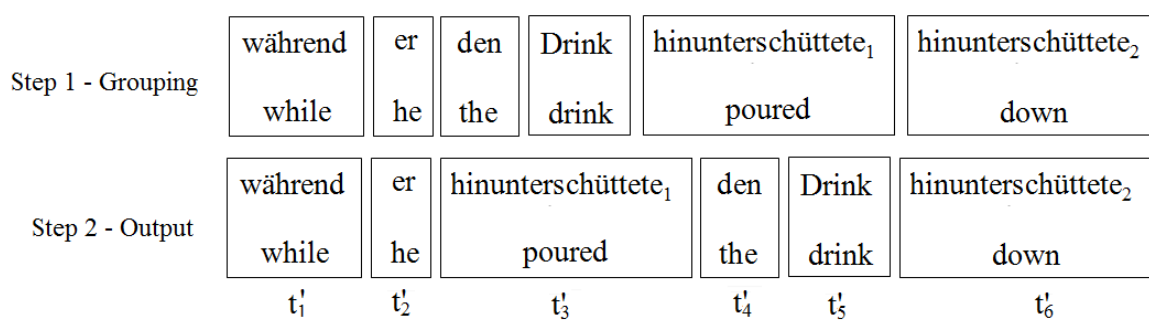


Figure 3.14: Handling Target Gaps Using Split Tokens

word F_i connected to several English words that are discontinuous is split into n tokens where n is the number of discontinuities on the English side. Each split token F_{i_x} aligns to one contiguous set of English words in the order left to right. This step is introduced before step 1 which groups German and English words into tuples. Each split token F_{i_x} now forms its own tuple. See Figure 3.14 for illustration. German word “hinunterschüttete” aligned to “poured...down” is split into two tokens “hinunterschüttete₁” and “hinunterschüttete₂” such that “hinunterschüttete₁” now aligns with “poured” and “hinunterschüttete₂” aligns with “down”. Each instance of “hinunterschüttete” forms its own tuple. Step 2 outputs the tuples in the order of the English sentence moving the tuple “hinunterschüttete₁ – poured” to the middle of the sentence after the tuple “er – he”, thus maintaining the English sentence order.

Using split tokens unfolds the embedded tuples and provides a mechanism of handling discontinuities on the output side. Similar to the reordering rules split rules are also learned with POS tags and syntactic categories to generalize well on the unseen test. However, the identity of the German token that splits is maintained in these rules to prevent spurious generalizations. For example the learned rule for the running example would be:

$$\text{DT NN hinunterschüttete} \mapsto 2_1 \ 0 \ 1 \ 2_2$$

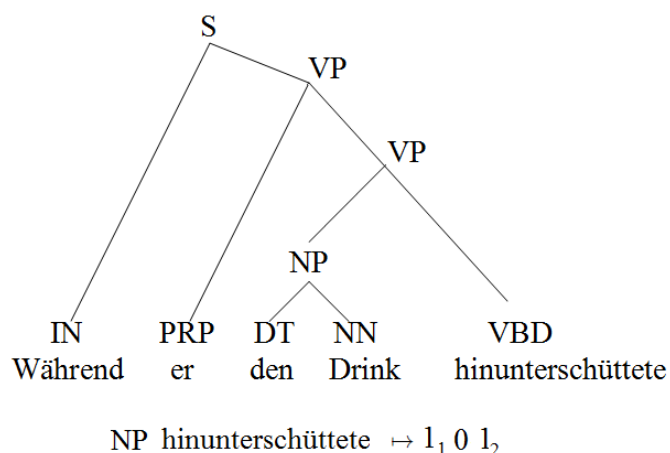


Figure 3.15: Syntax Aware Split Rules

More generalized rules can be obtained by using syntactic categories to merge POS-tags as before. For example “den Drink – DT NN” can form a noun phrase resulting in the rule shown in Figure 3.15.

3.2.4 Factored Bilingual Units

Using POS-tags and syntactic categories in rewrite rules helps improve the ability to generalize. However, although using POS-based reordering rules enables the decoder to hypothesize useful reordering patterns, the translation model might not have any evidence to support such a reordering. See Figure 3.16. Although a POS-based reordering rule $DT\ NN\ VBN \mapsto 2\ 0\ 1$ is learned during tuple unfolding and is successfully applied to hypothesize the reordering “fabriziert” during test, the translation model estimated from the bilingual corpus might not have any evidence to support this reordering if “fabriziert” is an unknown word or not seen with this reordering pattern. The decoder might translate “fabriziert” monotonically resulting in an output sentence “we have a pizza fabricated”. This is likely to happen because the tuple sequence $\langle wir - we \rangle \langle haben - have \rangle \langle eine - a \rangle$ is a commonly occurring tri-

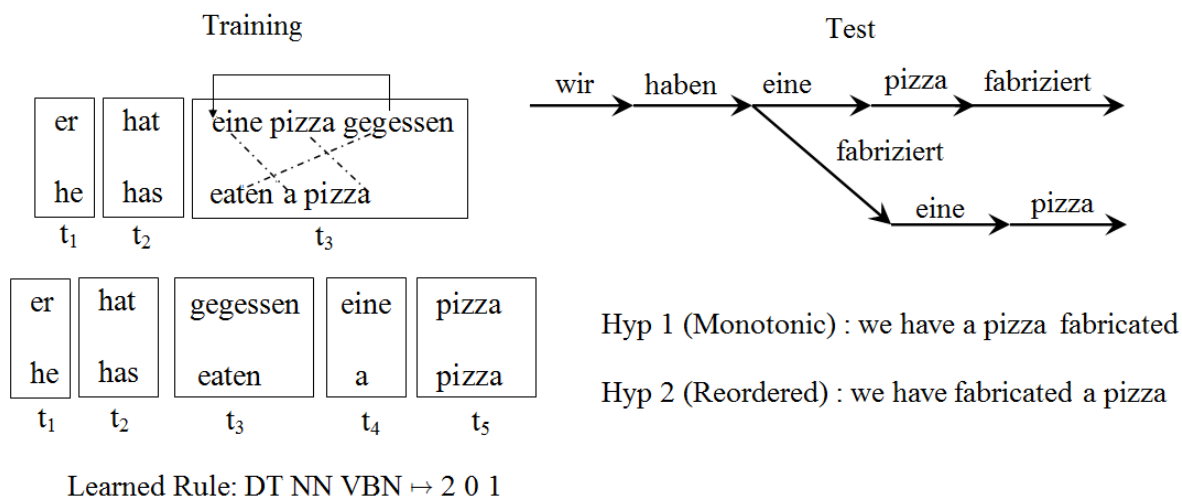


Figure 3.16: No Evidence of Reordering During Test

gram. The English-side language model also tends to favor “we have a” more than “we have fabricated”. This problem can be alleviated by introducing POS-tags and other bilingual linguistic categories directly in the translation model. Crego and Yvon (2010) use POS tags in place of word forms when estimating the tuple corpus. The POS-based translation model computes an n-gram probability over a sequence of POS-tags of tuples given as:

$$p_{bLM}(e, f) \approx \prod_{j=1}^J p(t_j | t_{j-m} \dots t_{j-1})$$

where each bilingual $t_j = \langle pos(F_j), pos(E_j) \rangle$ couples POS tags for German and English strings, m indicates the amount of context used. Similar to the translation model, the POS-tuple model is also implemented as an N-gram model using the SRILM toolkit with Kneser-Ney smoothing. It is used as an additional feature. Factored translation models have also helped phrase-based machine translation to improve translation quality (Koehn and Hoang, 2007).

3.3 Features Used in N-gram-based SMT

In this section we enumerate the features that are used in N-gram-based SMT that are introduced to improve end-to-end translation accuracy. These feature functions are combined in a log-linear model and are trained with MERT.

1. **Target-side Language Model** : English side language model based on n-gram statistics estimated with the SRI-Toolkit (Stolcke, 2002) with Kneser-Ney smoothing. 5-grams are typically used.
2. **Translation Model** : The tuple N-gram language model is the central feature of N-gram-based SMT. It is estimated from the unfolded tuple corpus. The translation probabilities are estimated using an n-gram model. The tri-grams are typically used.
3. **POS-tagged Target-side Language Model**: Target-side language model estimated from POS tags instead of word forms.
4. **POS-tagged Source-side Language Model**: German-side language model estimated from POS tags instead of word forms. This is estimated from the German side of the unfolded tuple corpus. This means that German has been linearized according to the English order.
5. **Lexical Translation Probabilities**: These are exactly the same as used in the phrase-based system. These probabilities are estimated from the word-alignments based on IBM Model 1.
6. **Word Bonus**: Word bonus feature controls the bias of the target-side language model against longer outputs.
7. **Reordering Distance**: This feature calculates the distance by which a word is displaced after applying a reordering rule to add a new arc. The distance is calculated as $|j - R(j)|$ where j is the original index of the German word F_j and $R(j)$ its index after applying a reordering rule.

3.4 Search

In this section we will discuss the most recent decoding mechanism (Crego et al., 2011) for N-gram-based SMT. Like phrase-based and other SMT systems N-gram-based SMT also tries to find an output that maximizes the weighted sum of the feature components:

$$\operatorname{argmax}_e p(e|f) = \operatorname{argmax}_e \left\{ \sum_{j=1}^J \lambda_j h_j(f, e) \right\}$$

$h_j(f, e)$ is the log-scaled probability of the feature components enumerated in the previous section. Decoding for the N-gram system is a two step process:

1. Reordering the input sentence to form word lattices
2. Searching the word graph

3.4.1 Reordering

Given a German sentence the first step is to form a search graph. The search space in N-gram-based SMT is not built dynamically during search but constructed in a preprocessing step. The input sentence is encoded as a word lattice with the help of reordering rules learned during training. The input sentence is POS-tagged or parsed (if syntactic rules are to be applied). Next the search graph is initialized with the path that constitutes the monotonic word order as it appears during test. The monotonic arc is extended with different word permutations obtained by applying reordering rules to any part of the test sentence. Larger rules are applied first so that the process can repeat recursively to the newly formed arcs. Rules can be filtered using rule probability or size of the rule (number of German or English words). See Figure 3.17 for the illustration, when rules in Table 3.1 are applied (word forms are used instead of POS-tags for simplicity). The larger rule “gestern hat er eine pizza gegessen \mapsto 1 2 3 4 5 0” is applied first which forms a new arc. The smaller

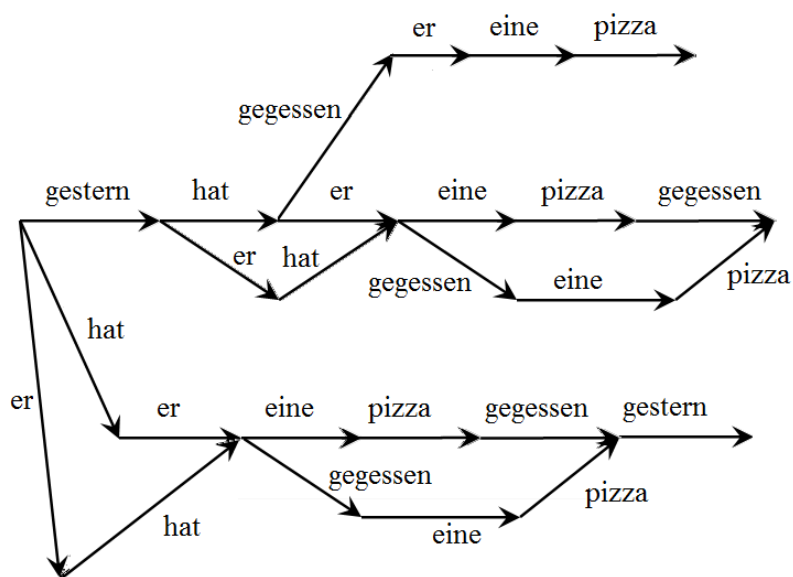


Figure 3.17: Lattice-based Search Graph

rule “eine pizza gegessen \mapsto 2 1 0” is then applied on both the monotonic and initially reordered word permutations. The process is recursively repeated until no new rules could be applied.

Reordering Rule	Permuted Sequence
gestern hat er eine pizza gegessen \mapsto 1 2 3 4 5 0	hat er eine pizza gegessen gestern
hat er eine pizza gegessen \mapsto 0 4 1 2 3	hat gegessen er eine pizza
eine pizza gegessen \mapsto 2 1 0	gegessen eine pizza
hat er \mapsto 1 0	er hat

Table 3.1: Reordering Rules

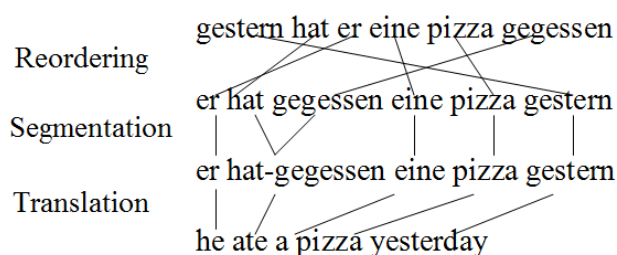


Figure 3.18: Abstract Translation Process

3.4.2 Searching the Word Graph

Once the word lattice has been established, the next step is to search for the optimal path. All applicable tuples are extracted from the tuple corpus. Because multiple German words can be part of a tuple and thus required to be translated in single step, a segmentation step is required to group the words into tuples. For example “hat” and “gegessen” can be translated independently of each other as “has” and “eaten” respectively. But they can also translate as a joint unit as “hat gegessen...ate”. In the latter case, “hat gegessen” should be merged. The next step is to hypothesize English translations and calculate the language model score. All other features can be estimated in advance before the search step. See Figure 3.18 for illustration of the overall translation process.

Hypotheses are maintained in stacks. Each stack contains hypotheses that translate the same German words. Therefore theoretically at most 2^n stacks are required for a German sentence containing n words. An alternative to this configuration is to maintain stacks such that each stack translates the same number of German words. This however, results in an unfair comparison, for the decoder would prefer hypotheses that translate easier parts of the German sentence and prune out the hypotheses translating the difficult parts. A future cost estimate is required to overcome this problem (See Section 2.2.7 for future cost estimation in phrase-based system). *N*-gram SMT models do not use future cost estimation (Crego et al., 2011). Therefore arranging hypotheses in 2^n stacks is necessary to prevent search errors. This solution is feasible con-

sidering that the search space in *N*-gram-based models is highly constrained by applying only the pre-calculated orderings. Most of the 2^n stacks are never utilized to extend any hypothesis. The search space is further constrained by applying histogram pruning and limiting the tuple translation options to using the *n*-best translation options for each sequence of German words. Recombination is also applied just as in the phrase-based system. All hypotheses that cover the same German words and have the same language and translation model contexts can be merged.

3.5 Comparison with Phrase-based SMT

In this section we will discuss the properties of *N*-gram-based SMT in comparison with the phrase-based system. We will highlight important difference in terms of translation units, reordering approach and decoding.

3.5.1 Translation Model

The main difference between phrase-based and *N*-gram-based SMT is the statistical modeling of the translation units. Phrases are larger translation units that memorize useful information such as local reordering, insertions, deletions, handling of local gaps etc. Tuples on the other hand are minimal bilingual units that do not represent the above phenomenon directly but through an *n*-gram language model.

Independence Assumption

A drawback of the phrase-based system is that it makes independence assumptions over phrases. Because of the phrasal segmentation phrase-based system can not capture dependencies in some cases. Assume that the sequences “er hat – he has” and “hat gegessen - has eaten” appear very commonly in the bilingual text but “er hat gegessen – he has eaten” is unseen. If this sequence appears during test, the phrase-based system can not capture both of these

dependencies simultaneously. It has to choose one of the segmentations. On the contrary the *N*-gram model can capture both these dependencies because it does not make any independence assumption (other than context size) over tuples and has access to the previously generated tuples.

Bilingual Context

Because of the phrasal independence assumption, phrase-based system ignores the contextual information on the German side outside phrases. Contextual information on the English side is obtained through a English-side monolingual language model. The reordering, is thus justified only through the language model. On the contrary *N*-gram model makes use of bilingual contextual information also taking into account which German words were previously translated. Feng et al. (2010) addressed this by adding a German-side language model for the phrase-based system and showed improvements in BLEU score. Words in the German sentences are first permuted to make their order identical to the English side. The German-side language model is added as a feature in the log-linear equation.

Spurious Phrasal Segmentation

An advantage of *N*-gram-based SMT over phrase-based SMT is that tuples do not overlap unlike phrases. Each bilingual sentence pair given the alignment has exactly one possible segmentation. This enables the *N*-gram-based SMT to avoid spurious segmentations during training. The number of translation units extracted by the Phrase-based SMT is many times more than that extracted by the *N*-gram-based system. See Crego et al. (2005a) and Crego et al. (2011) for a comparison of extracted translation units and Section 3.5.3 for more on the phrasal segmentation problem. Table 3.2 shows the extracted translation units, for the sentence pair shown in Figure 3.19, under both models. Phrases of up to 6 German words are extracted.

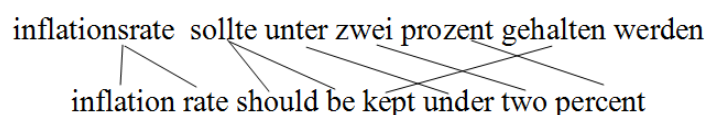


Figure 3.19: Training Example

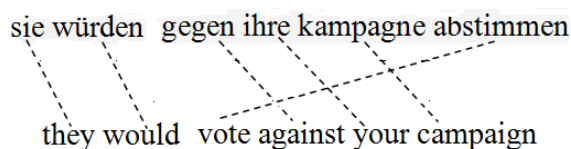


Figure 3.20: Handling Local and Non-local Reorderings

Reordering

Memorizing phrases enables the phrase-based system to learn local reorderings. However, non-local reorderings are not represented by the translation model but handled through other models such as lexicalized reordering and the language model. This is one of the drawbacks of the phrase-based model (See section 2.3.1 for details). N-gram-based models learn reordering through source linearization. Monotonizing German according to the English order enables the N-gram model to represent both local and non-local reorderings through the translation model. Having observed the bilingual sentence pair shown in Figure 3.20, during training, the phrase-based system learns the reordering of “abstimmen – vote” only through the phrase “gegen ihre kampagne abstimmen” which can not generalize to the test sentence “sie würden gegen meine außenpolitik abstimmen”. On the contrary N-gram-based machine translation can generalize to this test sentence by learning an unfolded tuple sequence $\langle sie - they \rangle \langle würden - would \rangle \langle abstimmen - vote \rangle$. If the reordering of “abstimmen” is hypothesized, the translation model for the N-gram SMT has the evidence to support this reordering. The same is not true for the phrase-based system.

3 N-gram-based SMT

Phrases	Tuples
inflationsrate – inflation rate	inflationsrate – inflation rate
sollte – should be	sollte – should be
unter – under	unter – under
zwei – two	zwei – two
prozent – percent	prozent – percent
gehalten – kept	gehalten – kept
prozent – percent	prozent – percent
gehalten werden – kept	werden – null
inflationsrate sollte – inflation rate should be	
unter zwei – under two	
unter zwei prozent – under two percent	
zwei prozent – two percent	
unter zwei prozent gehalten – kept under two percent	
unter zwei prozent gehalten werden – kept under two percent	
sollte unter zwei prozent gehalten – should be kept under two percent	
sollte unter zwei prozent gehalten werden – – should be kept under two percent	
inflationsrate sollte unter zwei prozent gehalten – inflation rate should be kept under two percent	

Table 3.2: Extracted Phrases and Tuples

However, by linearizing the source, N-gram-based SMT gives up on the true representation of the data. The information that “abstimmen – vote” follows discontinuously after “würden – would” is not represented in the translation model. On the contrary hierarchical (Chiang, 2005) and discontinuous phrase-based system (Galley and Manning, 2010) can represent this information as “würden X abstimmen – would vote”. Moreover, the non-terminal “X” in the discontinuous phrase gives more flexibility to “würden” and “abstimmen” to be apart from each other by any number of words. In contrast the N-gram

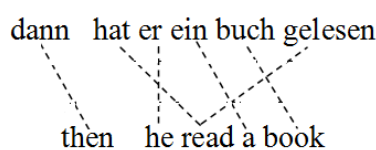


Figure 3.21: Handling German-Side Discontinuities

model requires the test sentence to occur with the same sequence of POS-tags as occurred during training, for it to fire a rule.

Handling Discontinuities

The phrase-based system memorizes discontinuities both on the German and English side inside of phrases. Discontinuities across phrases can not be handled by the traditional phrase-based system (See Section 2.3.2). The N-gram-based model handles discontinuous German side phrases through source linearization and English-side discontinuities through split tokens. Source linearization helps the N-gram model to represent both local and non-local source-side discontinuities. From the training example of Figure 3.21 N-gram-based SMT can linearize the German side and learn a tuple translation $\langle \textit{hat} \dots \textit{gelesen} - \textit{read} \rangle$ which can be used to generalize to the test sentence “dann hat er eine märchenbuch gelesen”.

Discontinuities on the English side are represented in the N-gram model, by splitting German words that align to discontinuous English words. This approach has a minor pitfall. If the split tokens are apart by more than 3 tuples, the translation model can not capture the dependency. Discontinuous phrase-based systems in comparison represents the phrase as “hinunterschüttete – poured X down”, modeling the dependency between “poured” and “down” irrespective of the number of intervening words that separate them.

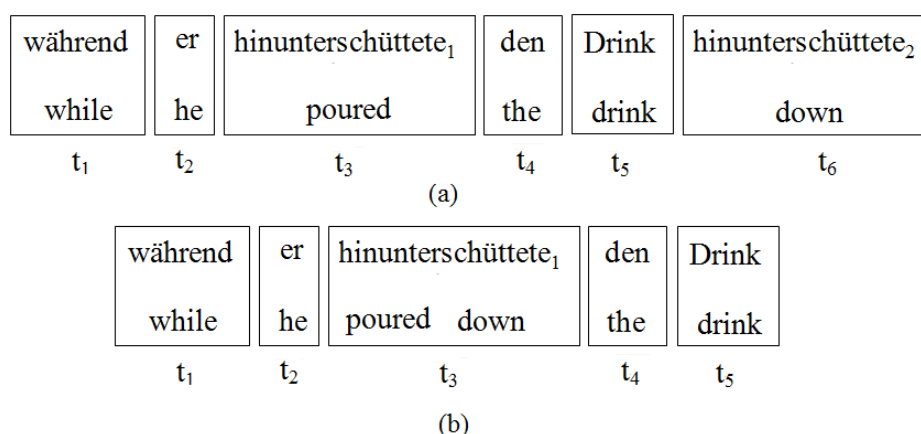


Figure 3.22: Handling English-Side Discontinuities

Handling Unaligned Words

The phrase-based system can memorize insertions and deletions of words inside of the phrase but can not delete or insert words outside phrases. The *N*-gram-based model can learn deletions of German words with context just as the phrase-based system but can also arbitrarily delete words in unseen contexts which can be helpful.

The phrase-based system can represent unaligned English words inside of phrases but *N*-gram models can not do the same, because of the minimal tuple restriction. Unaligned English words are attached left or right as a preprocessing step thus losing one of the dependencies. Consider a bilingual chunk “A C – a b c” (capital letters align only with their corresponding small letter). The unaligned target word “b” either attach to “a” to form a tuple “A – a b” or to “c” to form a tuple “C – b c”. On the contrary the phrase-based system preserves both attachment options by extracting two phrases “A - a b” and “C - b c”.

3.5.2 Reordering Framework

The reordering models used in phrase-based and N-gram-based SMT are not strictly part of these models, meaning one can be used inside another framework. Ideas of one paradigm have been borrowed by the other. Lexicalized reordering has been implemented for N-gram models (Crego and Yvon, 2010). Similarly unfolding of source has been experimented in a phrase-based system (Feng et al., 2010). In this section we compare these two reordering mechanism.

N-gram-based SMT couples reordering and search with the help of POS-based rewrite rules. Phrase-based SMT hypothesizes all possible reorderings in a fixed reordering window of 5-8 words. A major drawback of the N-gram-based approach is that search is only performed on a small number of reorderings that are pre-calculated on the source side independently of the target side. Often, the evidence for the correct ordering is provided by the target-side language model (LM). In the N-gram approach, the LM only plays a role in selecting between the pre-calculated orderings.

Secondly, the standard N-gram model heavily relies on POS tags to hypothesize reorderings. This limits the N-gram model to be used only for the language pairs for which POS-tags are available. Using POS tags improves the ability to generalize, however, POS-based rules still face data sparsity problems specially for long distance reordering. Recall our discussion from section 3.2.2 that motivates syntax-based rules, a POS rule such as DT NN VBN \mapsto 2 0 1 can generalize to “einen Erdbeerkuchen gegessen” but not to “eine Menge von Butterkekse gegessen”, because of the additional number of German words in the latter sequence. Despite that both reorder “gegessen” over a noun phrase. This problem is addressed by using syntactic trees which are only available for a few language pairs.

Data sparsity in POS sequences also hinder the N-gram model from hypothesizing source-side discontinuities. Recall our discussion on source-side gaps from Section 3.2.3. A rule such as VBZ PRP DT NN VBN \mapsto 0 4 1 2 3 can generalize to reorder “gelesen” in the sequence “hat er ein Buch gelesen” but can not generalize to “er hat ein Buch gelesen” because of different number of

words. A different POS rule would be required for a different number of words. If a rule fails to trigger, N-gram model can not obtain an arc that monotonize “hat” and “gelesen”. Consequently the following step that segments “hat gelesen” into a single unit can not execute. See Figure 3.18 in Section 3.4.2 for a similar example. For the segmentation module to collapse “hat gegessen” as a single unit, it must first be reordered to its correct position following “hat”. If the POS rule fails, then “hat gegessen” can not collapse and the decoder can not hypothesize a tuple “hat gegessen - ate”. Consequently “hat” translates to “has” and “gegessen” translate to “eaten”. However, “hat...gegessen - ate” is the right translation for the given example.

Similarly split rules for handling target-side discontinuities are based on POS-tags and will have the same problem. Refer to Figure 3.22(a) again. The captured rule “schalten PRP DT JJ NNS $\mapsto 0_1 1 2 3 4 0_2$ ” can not generalize to “schalten sie ihr handy aus – turn your cellphone off”. If the POS-rule fails then the decoder can not hypothesize splitting “schalten” and is unable to produce “turn” and “off” discontinuously but only monotonically translating as “turn off your cell phone” which is a more acceptable translation but it is easy to find an example where the resulting output is undesirable. Consider translating from English to German. In a sentence pair “you are looking good – sie sehen gut aus”. The verb “looking” aligns to “sehen...aus”. To generate the discontinuous tuple “looking” must be split to “looking₁” and “looking₂” so that “looking₁” produces “sehen” and “looking₂” generates “aus”. If the split rule fails the decoder translates “looking” to “suchen”.

The reordering mechanism of the N-gram model, however is more informative than the lexicalized reordering model because it takes into account previously translated German and English words. On the contrary the lexicalized reordering model merely records orientation of each phrase or translation unit with respect to the previously translated phrase without taking previous translations into account. See Section 2.3.3 for a complete discussion on the drawbacks of the lexicalized reordering model.

3.5.3 Decoding

The decoding framework of N-gram-based SMT is fairly different than that of the phrase-based system. In this section we enumerate these differences.

Search space

Search space in phrase-based SMT is built dynamically. All possible reorderings within a window of 5-8 words are hypothesized. On the contrary the search space in the N-gram-based SMT is built as a preprocessing step using POS-based rules. This makes the decoding space extremely compact contributing towards efficiency. However, search is only performed on a small number of reorderings that are pre-calculated on the source side independently of the target side. Often, the evidence for the correct ordering is provided by the target-side language model (LM). In the N-gram approach, the LM only plays a role in selecting between the pre-calculated orderings.

Spurious Phrasal Segmentation

The search space in N-gram-based SMT is also reduced by the fact that only minimal translation units are hypothesized. This also helps the N-gram model to avoid spurious segmentations in the search graph. In comparison different compositions of the same phrasal unit exist and compete with each other in the phrase-based system (See Section 2.3.5). See Figure 3.23 for decoding of the test sentence “inflation sollte unter zwei prozent” under both models (only monotonic paths are shown). Because the N-gram model uses only minimal translation units, there is only one possible segmentation (shown at the bottom of the figure). On the contrary Phrase-based SMT produces the exact same output with different possible segmentations. Although this problem is eased with recombination, however, recombination can take place only after the hypothesis has been created and the values for all its feature components have been computed, thus making decoding inefficient.

3 N-gram-based SMT

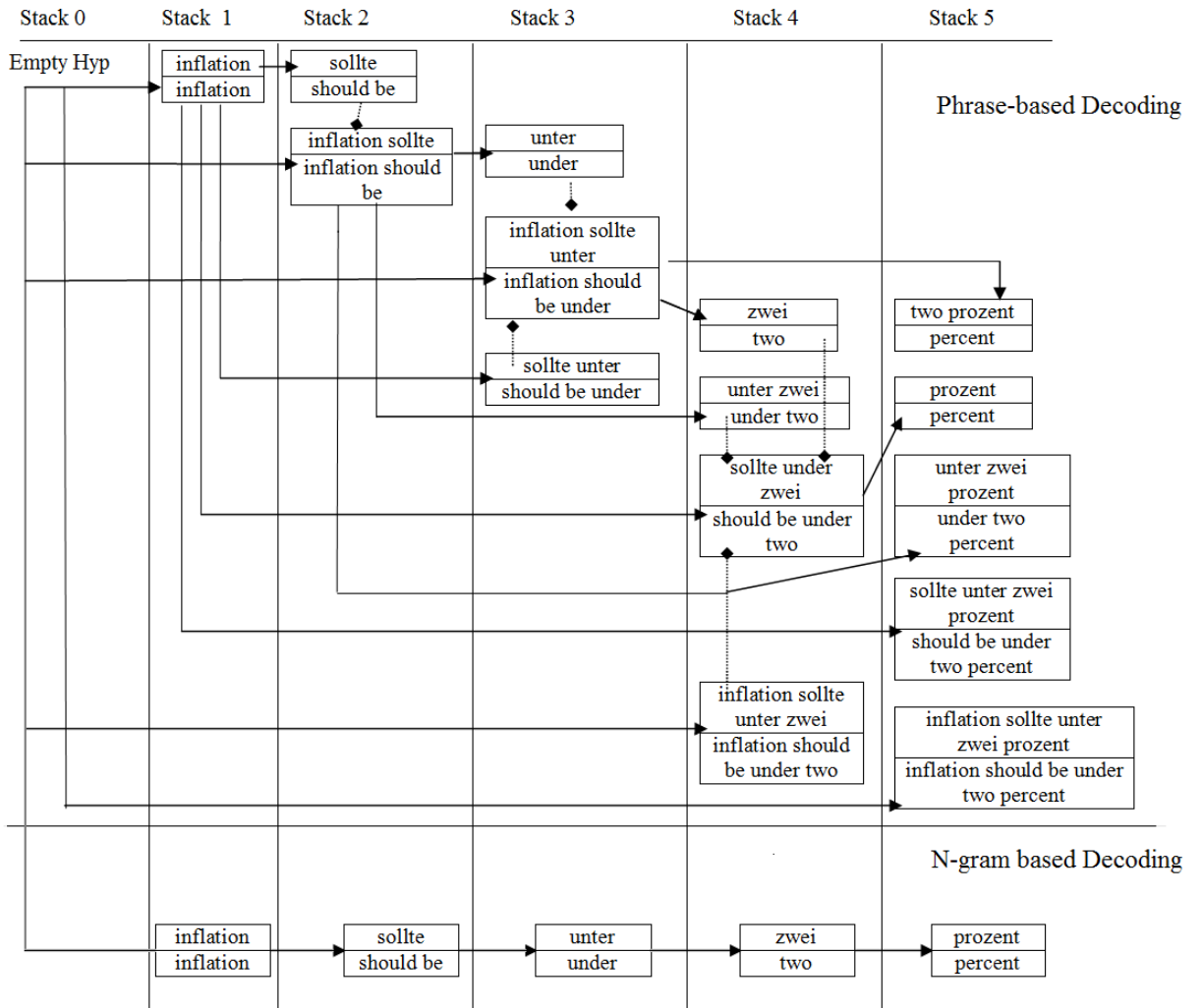


Figure 3.23: Comparison of Search - Phrase-based verses N-gram-based SMT

Using Phrases verses Tuples in Search

Using tuples during search as compared to phrases has a distinct drawback. Phrases are larger translation units that capture more dependencies. An N-gram model can capture the same dependencies by using multiple tuples. This however, has to occur in several steps during decoding resulting in a difficult search problem. Consider the German idiom “nach meine meinung - in my opinion”. The phrase-based decoder can translate it as a single phrase and place it on stack $j + 3$ where j represents the current stack. In comparison N-gram model translates this with three tuples $\langle nach - in \rangle \langle meine - my \rangle \langle meinung - opinion \rangle$. Because “in” is not a common translation of the German word “nach”, which usually translates to “after” or “to”, it will be ranked quite low in the stack until the following tuples $\langle meine - my \rangle \langle meinung - opinion \rangle$ appear in the subsequent stacks. See Figure 3.24 for an example. In order for $\langle nach - in \rangle$ to survive pruning it must appear in the N-best list of hypotheses in the stack it is placed. N-gram models might thus require a higher beam size to prevent the pruning of the hypothesis that translates “nach” to “in”. Costa-jussà et al. (2007) reports a significant drop in the performance of N-gram-based SMT when a beam size of 10 is used instead of 50 in Spanish-to-English and English-to-Spanish experiments. In comparison, translation accuracy of phrase-based SMT remains the same when varying the beam size between 5 and 50. Koehn and his colleagues have also repeatedly shown that increasing the Moses stack size from 200 to 1000 does not have a significant effect on translation into English, see (Koehn and Haddow, 2009) and other shared task papers.

Reordering Limit

Unlike Phrase-based SMT, the N-gram model does not explicitly impose a hard reordering limit i.e. allowing reordering only in a fixed window of 5-8 words. But this limit is implicitly imposed by restricting the rewrite rules to have up to 6 POS tags in the left hand side of the rules. Because each POS tag represents exactly 1 German word, N-gram model, like phrase-based

3 N-gram-based SMT

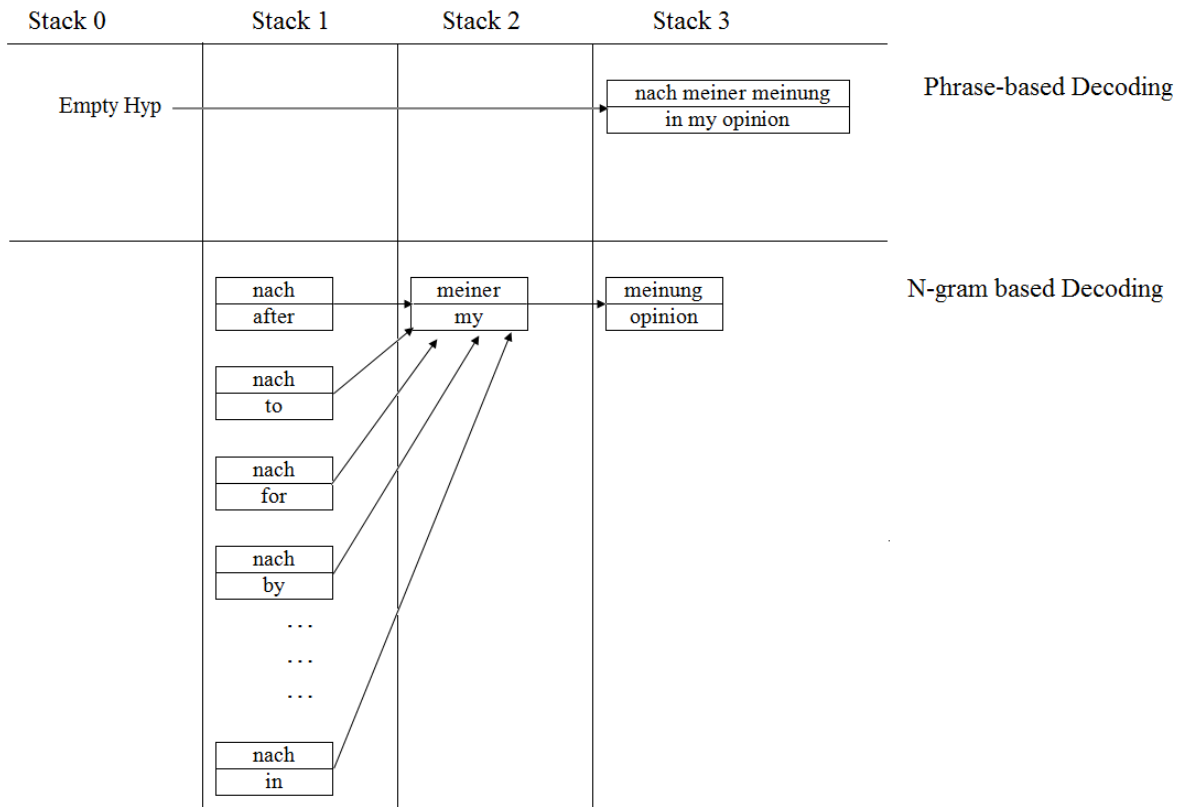


Figure 3.24: Comparison of Search - Phrases vs. Tuple Placement on Stack

3 N-gram-based SMT

74% würden gegen die Studiengebühren, 79% gegen die Praxisgebühr, und 84% gegen das Krankenhaus-Taggeld stimmen

74% would vote against the tuition fee, 79% against the clinical practice, and 84% against the hospital daily allowance

Figure 3.25: Long Distance Reordering

SMT can only do reordering within a fixed window. Using larger rewrite rules introduces sparsity problems. The N-gram model, therefore, like phrasal SMT fails to reorder “stimmen” in Figure 3.25 which requires a jump over 15 German words. See Section 2.3.4 for the discussion of hard-distortion limit in phrase-based SMT.

3.5.4 Drawbacks of N-gram-based SMT

In this section we summarize the drawbacks of the N-gram-based SMT as discussed in comparison with Phrase-based SMT in the previous section.

1. By linearizing the source, N-gram-based SMT throws away useful information about how a particular word is translated with respect to the previous word. This information is rather stored in form of rewrite rules. However, because POS tags are used instead of word forms, a rewrite rule might fail to retrieve the reordering observed during training.
2. Using split tokens to represent target-side discontinuities introduce spurious representations of a German token that aligns to the discontinuous English-side words. Moreover when the tuples representing discontinuous units are further apart than 3 tuples the dependency can not be captured.

3. Unaligned words on the English-side are handled through a post-processing heuristic that attach unaligned English words to the left or right neighboring word. One of the dependencies is lost as a result of the decision.
4. Standard *N*-gram SMT relies heavily on POS-tags. POS-based rewrite rules represent a fixed number of German words and can not generalize to long-distance reordering. Re-usability of these rules is hampered by the problem of data sparsity. If a rule fails to trigger the *N*-gram model can not hypothesize a particular phenomenon such as reordering and handling source or target-side words.
5. The *N*-gram model hypothesizes for the pre-calculated word permutations based only on the source-side words. Often the evidence of correct reordering is provided by the target-side language model. All potential reorderings that are not supported by the rewrite rules are pruned in the pre-processing step.
6. Using tuples presents a more difficult search problem than that in phrase-based SMT. A hypothesis that, if allowed to continue might turn out to produce the best translation in the final stack, might rank very low in the initial stacks and may get pruned. This is likely to occur when translating idioms and phrases where individual words do not have their literal meanings.
7. Using POS-tags imposes an implicit constraint of reordering within a window of 6 or less words preventing the *N*-gram model to hypothesize long range reordering which requires larger jumps.

3.6 Chapter Summary

In this chapter we gave a comprehensive overview of a state-of-the-art *N*-gram-based system and its component features. *N*-gram SMT is an instance of a joint model. Translations are generated as bilingual units called tuples.

3 *N-gram-based SMT*

Tuples are minimal translation units that encapsulate source and target information and are generated so that a unique and monotonic segmentation of a bilingual sentence is produced. To unfold the embedded tuples, linearization of the German-side is done. Linearization of the source also enables the N-gram model to handle local and non-local dependencies such as long distance reordering, German-side discontinuities etc. English-side discontinuities are handled using split tokens. The decoding framework of the N-gram model is based on POS-based rewrite rules, extracted during training. Only pre-calculated orderings are hypothesized during search. This leads to a compact search graph contributing towards efficient decoding. The N-gram model has several advantages over phrase-based model. Using bilingual units provides mutual context. The N-gram model does not have a spurious phrasal segmentation problem. However, using smaller translation units makes the search problem more difficult causing more search errors. A major disadvantage of the N-gram model is that it is a pre-ordered approach. Not all possible word permutations are hypothesized during search. Secondly the search graph is constructed with POS-based rewrite rules which can only capture short-distance reorderings. Both approaches have their pros and cons. In the next chapter we propose a mechanism to capture the benefits of both techniques.

4 A Joint Sequence Translation Model with Integrated Reordering

The last two chapters were spent going through the details of the state-of-the-art Phrase-based and N-gram-based SMT models. We discussed the pros and cons of each model. In this chapter we present a novel operation sequence model that tries to capitalize on the advantages of both approaches and aims at rectifying some of the mentioned weaknesses. In the first few sections we will discuss our model in terms of translation modeling (translation units and reordering framework etc), different supporting features and the decoding strategy. In the second half we perform an empirical evaluation comparing our model outputs with the state-of-the-art phrase-based system Moses¹, Phrasal (Cer et al., 2010)², a discontinuous phrase-based system and the state-of-the-art N-gram system Ncode.³

4.1 Introduction

We present a novel generative model that explains the translation process as a linear sequence of operations which generate a source and target sentence in parallel. Possible operations are (i) generation of a sequence of source and

¹<http://www.statmt.org/moses/>

²<http://nlp.stanford.edu/phrasal/>

³<http://www.limsi.fr/Individu/jmcrego/bincoder/>

target words (ii) insertion of gaps as explicit target positions for reordering operations, and (iii) forward and backward jump operations which do the actual reordering. The probability of a sequence of operations is defined according to an n -gram model, i.e. the probability of an operation depends on the $n - 1$ preceding operations. Since the translation (generation) and reordering operations are coupled in a single generative story, the reordering decisions may depend on preceding translation decisions and translation decisions may depend on preceding reordering decisions. This provides a natural reordering mechanism which is able to deal with local and long-distance reorderings in a consistent way. Our approach can be viewed as an extension of the N -gram SMT approach but N -gram SMT does reordering as a preprocessing step and not as an integral part of a generative model.

4.2 Generative Story

The generative story of the model is motivated by the complex reordering in the German-to-English translation task. Our model, like the N -gram model, is an instance of a joint source channel model. The English words are generated in linear order⁴ while the German words are generated in parallel with their English translations. Mostly the generation is done monotonically. Occasionally the translator jumps back on the German side to insert some material into a gap that was inserted earlier. Each inserted gap acts as a designated landing site for the translator to jump back. After this is done, the translator jumps forward again and continues the translation. We will now, step by step, present the characteristics of the new model by means of examples.

⁴Generating the English words in order is also what the decoder does when translating from German to English.

4.2.1 Basic Operations

The generation of the German translation **Peter liest** of the English sentence **Peter reads** is straightforward because it is a simple 1-to-1 word-based translation without reordering:

- Generate “Peter – Peter”
- Generate “liest – reads”

The translation **Peter reads – Peter liest also**, requires the *insertion* of an additional German word at the end. Conversely, the translation **Peter reads then – Peter liest** requires the *deletion* of an untranslated English word.

4.2.2 Reordering Operations

Let us now turn to an example which requires reordering. Consider the following example:

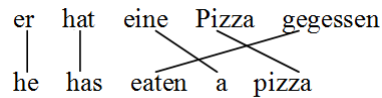


Figure 4.1: Reordering Example

The generation of this sentence in our generative story could proceed as below:

- Generate “er – he”

$$\begin{array}{c} \text{er} \downarrow \\ | \\ \text{he} \end{array}$$
- Generate “hat – has”

$$\begin{array}{cc} \text{er} & \text{hat} \downarrow \\ | & | \\ \text{he} & \text{has} \end{array}$$

The down arrow (\downarrow) represents position of the translator i.e. the position of the German word after the previously translated German word.

4 A Joint Sequence Translation Model with Integrated Reordering

The sequence is so far monotonic because the German words generated so far are in the same order as the English words. However, the next generation step requires a reordering. In order to generate the German word “gegessen”, the translator needs to jump over the intervening words “eine” and “Pizza”. A gap is first inserted on the German side, followed by the generation of “gegessen – eaten”.

- Insert Gap

er	hat	□	↓
he	has		

- Generate “gegessen - eaten”

er	hat	□	gegessen	↓
			/	
he	has	eaten		

The inserted gap acts as a place-holder for the skipped German words. When the translator requires to generate any of the skipped words it jumps back to the open gap and proceeds with the generation. After the generation of “gegessen – eaten”, the translator must generate the skipped words “eine – a” and “Pizza – pizza”. The generative story proceeds as below:

- Jump Back

er	hat	↓	□	gegessen
				/
he	has	eaten		

- Generate “eine – a”

er	hat	eine	↓	gegessen
		/	/	
he	has	eaten	a	

- Generate “Pizza – pizza”

er	hat	eine	Pizza	gegessen	↓
		/	/	/	
he	has	eaten	a	pizza	

The backward jump operation makes the translator jump to the open gap and close it. The translator then proceeds monotonically with the generation of “eine – a” and “Pizza – pizza”.

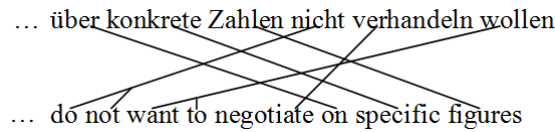


Figure 4.2: Sub-ordinate German-English Clause Pair

Complex Reorderings: Multiple gaps can exist at a single time. The translator decides based on the next English word to be covered which open gap to jump to. Figure 4.2 shows a German-English subordinate clause pair. The generation of this example is carried out as follows:

- Insert Gap □ ↓
- Generate “nicht – do not” □ nicht ↓
do not

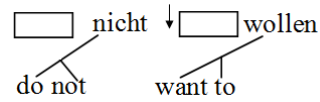
The inserted gap acts as a place holder for the German words “über konkrete Zahlen” that are skipped to be generated later. The next English cept in the order is “want to”. But first, another gap is required to be inserted before the generation of “wollen – want to”. This gap serves as the place holder for the German word “verhandeln”.

- Insert Gap □ nicht □ ↓
do not
- Generate “wollen – want to” □ nicht □ wollen ↓
do not want to

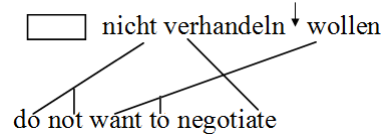
The two open gaps are the landing sites for the “backward jump” operation. When the translator decides to generate any of the skipped words it jumps back to one of the open gaps. The “backward jump” operation closes the gap that it jumps to. The translator proceeds monotonically from that point until it needs to jump again. The generation proceeds as follows:

4 A Joint Sequence Translation Model with Integrated Reordering

- Jump Back

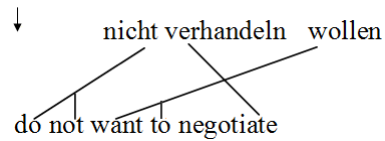


- Generate “verhandeln – negotiate”

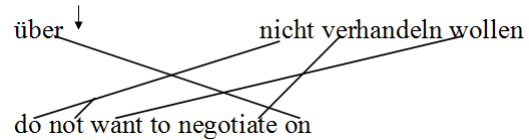


Lastly the translator jumps back to the other open gap and generates the remaining German and English words.

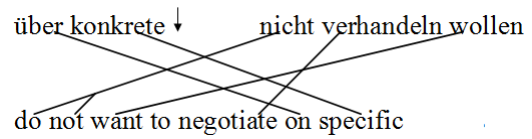
- Jump Back



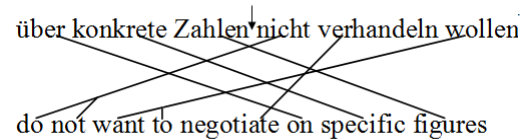
- Generate “über – on”



- Generate “konkrete – specific”



- Generate “Zahlen – figures”



Recursive Reorderings: Some reorderings require recursively inserting a gap within a gap. Consider the translation of another German-English subordinate clause pair shown in Figure 4.3. The generation requires a long distance reordering of the verb phrase “senken wird” and an internal reordering of “senken – cut” and “wird – will”. After the generation of “dass – that” “die – the” and “EZB – ECB”, the translator proceeds by inserting a gap for the skipped German words followed by the generation of “wird – will”.

4 A Joint Sequence Translation Model with Integrated Reordering

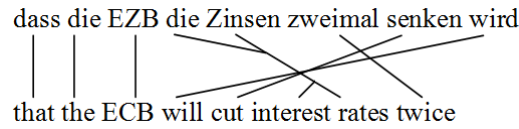
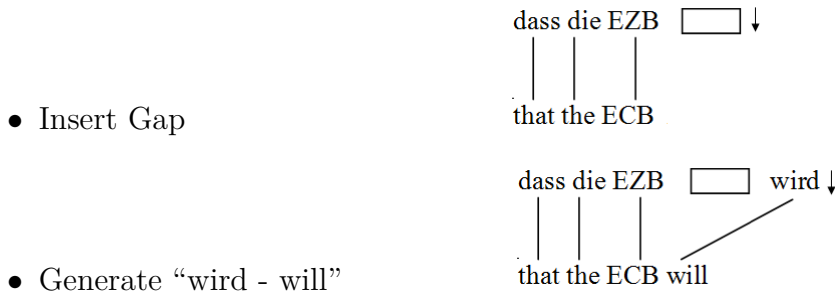
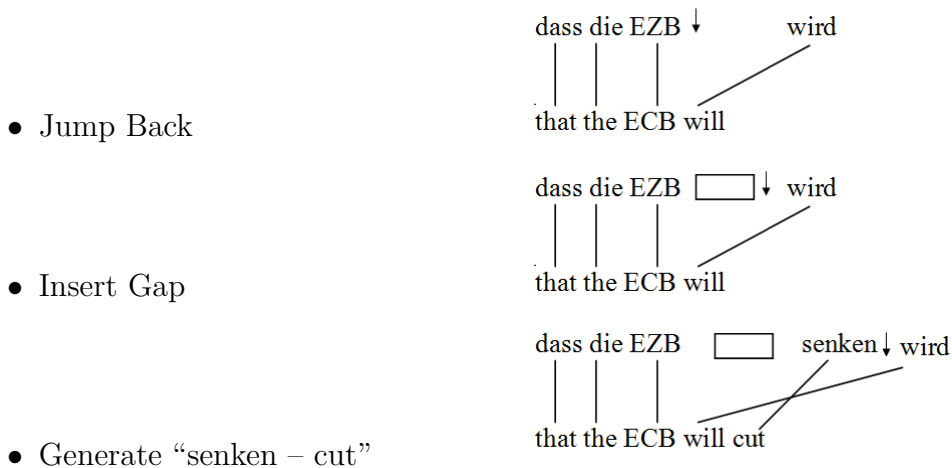


Figure 4.3: Complex Reordering Pattern in German-English Translation



The translator continues by jumping back to the open gap. But before it generates “senken – cut”, another gap has to be inserted. Recall from the discussion above that a “jump back” operation closes the gap it jumps to. Because “senken” was not the first word of the gap. A new gap has to be inserted within the closing gap for the words “die Zinsen zweimal” to be generated later. The translator then continues with the generation of “senken – cut”. After generating “senken – cut” the translator will jump back to the open gap and generate the skipped words “die Zinsen zweimal”.



4.2.3 Discontinuous Translation Units

Now we discuss how discontinuous cepts can be represented in our generative model. The insert gap operation discussed in the previous section, can also be used to generate discontinuous source cepts. The generation of any such cept is done in several steps. See the example in Figure 4.4. The generation of the gappy cept⁵ “hat...gelesen – read” can be done as follow:

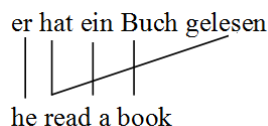
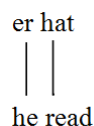
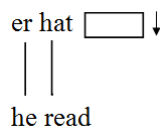


Figure 4.4: Discontinuous German-side Cept

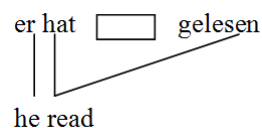
- Generate “er – he”



- Generate “hat...[gelesen] – read”



- Insert Gap



- Continue Source Cept

After the generation of “er – he”, the first part of the German complex verb “hat” is generated as an incomplete translation of “read”. The second part “gelesen” is added to a queue to be generated later. A gap is then inserted for

⁵A cept is a group of words in one language translated as a minimal unit in one specific context (Brown et al., 1993).

the skipped words “ein” and “Buch”. Lastly the second word (“gelesen”) of the unfinished German cept “hat...gelesen” is added to complete the translation of “read”.

Discontinuous cepts on the English-side can not be generated orthogonally because of the fundamental assumption of the model that English will be generated from left-to-right. This is a drawback of our approach which we will discuss in the next chapter.

4.2.4 Insertions and Deletions

Unaligned German and English words can be inserted or deleted through special operations “Generate Source Only” and “Generate Target Only”. See Figure 4.5 for example. The German and English pronouns “Sie” and “me” are unaligned. The bilingual sentence pair can be generated with the following sequence:

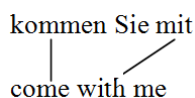


Figure 4.5: Insertion and Deletion

- Generate “kommen – come”
- Generate Source Only “Sie”
- Generate “mit – with”
- Generate Target Only “me”

4.2.5 Formal Definition of Operations

In this section we define the operation set formally. Our model uses five translation and three reordering operations which are repeatedly applied in a sequence. The following is a formal definition of each of these operations:

- **Generate (X,Y)** The source and target sentences are generated through the $\text{Generate}(X,Y)$ operation. X and Y are German and English cepts respectively, each with one or more words. Words in X (German) may be consecutive or discontinuous, but the words in Y (English) must be consecutive. This operation causes the words in Y and the first word in X to be added to the English and German strings respectively, that were generated so far. Subsequent words in X are added to a queue to be generated later. All the English words in Y are generated immediately because English is generated in linear order, for example $\text{Generate}(\text{Inflationsraten}, \text{inflation rates})$. The generation of the second (and subsequent) German word in a multi-word cept can be delayed by gaps, jumps and the Generate Source Only operation defined below.
- **Continue Source Cept:** The German words added to the queue by the $\text{Generate}(X,Y)$ operation are generated by the “Continue Source Cept operation”. Each Continue Source Cept operation removes one German word from the queue and copies it to the German string. If X contains more than one German word, say n many, then it requires n translation operations, an initial $\text{Generate}(X_1\dots X_n, Y)$ ⁶ operation and $n - 1$ Continue Source Cept operations. For example “kehrten...zurück – returned” is generated by the operation $\text{Generate}(\text{kehrten zurück}, \text{returned})$, which adds “kehrten” and “returned” to the German and English strings and “zurück” to a queue. A Continue Source Cept operation later removes “zurück” from the queue and adds it to the German string.
- **Generate Source Only (X):** The string X is added at the current position in the German string. This operation is used to generate a German word X with no corresponding English word. It is performed immediately after its preceding German word is generated. This is because there is no evidence on the English-side which indicates when to generate X. Gen-

⁶Note that the dots (...) within “ $X_1\dots X_n$ ” do not represent discontinuity in our case but only that they are two separate source words which may be adjacent or discontinuous.

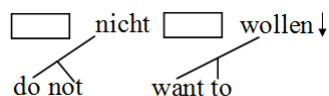
erate Source Only (X) helps us learn a source word deletion model. It is used during decoding, where a German word (X) is either translated to some English word(s) by a Generate (X,Y) operation or deleted with a Generate Source Only (X) operation.

- **Generate Target Only (Y):** The string Y is added to the current position in the English string. Because English is generated left-to-right, Generate Target Only (Y) is carried out after the $(Y - 1)^{th}$ English word. Generate Target Only (Y) operation is symmetrical to its counter part Generate Source Only (X). While it might be useful to learn a target word insertion model for the word alignment task, hypothesizing insertions during decoding is a challenging task. Both German and English sentences are available to the word alignment task. However, the English sentence is hidden to the decoder and is actually what the decoder tries to construct. Because there is no evidence in the source sentence to indicate which English words are the most probable candidates for insertion, inserting the right English words is a non-trivial task. Hypothesizing all unaligned English words observed during training, increases the decoding complexity. We will return to this challenge later and propose a way to address this problem.
- **Generate Identical:** This operation adds the same word at the current position in both the German and English strings. The Generate Identical operation is used during decoding for the translation of unknown words. The probability of this operation is estimated from singleton German words that are translated to an identical string. For example, for a tuple “Portland – Portland”, where German “Portland” was observed exactly once during training, we use a Generate Identical operation rather than Generate (Portland, Portland).

We now formally define the set of reordering operations used by the generative story. Reordering has to be performed whenever the German word to be

generated next does not immediately follow the previously generated German word.

- **Insert Gap:** This operation inserts a gap which acts as a place-holder for the skipped words. A gap represents a set of continuous German words that the translator decides to jump over and leaves to generate later during the translation. Whenever the translator decides to cover any of these skipped words it has to jump back to the open gap. There can be more than one open gap at a time.
- **Jump Back (W):** This operation lets the translator jump back to an open gap. It takes a parameter W specifying which gap to jump to. Each open gap has an id starting from 1...n such that the gap closest to the right-most German word covered so far (Z) gets the lowest id i.e. 1 and so on. Jump Back (1) jumps to the closest gap to Z , Jump Back (2) jumps to the second closest gap to Z , etc. Referring back to the example shown in Figure 4.2 (Page 133), at an intermediate step during generation the example has two open gaps:



The right-most German word so far covered is “wollen”. The gap closest to “wollen” is the one that represents the word “verhandeln”. The other gap that is on the left represents the words “über konkrete Zahlen”. If the translator decides to cover “verhandeln” it will perform a Jump Back (1) operation. If the translator decides to cover “über”, “konkrete” or “Zahlen”. It will perform a Jump Back (2) operation. The gap ids are not fixed but dynamically updated after each Insert Gap and Jump Back (W) operation because the former inserts a new gap and the latter closes one. The jump back operation can not directly jump to any of the German words represented by that gap. It always jumps to the first word represented by that gap. However, if the translator does not intend to generate

the first word of that gap it has to insert another gap. Referring back to the example shown in Figure 4.3, at an intermediate step during generation the translator decides to jump back to the open gap that represents the German words “die Zinsen zweimal senken”. Although the translator intends to generate “senken – cut”, it first needs to insert another gap representing the new set of skipped words “die Zinsen zweimal”.

- **Jump Forward:** After a translator jumps to any open gap and covers any or all of the German words represented by that gap, it may want to jump to another gap or to the right-most German word so far covered (Z). The Jump Forward operation makes the translator jump to Z . Consider the example in Figure 4.6.

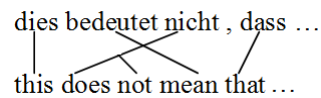
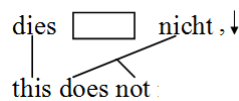


Figure 4.6: An Example to Illustrate Jump Forward Operation

The translation is carried out with the following sequence of operations:

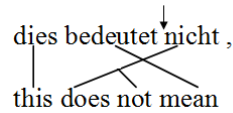
- Generate(dies,this)
- Insert Gap
- Generate(nicht, does not)
- Generate Source Only (,)

At this point, the (partial) German and English sentences look as follows:

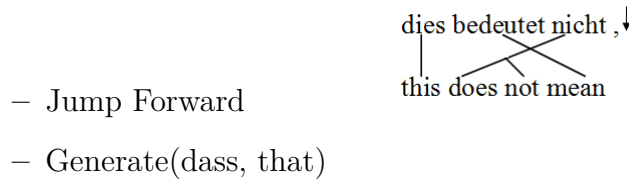


The sequence proceeds as follows:

- Jump Back(1)
- Generate(bedeutet, mean)



At this point, the (partial) German and English sentences look as follows:
 The translator now needs to jump forward to the right of the sentence Z and cover the German word “dass”. The sequence proceeds as :



The Jump Forward operation is executed when the previously covered German word (j) is not the right most German word so far covered and the translator intends to cover a German word that is to the right of the previously covered German word and is not following that word immediately. Thus in order to jump to Z or any open gap to the right of the previously covered German word (j), a Jump Forward operation has to be performed. The two scenarios are illustrated in Figure 4.7.

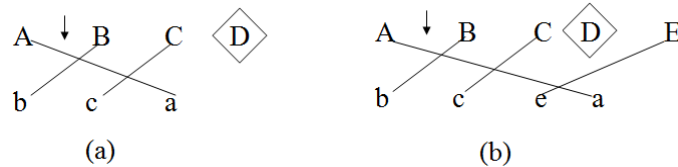


Figure 4.7: Jump Forward Scenarios

The arrow-sign \downarrow denotes the position after the previously covered German word. The next German word to be covered is shown inside a diamond-shaped box. Notice that in both cases $j < Z$ i.e. the current position of the translator is before the right-most German word so far generated and the word to be covered next is at the right of, and not

immediately following the word just covered. In the first scenario (Figure 4.7 a), the translator performs a Jump Forward to the position after “C” and generates “D”. In the second scenario (Figure 4.7 b), the translator performs a Jump Forward to the position after “E” and then executes a Jump Back(1) to generate “D”.

4.2.6 Conversion Algorithm

In this section we will present an algorithm that converts an aligned bilingual sentence pair, to a sequence of operations discussed in the previous section. Before presenting the algorithm let us sum up the basic considerations from the previous section given below. A formal algorithm for converting a word-aligned bilingual corpus into an operation sequence is presented in Algorithm 1. At the end of the section we will dry-run a sample example and show a step by step generation of a German/English sentence pair, giving the translation operations and the respective values of the index variables at each step.

- The algorithm assumes that the English cepts are composed of consecutive words and the word alignments do not contain target-side discontinuities. Removing target-side discontinuities will be discussed later in Section 4.4.
- Each translation operation generates zero or one German word and zero or one English cept. We use cept positions for English (not word positions) because English cepts are composed of consecutive words. German positions are word-based.
- The sequence of operations is linear in the sequence of English words, i.e. the translation operations for some English cept e_i precede the translation operations for another cept $e_{i'}$ if $i < i'$.
- The operation **Generate Source Only (X)** is executed immediately after its preceding German word has been covered.

Algorithm 1 Conversion Algorithm

i	Position of current English cept	F_i	Sequence of German words linked to E_i
j	Position of current German word	L_i	# of German words linked with E_i
j'	Position of next German word	k	# of already generated German words for E_i
N	Total number of English cepts	a_{ik}	Position of k^{th} German translation of E_i
f_j	German word at position j	Z	Position after right-most generated German word
E_i	English cept at position i	S	Position of the first word of a target gap W

```

 $i := 0; j := 0; k := 0$ 
while  $f_j$  is an unaligned word do
  Generate Source Only ( $f_j$ )
   $j := j + 1$ 
while  $e_i$  is an unaligned cept do
  Generate Target Only ( $e_i$ )
   $i := i + 1$ 
 $Z := j$ 

while  $i < N$  do
   $j' := a_{ik}$ 
  if  $j < j'$  then
    if  $f_j$  was not generated yet then
      Insert Gap
    if  $j = Z$  then
       $j := j'$ 
    else
      Jump Forward
  if  $j' < j$  then
    if  $j < Z$  and  $f_j$  was not generated yet then
      Insert Gap
       $W :=$  relative position of target gap
      Jump Back (W)
       $j := S(W)$ 
    if  $j < j'$  then
      Insert Gap
       $j := j'$ 
    if  $k = 0$  then
      Generate ( $F_i, E_i$ ) {or Generate Identical}
    else
      Continue Source Cept
       $j := j + 1; k := k + 1$ 
    while  $f_j$  is an unaligned word do
      Generate Source Only ( $f_j$ )
       $j := j + 1$ 
    if  $Z < j$  then
       $Z := j$ 
    if  $k = L_i$  then
       $i := i + 1; k := 0$ 
    while  $e_i$  is an unaligned cept do
      Generate Target Only ( $e_i$ )
       $i := i + 1$ 

```

4 A Joint Sequence Translation Model with Integrated Reordering

- The operation **Generate Target Only (Y)** is executed immediately after its preceding English word has been covered.
- The operation **Generate Identical** is chosen if the German and English strings are the same and the overall frequency of the German word is 1.
- Gaps and/or jumps are needed whenever the German word to be generated next does not immediately follow the previously generated German word.
- A gap is inserted when the translator decides to skip the immediately following German word(s) (which is/are not generated yet) to generate another German word.
- A forward jump is necessary if the German word to be generated next is at the right and some already generated German word is between the previously generated word and the German word to be generated next.
- A backward jump is needed if the next German word is located before the previously generated German word.
- The relative position of the gap is 1 if it is closest to the right-most German word so far covered, 2 if it is the second closest gap etc.

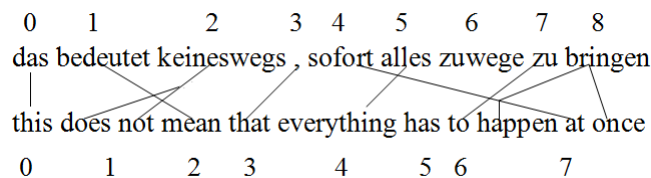


Figure 4.8: Sample Bilingual Sentence for the Dry-Run of the Conversion Algorithm – German Indexes Represent Word Positions and the English Indexes Represent Cept Positions

4 A Joint Sequence Translation Model with Integrated Reordering

Operation	Generation	States
Start		$j = 0 ; j' = 0$ $i = 0 ; Z = 0$
Generate(das, this)	das ↓ this	$j = 1 ; j' = 2$ $i = 1 ; Z = 1$
Insert Gap	das <input type="checkbox"/> ↓ this	$j = 2 ; j' = 2$ $i = 1 ; Z = 1$
Generate(keineswegs, does not)	das <input type="checkbox"/> keineswegs ↓ this does not	$j = 3 ; j' = 1$ $i = 2 ; Z = 3$
Jump Back(1)	das ↓ keineswegs this does not mean	$j = 1 ; j' = 1$ $i = 2 ; Z = 3$
Generate(bedeutet, mean)	das bedeutet keineswegs ↓ this does not mean	$j = 2 ; j' = 3$ $i = 3 ; Z = 3$
Jump Forward	das bedeutet keineswegs ↓ this does not mean	$j = 3 ; j' = 3$ $i = 3 ; Z = 3$
Generate(, , that)	bedeutet keineswegs , ↓ does not mean that	$j = 4 ; j' = 5$ $i = 3 ; Z = 4$
Insert Gap	bedeutet keineswegs , <input type="checkbox"/> ↓ does not mean that	$j = 5 ; j' = 5$ $i = 3 ; Z = 4$
Generate(alles , everything)	bedeutet keineswegs , <input type="checkbox"/> alles ↓ does not mean that everything	$j = 6 ; j' = 6$ $i = 5 ; Z = 6$
Continued on Next Page...		

Generate Source Only(zuwege)		$j = 7 ; j' = 7$ $i = 5 ; Z = 7$
Generate Target Only (has)		$j = 7 ; j' = 7$ $i = 6 ; Z = 7$
Generate(zu , to)		$j = 8 ; j' = 4$ $i = 7 ; Z = 8 ; k = 0$
Jump Back(1)		$j = 4 ; j' = 4$ $i = 7 ; Z = 8 ; k = 0$
Generate(sofort bringen, happen at once)		$j = 5 ; j' = 8$ $i = 8 ; Z = 8 ; k = 1$
Jump Forward		$j = 8 ; j' = 8$ $i = 8 ; Z = 8 ; k = 1$
Continue Source Cept		$j = 9 ;$ $i = 8 ; Z = 9$

Table 4.1: Step-wise Generation of Example 4.8. The arrow indicates position j

4.2.7 Algorithmic Complexity

The complexity of the corpus conversion algorithm is $O(NM)$, where N = number of English cepts and M = number of open gaps. The generation of English cepts can be done in an $O(N)$ time. But each time the translator jumps

back to an open gap, finding the relative position of the gap with respect to Z requires $O(M)$ time.

4.3 Model

Our model is estimated from a sequence of operations obtained through the transformation of a bilingual sentence pair. An operation can be to generate source and target words or to perform reordering by inserting gaps and through forward and backward jumps. Let $O = o_1, \dots, o_{j-1}$ be a sequence of operations as hypothesized by the translator to generate the bilingual sentence pair $\langle F, E \rangle$, the translation model is defined as:

$$p(F, E, A) = \prod_{j=1}^J p(o_j | o_{j-n+1} \dots o_{j-1})$$

where n indicates the amount of context used, A defines the alignment function between E and F . Our translation model is implemented as an N-gram model of operations using SRILM Toolkit (Stolcke, 2002) with Kneser-Ney smoothing. The translate operations (Generate(*)) encapsulate tuples. Tuples are minimal translation units extracted from the M-N word aligned corpus such that, given the alignment, each bilingual sentence pair has a unique segmentation and each tuple is an atomic unit. The idea is similar to N-gram-based SMT except that the tuples in the N-gram model are generated monotonically. The bilingual corpus is monotonized by linearizing the source according to the target. We do not impose the restriction of monotonicity in our model but integrate reordering operations inside a generative model. A detailed comparison with the N-gram and Phrase-based models follows later.

4.3.1 Search

Given a source string F , a sequence of tuples $T = (t_1, \dots, t_n)$ as hypothesized by the decoder to generate a target string E , the search problem can be defined as:

$$\hat{E}, \hat{A} = \operatorname{argmax}_{E,A} p(F, E, A)$$

The computation of $p(E, F)$ requires summation over all possible alignments A which is not feasible. Therefore we assume a Viterbi approximation, that the probability of the most probable alignment of E and F is close to the sum of the probabilities of all possible alignments of E and F .

Because the English data for training the monolingual language model is available in a lot more quantity the search problem can benefit from it. Integrating the language model the search is defined as:

$$\hat{E}, \hat{A} = \operatorname{argmax}_{E,A} p_{LM}(E)p(F, E, A)$$

where $p_{LM}(E)$ is the target-side monolingual language model and $p(F, E, A)$ is the translation model. But in this model E would be generated twice. To negate this effect we divide by the marginal $\sum_{F',A'} p(F', E, A')$.

$$\hat{E}, \hat{A} = \operatorname{argmax}_{E,A} p_{LM}(E) \frac{p(F, E, A)}{\sum_{F',A'} p(F', E, A')}$$

The marginal is approximated by $p_{pr}(E)$, a monolingual English ngram model whose parameters are estimated on the English part of the bilingual corpus. The search can then be defined as:

$$\hat{E}, \hat{A} = \operatorname{argmax}_{E,A} p_{LM}(E) \frac{p(F, E, A)}{p_{pr}(E)}$$

Both, the monolingual language and the prior probability model are implemented as standard word-based n-gram models:

$$p_x(E) \approx \prod_{j=1}^J p(w_j | w_{j-m+1}, \dots, w_{j-1})$$

where m indicates the amount of context used. We use a 5 gram for the monolingual language model and a 9 gram for the operation language model. The prior probability model is estimated from the English-side of the bilingual data.

In decoding, the amount of context used for the prior probability is synchronized with the position of back-off in the operation model. For example generating the bilingual sentence pair in Figure 4.8, say the decoder hypothesizes a translation tuple “sofort bringen – happen at once”. The operation history is:

“Generate(das,this) → Insert Gap → Generate(keineswegs, does not) → Jump Back(1) → Generate(bedeutet,mean) → Jump Forward → Generate(, , that) → Insert Gap → Generate(alles , everything) → Generate Source Only(zuwege) → Generate Target Only (has) → Generate(zu , to)

Let us assume that the operation model backs-off at the operation Generate(alles,everything). The context used for the prior probability model, in this case, will be “everything has to”. If the operation model backs-off the operation Generate(zu,to) then a bigram model is used for the prior probability model and the look-up history will only have the English word “to”. The idea of the prior probability model is to negate the effect of E generated in $P(F, E, A)$.

Using a prior probability model makes our formulation similar to the Noisy Channel model which tries to maximize the translation probability as:

$$p(E|F) = \operatorname{argmax}_E p(F|E)p_{LM}(E)$$

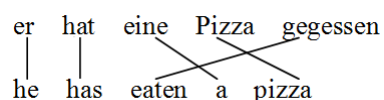
Substituting $p(F|E)$ with $\frac{p(F,E)}{p(E)}$ gives the search problem as used by our model.

4.3.2 Discussion

In this section we will enumerate some useful properties of our model.

- **Integrating Translation and Reordering:** The model provides a novel formulation that integrates translation and reordering into a single operation model. Since the translation (generation) and reordering operations are coupled in a single generative story, the reordering decisions may depend on preceding translation decisions and translation decisions

may depend on preceding reordering decisions. This provides a natural reordering mechanism. The placement of gaps is conditioned on the local context and therefore quite restrictive. Similarly, the backward jump operation is also conditioned on the local context. Recall the movement of “gegessen” in the following example:



Which operation is likely after the initial generation of the translation pair “hat – has”? In the English sentence, the verb is likely to follow, whereas in the German sentence we expect a noun phrase as a probable sequence of words. Consequently, the probability of a gap insertion operation after the generation of the auxiliaries “hat – has” will be high because reordering is necessary in order to move the second part of the German verb complex (“gegessen”) to its correct position at the end of the clause. After inserting the gap and generating the translation pair “gegessen – eaten”, we are likely to see a noun phrase (an eatable object) in the English sentence. Therefore a probable next operation is a backward jump to the gap position in the German sentence.

- **Learning Lexical Triggers:** Our model has the ability to learn lexical triggers that it can apply to the unseen patterns. The generation of the above sentence in our model starts with generating “er – he”, “hat – has”. Then a gap is inserted on the German side, followed by the generation of “gegessen – eaten”. At this point, the (partial) German and English sentences look as follows:

er hat gegessen

he has eaten

The translator then jumps back on the German side and fills it by generating “eine – a” and “Pizza – pizza”. A reordering pattern is learned by memorizing the operation sequence:

“Generate(hat, has) → Insert Gap → Generate(gegessen, eaten) → Jump Back(1)”

The learned pattern can be successfully applied to the German sentence “sie hat eine menge butterkekse gegessen – she has eaten a lot of butter cookies” or any other German sentences having a construction such as “hat X gegessen – has eaten X”, where X is a non-terminal category (represented by a gap in our model), a placeholder for the words to be filled in further translation steps.

Multiple open gaps exhibit a behavior similar to that of having multiple non-terminals in a hierarchical or discontinuous phrase-based system. Recall the example shown below:

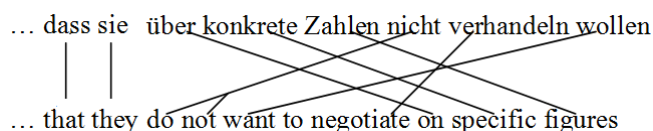
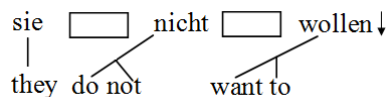


Figure 4.9: Sub-ordinate German-English Clause Pair

From the generation of the following example:

“ ... Generate(sie, they) → Insert Gap → Generate(nicht,do not) → Insert Gap → Generate(wollen, want to) → Jump Back(1) ...”

the model learns a pattern that will be useful for translating any construction such as “sie X₁ nicht X₂ wollen – they do not want X₁ X₂.”



- **Handling of Local and Non-local Reorderings:** The operation model gets away from the “shot gun” reordering model by restricting jumps

to gap positions and is able to represent dependencies which cross the “phrase boundaries” of phrase-based SMT. Both local and long distance dependencies are handled in a unified manner. Consider the sentence pair shown in Figure 4.10 (a).

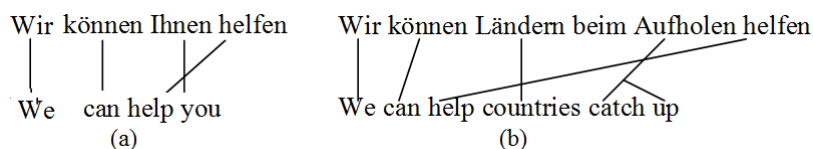


Figure 4.10: Handling Dependencies (a) Local (b) Non-local

The generation of the above sentence in our model starts with generating “Wir – We”, “können – can”. Then a gap is inserted on the German side, followed by the generation of “helfen – help”. At this point, the (partial) German and English sentences look as follows:

Wir können helfen

We can help

The translator then jumps back on the German side and fills the gap by generating “Ihnen – you”, for the first example and generating “Ländern – countries”, “beim – null” and “Aufholen – catch up” for the second example, thus handling both short and long distance reordering in a unified manner.

- **Discontinuous Source-Side Units:** With the help of “Insert Gap” and “Continue Source Cept” operations our model can learn discontinuous source-side units like “hat ... gelesen – read” and “kehrten...zurück – returned”. Our model does not handle target-side discontinuous units. But we will give a heuristic (discussed in Section 4.4) to deal with such units.

- **Minimal Translation Units:** Our model, like the N-gram model, uses only minimal translation units.⁷ Using minimal units prevents our model from having the spurious phrasal segmentation problem. However, it presents a more difficult challenge in search than the N-gram model faces. We will discuss this in detail in Section 4.5.
- **Memorizing Phrases:** Although it is based on minimal translation units, our model has the ability to learn larger phrasal units. Through the operation sequences, the model not only memorizes reorderings local to a phrase but also the non-local inter phrase reorderings. Recall the example in Figure 4.2 (Page 133) and its generation. The model memorizes the verb phrase “nicht verhandeln wollen – do not want to negotiate” through the operation sequence:

“Generate (nicht , do not) → Insert Gap → Generate (wollen , want to) → Jump Back(1) → Generate (verhandeln , negotiate)”

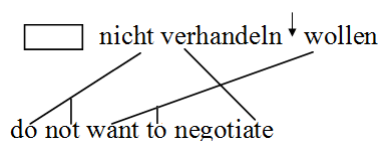
and a prepositional phrase “über konkrete Zahlen – on specific figures” through the operation sequence:

“Generate(über, on) → Generate(konkrete, specific) → Generate(Zahlen, figures)”

It also memorizes that the prepositional phrase gets swapped with the verb phrase. This is done in two steps. In the first step the translator inserts a gap and skips the generation of the prepositional phrase “über konkrete Zahlen – on specific figures” and continues with the generation of the verb phrase “über konkrete Zahlen – on specific figures” through the above sequence. At this point, the (partial) German and English sentence look as follow:

The translator then jumps back and generates the prepositional phrase to complete the swap.

⁷M-N word alignments obtained from the symmetrization of GIZA++ alignments



- **Bilingual Context:** Our model, like the N-gram model, takes into account bilingual context when generating translations. The model has access to n preceding operations, each encapsulating translation or reordering information on how previous German words were translated or reordered.
- **Insertions and Deletions:preposition phrase** Our model can learn insertions and deletions in context just like the phrase-based system through the “Generate Source Only(X)” and “Generate Target Only(Y)” operations. The decoder can hypothesize a translation unit such as “Sie – null” when translating “kommen Sie mit”, through “Generate Source Only (Sie)” operation. Although the generative story has the “Generate Target Only(Y)” operation and the operation model can score good versus bad insertions with the help of context, selecting the most probable candidates for insertion during decoding is a non-trivial problem. Our model shares this drawback with the N-gram model. In order to remedy this problem a heuristic is applied to remove the unaligned target-words from the word alignments. See Section 4.4 for details.

4.3.3 Discriminative Modeling

In order to improve end-to-end accuracy, we introduce new features for our model and shift from the generative model to the standard log-linear approach (Och and Ney, 2004) to tune⁸ these. We search for a target string E which

⁸We tune the operation, monolingual and prior probability models as separate features. We expect the prior probability model to get a negative weight but we do not force MERT

maximizes a linear combination of feature functions:

$$\hat{E} = \arg \max_E \left\{ \sum_{j=1}^J \lambda_j h_j(F, E) \right\}$$

where λ_j is the weight associated with the feature $h_j(F, E)$. Other than the 3 features discussed above (log probabilities of the operation model, monolingual language model and prior probability model), we train 9 additional features discussed below:

Length Bonus

During decoding when hypotheses compete, a hypothesis having lesser number of target words is favored by the language model because less n-grams have to be scored. In order to counter this effect and compensate for the model's bias towards producing shorter sentences, a length bonus feature (also called length penalty or word bonus/penalty) is used by many SMT systems. The length bonus feature counts the length of the target sentence in words:

$$h_{lb}(F, E) = h_{lb}(E) = T$$

where T is the number of target words hypothesized by the decoder to produce the output sentence E .

By adjusting the feature weight λ_{lm} of the length bonus feature we can make the system produce shorter or longer sentences. If λ_{lm} is negative, the system will produce shorter output sentences and vice versa.

Deletion Penalty

Another feature for avoiding too short translations is the deletion penalty. Deleting a source word (Generate Source Only (X)) is a common operation in

to assign a negative weight to this feature.

the generative story. Because there is no corresponding target-side word, the monolingual language model score tends to favor this operation. The deletion penalty counts the number of deleted source words:

$$h_{dp}(F, E) = h_{dp}(F) = S$$

where S is the number of source words deleted by the decoder to produce the output sentence E .

Gap Bonus and Open Gap Penalty

These features are introduced to guide the reordering decisions. We observe a large amount of reordering in the automatically word aligned training text. However, given only the source sentence (and little world knowledge), it is not realistic to try to model the reasons for all of this reordering. Therefore we can use a more robust model that reorders less than humans. The gap bonus feature sums to the total number of gaps inserted to produce a target sentence:

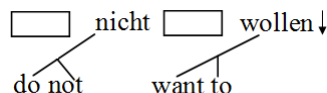
$$h_{gp}(F, E) = G$$

where G is the number of gaps inserted by the decoder to produce the output sentence E .

The open gap penalty feature is a penalty (paid once for each translation operation (Generate, Generate Identical, Generate Source Only) performed) whose value is the number of open gaps. Each generate operation translates one German word at a time. The model penalizes a hypothesis for any open gaps every time it decides to generate a source word instead of closing an open gap and generating the skipped source words. This penalty controls how quickly gaps are closed. Given a source sentence F and a sequence of tuples $T = (t_1, \dots, t_J)$ as hypothesized by the decoder to produce a target sentence E , let q_j be the number of open gaps when translating the tuple t_j , the open gap feature is estimated as:

$$h_{ogp}(E, F) = \sum_{j=1}^J q_j$$

Recall the example 4.9. At an intermediate step the partial German and English sentences are:



The open gap penalty paid by the hypothesis after generating the tuple “wollen – want” is 2. In the next step if the decoder chooses not to jump back and generate another source word excluding the ones represented by the gaps, it will have to pay the open gap penalty of 2 again. If the decoder decides to jump back and cover the skipped words, it will still pay an open gap penalty of 1 for the second open gap in all the subsequent generations until there are no more open gaps. The overall idea is to close any open gaps quickly unless the other features provide strong evidence for a different ordering.

Distance-based Penalties

Reordering Distance Penalty: We apply three additional features to control the reordering decisions. One of these is similar to the distance-based reordering model used by phrase-based SMT. Let x_1, \dots, x_n and y_1, \dots, y_m represent indexes of the source words covered by the tuples t_j and t_{j-1} respectively. The distance between t_j and t_{j-1} is given as:

$$d_j = \min(|x_k - y_l| - 1) \text{ where } x_1 \leq x_k \leq x_n, y_1 \leq y_l \leq y_m$$

The accumulative reordering penalty is given as:

$$h_{rdp}(E, F) = \sum_{j=1}^J d_j$$

This is different than the phrase-based system which calculates the distance as $d = |x_1 - y_m - 1|$, where $x_1 =$ index of the current source phrase and

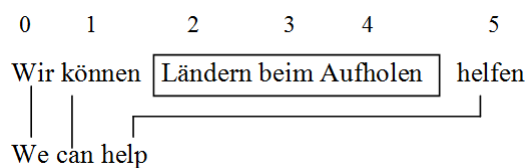


Figure 4.11: An Example to Demonstrate Gap Distance Penalty

$y_m - 1$ = index of the last word of the previous source phrase. In our formulation monotonic and swap reorderings both get a reordering distance of 0. Say phrase p_0 covers source words $\{4\ 5\}$ and phrase p_1 covers source words $\{2\ 3\}$. The reordering distance according to PBSMT is $d = |2 - 5 - 1| = 4$. The reordering distance according to our formulation is $|3 - 4| - 1 = 0$. The intuition is to penalize left and right jumps equally. This subtle difference leads to minor performance gains in some of our experiments.

Gap Distance Penalty: The other feature is the gap distance penalty which calculates the distance between the index of the first word of a source cept x and the index of the first word of the left-most gap. This cost is paid once for each Generate, Generate Identical and Generate Source Only. For a source cept covered by the indexes x_1, \dots, x_n , we get the feature value $g_j = x_1 - S$, where S is the index of the left-most source word where a gap starts. See Figure 4.11 for example. The decoder decides to cover the German word “helfen” (at index 5) by inserting a gap that represents the words “Ländern beim Aufholen”. The first German word represented by the gap is “Ländern” (at index 2). A gap distance penalty $5 - 2 = 3$ will be paid. Again the purpose of this feature is to close any previously opened gaps quickly. It penalizes any long distance jumps maintaining that the decoder should only reorder if there is sufficient evidence from other features (operation and monolingual language models). The accumulative gap distance penalty is defined as:

4 A Joint Sequence Translation Model with Integrated Reordering

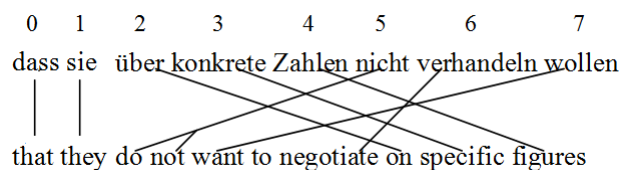
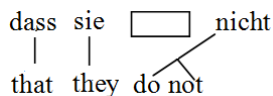


Figure 4.12: An Example to Demonstrate Relation Between Reordering and Gap Distance Penalties

$$h_{gdp}(E, F) = \sum_{j=1}^J g_j$$

The gap distance feature correlates with the previously discussed reordering penalty feature in some cases but there is an inverse relationship in other scenarios. Recall the generation of the example in Figure 4.12. After covering the tuple “sie – they”, the decoder inserts a gap for the skipped words “über konkrete Zahlen” to generate the tuple “nicht – do not”. Both the reorder-



ing and gap distance penalties operate in the same direction and penalize the jump. However, after the generation of “nicht – do not” the reordering penalty prefers the monotonic path i.e. the continuing with the German verb “verhandeln”. On the other hand the gap distance penalty prefers a “Jump Back(1)” operation to cover the skipped German word “über”. The hypothesis that covers “verhandeln” pays a gap distance penalty of $6 - 2 = 4$ but a reordering penalty of 0. The hypothesis that covers “über” pays a reordering penalty of $|2 - 6| - 1 = 3$ and no gap distance penalty.

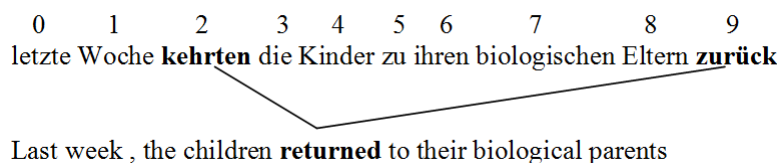


Figure 4.13: An Example to Demonstrate Gap Width Penalty

Gap Width Penalty: Another distance-based penalty used in our model is the gap width penalty. This feature only applies in the case of a gappy translation unit such as “kehrten...zurück – returned” in Figure 4.13. The value of this feature w_j is the width of the gap which is the same as the number of source words separating the words in the gappy unit. There are six German words separating the gappy unit “kehrten...zurück”. The decoder pays a gap-width penalty of 6 when it hypothesize the translation unit “kehrten...zurück – returned” for this example.

Formally let $f = f_1 \dots, f_i, \dots, f_n$ be a gappy source cept where x_i is the index of the i^{th} source word in the cept f . The value of the gap-width penalty is calculated as:

$$w_j = \sum_{i=2}^n x_i - x_{i-1} - 1$$

The accumulative gap width penalty is defined as:

$$h_{gwp}(E, F) = \sum_{j=1}^J w_j$$

Using tuples with source gaps increases the list of extracted n-best translation tuples multiple times, making the search problem more difficult. The purpose of this feature is to help the decoder prune out the tuples having source gaps, unless other features provide supporting evidence.

Lexical Features

We also use source-to-target $p_w(e|f)$ and target-to-source $p_w(f|e)$ lexical translation probabilities. Our lexical features are standard (Koehn et al., 2003). The estimation is motivated by IBM Model-1. Given a tuple t_i with source words $f = f_1, f_2, \dots, f_n$, target words $e = e_1, e_2, \dots, e_m$ and an alignment a between the source word positions $x = 1, \dots, n$ and the target word positions $y = 1, \dots, m$, the lexical feature $p_w(f|e)$ is computed as follows:

$$p_w(f|e, a) = \prod_{x=1}^n \frac{1}{|\{y : (x, y) \in a\}|} \sum_{\forall (x,y) \in a} w(f_x|e_y)$$

The lexical translation probabilities $w(f_x|e_y)$ are computed from the word-alignments. The lexical probability feature $p_w(f|e, a)$ for the example in Figure 4.14 can be computed as follow:

$$p_w(\text{mit allem}|\text{with all}) = \frac{1}{2}(w(\text{mit}|\text{with}) + w(\text{mit}|\text{all})) \times w(\text{allem}|\text{all})$$

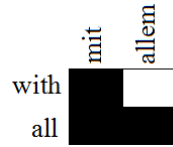


Figure 4.14: An Example to Demonstrate Lexical Probability Feature

$p_w(e|f, a)$ is computed in the same way:

$$p_w(e|f, a) = \prod_{y=1}^m \frac{1}{|\{x : (x, y) \in a\}|} \sum_{\forall (x,y) \in a} w(e_y|f_x)$$

The lexical probability feature $p_w(e|f, a)$ for the example in Figure 4.14 can be computed as follow:

$$p_w(\textit{with all}|\textit{mit allem}) = w(\textit{with}|\textit{mit}) \times \frac{1}{2}(w(\textit{all}|\textit{mit}) + w(\textit{all}|\textit{allem}))$$

When there are multiple alignments a for a tuple, we calculate $p_w(e|f)$ and $p_w(f|e)$ for each of these alignments and the average of two. Then we choose the alignment that gives the best average probability.

4.4 Word Alignments - Post-processing Heuristic

Now we will discuss two important issues that we have ignored so far:

- Target-Side Discontinuities
- Unaligned Target Words

4.4.1 Removing Target-Side Discontinuities

Problem: Our generative story does not handle target-side discontinuities. A principled drawback of the generative story is the assumption that the English sentence is generated from left to right linearly. The sequence of operations is linear in the sequence of English words, i.e. the translation operations for some English cept e_i precede the translation operations for another cept $e_{i'}$ if $i < i'$. The same is not true on the German-side where the words can be generated in any order. This assumption leads to a problem discussed below.

A discontinuous unit can be generated in two steps. The first step generates the tuple with an operation “Generate (X,Y)”, where “Y” is a discontinuous English cept such as “poured...down” in Figure 4.15. The second part of the discontinuous cept “down” is stored in a queue to be generated later. The operation “Continue Target Cept” generates the target word “down” of the cept “poured...down”.

This mechanism is analogous to that of handling source-side discontinuous units. However, the difference is that the target-side can only be generated

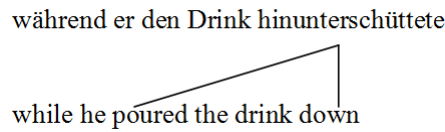


Figure 4.15: Target Side Discontinuities

from left-to-right in order. This implies that for the example in Figure 4.15, the “Continue Target Cept” operation executes after the intervening operations “Generate(den, the) → Generate(Drink, drink) →”. This makes the conditioning of the “Continue Target Cept” on the “Generate (hinunterschüttete, poured)” weak.

Secondly, in case of multiple discontinuous target-side cepts, the operation model does not know which “Continue Target Cept” operation corresponds to which “Generate(X, Y)” operation. The generation sequence for both Figure 4.16 (a) & (b) is:

“Generate(A, a₁ a₂) → Generate(B, b) → Generate (C, c₁ c₂) → Generate(D, d) → Continue Target Cept → Continue Target Cept”

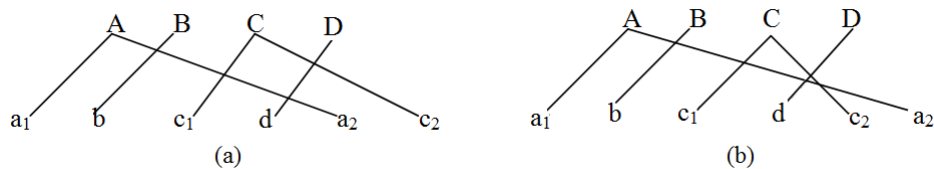


Figure 4.16: Multiple Target Side Discontinuities

We do not have the same problem with the source-side discontinuous units because the “Continue Source Cept” operation follows immediately or after an “Insert Gap” operation. The conditioning on the Generate(X,Y) operation is strong and there is no ambiguity in the case of multiple source discontinuous

cepts. Each “Continue Source Cept” operation follows immediately after its “Generate(X,Y)” counterpart.

Heuristic: In this section we discuss a method to eliminate target-side discontinuities from the training corpus. If a source word is aligned with multiple target words which are not consecutive, first the link to the least frequent target word is identified, and the group of links containing this word is retained while the others are deleted. The intuition here is to keep the alignments containing content words (which are less frequent than functional words). For the example in Figure 4.15, the alignment link between “hinunterschüttete – down” is deleted and only the link “hinunterschüttete – poured” is retained because “down” occurs more frequently than “poured”.

After applying this heuristic we will have new unaligned English words, along with the English words that were unaligned in the initial alignments. In the next section we discuss how to deal with these unaligned English words.

4.4.2 Removing Unaligned Target Words

Problem: Although the generative story provides an operation to generate the unaligned target words (Generate Target Only(Y)), inserting target-side words spuriously during decoding is a non-trivial problem. This problem has been addressed by using epsilon arcs (Knight and Al-Onaizan, 1998; Bangalore and Ricciardi, 2000), however, the decoding becomes increasingly complex (Mariño et al., 2006). N-gram-based SMT removes unaligned target words by attaching them with the preceding or the following tuple. A literature review on dealing with attaching unaligned words has been covered in the previous chapter (See Section 3.1.4 on Page 92).

Heuristic: Here we propose another heuristic for the attachment of unaligned target words. For each unaligned target word, we determine the (left or right) neighbor that it appears more frequently with and align it with the same source word as the neighbor. This is done through a simple counting procedure over

the training corpus to determine attachment preference of a word, the intuition is that words attach to words they frequently appear next to. See Algorithm 2 for the counting procedure to learn the attachment preference of the target-side words.

Algorithm 2 Learning Attachment Preferences of Target-Side Words

```

n = number of target words
i = target word at index x
j = source word aligned with i
for x = 1 to n do
  if i is unaligned then
    count(i - 1, i) ← count(i - 1, i) + 0.5
    count(i, i + 1) ← count(i, i + 1) + 0.5
  else if i - 1, i and i + 1 aligned to j then
    count(i - 1, i) ← count(i - 1, i) + 0.5
    count(i, i + 1) ← count(i, i + 1) + 0.5
  else if i - 1, i aligned to j then
    count(i - 1, i) ← count(i - 1, i) + 1
  else if i, i + 1 aligned to j then
    count(i, i + 1) ← count(i, i + 1) + 1
  
```

For a target word i in a bilingual sentence pair, if i is unaligned, we add a count 0.5 for the word pairs $(i - 1, i)$ and $(i, i + 1)$. If i , its preceding word $(i - 1)$ and the following word $(i + 1)$ are aligned with the source word j , we add a count 0.5 for the word pair $(i - 1, i)$ and $(i, i + 1)$. In the above two scenarios there is an uncertainty of whether to connect i to its left or to the right. We therefore give a partial count for i attaching to the right of $i - 1$ and to the left of $i + 1$. If there is no right or left word i.e. i is the first or the last word respectively then a count of 1 is given instead.

If i and its preceding word are aligned to a source word j , we add a count of 1.0 to the pair $(i - 1, i)$. If i and its following word are aligned to source word j , we add a count of 1.0 to the pair $(i, i + 1)$. In the above two scenarios we are certain that i either connects to the right or to the left so we give a count of 1. Figure 4.17 shows an example of unaligned target-words and the learned counts using Algorithm 2. Notice that $\text{count}(\mathbf{be}, \mathbf{just})$ and $\text{count}(\mathbf{be}, \mathbf{just})$ do

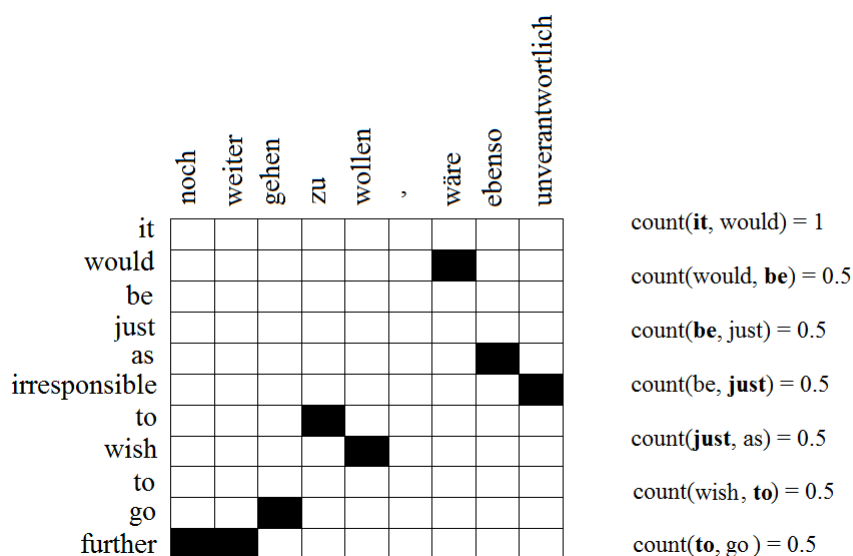


Figure 4.17: Learning Attachment Counts from Word Alignments

not mean the same and their counts do not sum up. The former maintains the information about the word “be” while the latter keeps a record for the word “just”. Learning $\text{count}(\mathbf{be}, \text{just})$ does not imply the learning of $\text{count}(\text{be}, \mathbf{just})$. In a scenario where “just” is aligned to some German word, the algorithm will not add to the $\text{count}(\text{be}, \mathbf{just})$.

Based on the counts collected by Algorithm 2, we can attach any unaligned target word to its left or right depending upon which neighboring word it is mostly observed with. The decision is made for each group of consecutive NULL words. If the leftmost NULL word is at position i , and the rightmost NULL word is at position i' then we attach to the left if $\text{count}(i-1, \mathbf{i}) \geq \text{count}(\mathbf{i}', i'+1)$ otherwise we attach to the right.

The overall justification of the word alignment post-processing heuristics is that high frequency words are more likely to carry out some function. We know that the function words in English can often be safely attached to the nearby

words. Low frequency words are more likely to be informative. The idea is analogous to the ideas of “non-head” and “head” in LEAF (Fraser and Marcu, 2007). In LEAF a cept is formed of exactly one head, and zero or more non-heads. A lot of the generative story depends only on the heads of the source and target cepts, rather than the full cepts.

4.4.3 Algorithmic Complexity

In this section we discuss the algorithmic complexity of different components in the post-processing heuristic:

- The complexity of removing the target-side discontinuities from the alignments is $O(N \log M)$, where N = number of discontinuous target-side cepts and M = size of the target-side vocabulary.
- The complexity of learning counts from the bilingual corpus is $O(N \log M)$ where N = number of English words in a sentence, M = size of the map that maintains the counts.
- The right-or-left attachment heuristic also requires $O(N \log M)$ time, where N = number of unaligned English words in a sentence and M = size of the map that maintains the counts.

Using hash tables for storing tuples, reduces the (amortized) complexity of a look up to be constant. The overall complexity of these components can then be reduced to $O(N)$.

4.5 Decoding

We implemented a custom-made decoder. In this section we will discuss the details of its different components. Our decoder performs a stack-based search with a beam-search algorithm similar to that used in Pharaoh (Koehn, 2004a). Hypotheses are arranged in stacks, such that each stack i stores hypotheses that have translated i many source words. This means that translating a sentence

of length n requires $n + 1$ stacks.⁹ The decoding process can be divided into the following steps:

- Tuple Lookup
- Hypothesis Extension
- Future Cost Estimation
- Recombination and Pruning

4.5.1 Tuple Extraction

Given an input sentence F , the decoder first extracts a set of matching source-side cepts along with their n -best translations to form a tuple inventory. The cept extraction routine extracts both gappy and non-gappy source cepts. Extraction of continuous source cept can be performed in polynomial time but extracting discontinuous source cepts requires an exponential number of look-ups (Galley and Manning, 2010). Notice that for a source sentence having length n , a maximum of $2^n - 1$ translation units can be extracted. Say we have a test sentence “ich muss jetzt gehen” having 4 German words. All possible translation units (total 15) can be represented in 4 bits as in Table 4.2. Each bit represents a source word. The left-most bit represents the first word and the right-most bit represents the fourth word. A zero bit means absence of that word in this phrase. For example a bit sequence “1 0 0 1” means a discontinuous cept “ich...gehen” and a bit sequence “0 1 0 1” means a discontinuous cept “muss...gehen”.

As can be seen, an n word source sentence can have $2^n - 1$ possible sequences.¹⁰ A brute-force algorithm would therefore require an exponential number of look-ups in the translation corpus to extract the valid (observed during training) discontinuous cepts.

⁹An additional stack is used to store the start (dummy) hypothesis that has translated no words.

¹⁰A sequence with all zero bits is not possible hence $2^n - 1$ instead of 2^n

4 A Joint Sequence Translation Model with Integrated Reordering

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Table 4.2: All Possible Discontinuous Phrases in a 4 Word Sentence

We can benefit from the fact that our model uses only minimal translation units. Most of the German cepts, continuous or discontinuous, only contain up to 2-3 words. This means that most of the bit sequences, no matter how big a source sentence is, are useless. But we need a mechanism to safely discard the not useful information and do only a few look-ups. Below we give an algorithm for this.

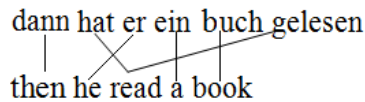


Figure 4.18: An Example to Demonstrate the Cept Extraction Algorithm

Given a source sentence $F = f_1, \dots, f_n$ our algorithm represents the $2^n - 1$ bit sequences in the form of a tree and performs search on that tree. The tree structure is formed such that each node in the tree represents a source word. Each node is represented by the index of the source word i . The root node is the first source word. Each node i has exactly $n - i$ children nodes such that the first child (left-most) is a source word at index $i + 1$, second child is a source word at index $i + 2$ and so on. For the example given in Figure 4.18 we form a tree-structure as shown in Figure 4.19. The tree represents $2^6 - 1$ cepts. The bit sequence “0 1 0 0 0 1” representing the source cept “hat...gelesen” can be obtained by starting at node 2 and traversing its right-most node i.e node 6. Similarly the bit sequence “0 0 1 0 1 0” representing the source cept

“er...Buch” can be obtained by starting at the node 3 and traversing its middle node represented by X ¹¹ until node 5.

In order to find all possible cepts the entire graph has to be searched with all possible permutations and stopping points. As the sentence length grows the tree grows exponentially making the search process computationally infeasible. However, notice that we are using very small translation units mostly containing 2-3 source words. This means that we do not have to search all possible permutations and can skip a sub-tree very early in the search process. Considering the example under discussion, we begin at the root node, finding cepts that begin with the word “dann”. At node 1 we search for the word “dann” in the cept lexicon. As we traverse to the node 2 we search for the cept “dann hat”. If we find the cept “dann hat” in the cept lexicon, we traverse the sub-trees of the node 2, looking for more cepts otherwise we can skip the entire tree represented by the node 2. But what about cepts like “dann hat er” or “dann hat Buch”? How can we skip these without checking whether or not these cepts appear in the cept lexicon? To ensure that we do not lose any cept, we also store a cept lexicon with partial cepts. It maintains information that the cept lexicon has cepts starting from the word “dann” or cepts having the first two words “dann hat” and so on. Again maintaining and doing a look-up from such partial cept lexicon is not computationally expensive because most cepts contain only 2-3 source words. If the look-up for “dann hat” in the cept lexicon fails then we try the partial cept lexicon for an entry such as “dann hat” to see if the cept “dann hat” has any promising continuation. Therefore our search for cept can have three results, hit, miss or partial hit. If there is a hit or a partial hit the search process traverses further into the sub-tree otherwise the algorithm skips the whole sub-tree. The same procedure is repeated by starting again at node 1 and traversing through other children of 1.

After all the sub-trees of the node i have been covered (skipped or traversed), we have all the cepts starting with the word f_i in F . The next step is to delete the node i and all its children except the left most child which is node $i + 1$

¹¹Due to the space limitation, the sub-trees are shown as non-terminals X , Y and Z .

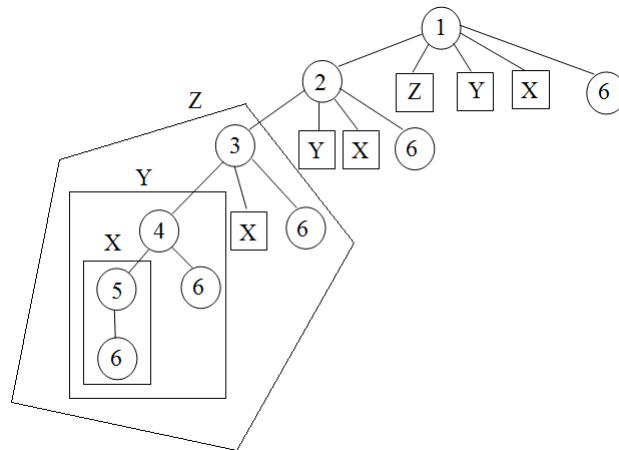


Figure 4.19: Search Tree for Source-side Cept Extraction

representing the next word. The search process starts again finding cepts for the word f_{i+1} . The node $i + 1$ then becomes the root. This process is repeated iteratively until there are no nodes left in the tree.

Algorithm 3 provides two routines to extract all possible cepts from a source sentence F . The *Extract-Cepts* routine traverses through the tree and calls the *Search-Cepts* function once for each source word in F . Each iteration collects all the source cepts having the word F_i as the starting word. The *Search-Cepts* routine calls itself recursively going deep in the tree as long as it is able to find cepts in *Cept Lexicon* or *Partial Cept Lexicon*.

Each source cept is extracted with the n-best translation options to form a tuple corpus. In our experiments we use 10-best English translations for each source cept.

Algorithmic Complexity

In order to extract all possible cepts from a German sentence of length n , at most $2^n - 1$ lookups are required. Let J be the size of the tuple lexicon that maintains both partial and complete cepts. Searching for a tuple in such a lex-

Algorithm 3 Cept Extraction Algorithm

root Root of tree structure
searchString A cept of one ore more source words
ceptLex A lexicon containing valid source cepts
p-ceptLex A lexicon of partial cepts
F Words in test sentence

Extract Cept Function

Function Extract-Cepts(*ceptLex* , *p-ceptLex*, *F*)

root := 1; {Index of the first word of *F*}
searchString := *F*[*root*];
while *root* is not empty **do**
 Search-Cepts(*root*, *searchString*, *ceptLex*, *p-ceptLex*, *F*);
 root := *Left-most child of root*; {Discard other children}

End Function

Search Cept Function

Function Search-Cepts(*root*, *searchString* , *ceptLex* , *p-ceptLex*, *F*)

if *searchString* is in *ceptLex* **then**
 extractedCepts.add(*searchString*); {add to a list of extracted cepts}
else
 if *searchString* is not in *p-ceptLex* **then**
 RETURN;
 for every child node *i* of *root* **do**
 Search-Cepts(*i*, *searchString* + *F*[*i*], *ceptLex*, *p-ceptLex*, *F*);

End Function

icon requires $O(\log J)$ time. The overall complexity of the algorithm therefore is $O(2^n \log J)$.

However, notice that it is impossible to have all possible $2^n - 1$ cepts. Most cepts that are extracted have an average of 2 source words. If we assume that only cepts with 3 or less source words are extracted the number of look-ups can be reduced to $\sum_{r=1}^d \binom{n}{r}$, where $d = 3$ (maximum size of the cept that can be extracted). Note that if we don't restrict the size of source cept to 3 then $d = n$ and $\sum_{r=1}^n \binom{n}{r} = 2^n - 1$. With this simplification, the average complexity of the algorithm would then be $\Theta(\sum_{r=1}^d \binom{n}{r} \log J)$.

4.5.2 Hypothesis Extension

During hypothesis expansion, the decoder picks a tuple from the inventory and generates the sequence of operations required for the translation with this tuple in light of the previous hypothesis. A hypothesis maintains the index of the last source word covered (j), the position of the right-most source word covered so far (Z). Recall that these variables are used in the conversion algorithm 1 discussed in Section 4.2.6. These are required to translate the tuples and their orderings into an operation sequence. A hypothesis also maintains, the number of open gaps, the number of gaps so far inserted, the previously generated operations, the generated target string, and the accumulated values of all the features discussed in Section 4.3.3. Other elements of a hypothesis include source cept (a set of words it covers), coverage vector specifying all the source words that have been covered by this hypothesis and its preceding hypotheses. Each hypothesis maintains a reference to its parent hypothesis. Figure 4.20 shows a sample hypothesis.

With the help of this information the decoder can now generate a sequence of operations required to cover the next source cept along with its English translation. After the operation sequence is generated, the probabilities and values for different features are computed. An overall cost of the hypothesis is calculated by summing the hypothesis cost with the cost of its parent hypothesis and a future cost estimate (Section 4.5.3). The coverage vector is

German : er
English : he
Coverage Vector : 101000
Operation Generation Variables: $j = 3$ $Z = 3$ Open-Gap-Indexes = [1]
Features Values: Language Model = -1.1 Operation Model = -1.25 Prior Probability = -2.90 Lexical Prob $p(E_j F_i) = -1.53$ Lexical Prob $p(F_i E_i) = -1.76$ Length Bonus = 2 Gap Penalty = 1 Open Gap Penalty = 1 Deletion Penalty = 0 Reordering Distance Penalty = 1 Gap Distance Penalty = 1 Source Gap Penalty = 0
Cost: -4.74
Previous Hypothesis : Back Pointer
Future Cost Estimate: -21.51

Figure 4.20: A Sample Hypothesis

updated and the new hypothesis is stored in the next Stack. A hypothesis may be recombined or pruned to keep the search space manageable (See Section 4.5.4).

4.5.3 Future Cost Estimation

In our decoding framework, like in the phrase-based system, the stacks are arranged based on the number of source words they have covered. One problem with using stacks is that there would be unfair comparisons of hypotheses that

have covered the same number of source words but have different coverage vectors, consequently resulting in more search errors. To overcome this problem an estimate called future cost is used in many SMT systems (See 2.2.7 on Page 65, for the discussion of future cost estimation).

Future cost estimates are computed in two steps:

- Future Cost for Cepts
- Future Cost for Larger Spans

Step 1 – Future Cost for Cepts

The first step is to estimate the future cost for each extracted cept. The cost of different feature components is first computed. Consider the test sentence “bitte schalten Sie Ihr Handy aus”. For a tuple “schalten aus – turn off” we can estimate the cost of different feature components as follow:

- Language Model:

$$p_{lm}(E) = p(\text{turn}) \times p(\text{off} \mid \text{turn})$$

- Operation Model:

$$p_{op}(E, F) = p(\text{Generate}(\text{schalten aus, turn off})) \times p(\text{Insert Gap} \\ \mid \text{Generate}(\text{schalten aus, turn off})) \times p(\text{Continue Source Cept} \\ \mid \text{Generate}(\text{schalten aus, turn off}) \text{ Insert Gap})$$

- Prior Probability:

$$p_{pr}(E) = p(\text{turn}) \times p(\text{off} \mid \text{turn})$$

- Length Bonus = 2

4 A Joint Sequence Translation Model with Integrated Reordering

- Deletion Penalty = 0
- Gap Penalty = 1
- Open Gap Penalty = 1
- Source Gap Penalty = 3

The prior probability model looks the same as the monolingual language model but since their language models are estimated from different data they give different estimates. We can not compute the values for the reordering distance penalty, and gap distance penalty because at the time of future cost estimation, the context is unknown.

Given all the feature values, the total cost of a cept pair is calculated. A German cept can have many possible English translations. We pick the tuples with the lowest cost. This process is repeated for all the extracted tuples.

Note that our future cost estimates are less accurate than those of the phrase-based system because our model uses minimal translation units. Phrasal SMT is aware during the preprocessing step that the words “nach meiner meinung” will be translated as a phrase. This is helpful for estimating a more accurate future cost because the context is already available. The same is not true for our model, to which only minimal units are available. Our model does not have the information that “nach meiner meinung” will be translated as a phrase during decoding. The future cost estimate available to our model for the span covering “nach meiner meinung” will have unigram probabilities for both the operation and language model:

$$p_{lm} = p(\text{in}) \times p(\text{my}) \times p(\text{opinion})$$

The operation model cost is estimated as:

$$p_{op} = p(\text{Generate}(\text{nach, in})) \times p(\text{Generate}(\text{meiner, my})) \times p(\text{Generate}(\text{meinung, opinion}))$$

Poor future cost estimates result in more search errors for our model making the search problem harder than that of the phrase-based model. The N-gram model faces the same search problem as ours but in their decoding framework future cost is not required, because there are 2^J stacks and all hypotheses in a stack cover the same source words. See Section 3.4.2 (Page 113) for details.

Step 2 – Future Cost for Larger Spans

The previous step provides us with an estimate of the cost for covering each cept. The next step is to estimate the cost for covering larger spans. Phrase-based system uses a dynamic programming technique to find the cost of each span (i, j) :

$$\text{cost}(i, n) = \min_{i \leq j \leq n} \{ \text{cost}(i, j) + \text{cost}(j + 1, n) \}$$

However, notice that this formulation does not work for calculation of future cost for gappy units. Consider calculating the future cost estimate of the $\text{span}(1, 8)$. If the best way to cover the span is through a gappy cept $1 \dots 4 \dots 8$, then the above formulation can not capture this.

We tailored the dynamic programming algorithm to handle this problem. We do not interfere with the existing algorithm but after the cost of the $\text{span}(i, j)$ has been calculated we do an additional pass for all the gappy cepts having start index i and end index j , calculating whether using the gappy cept gives a better cost. For example, for the $\text{span}(1, 8)$ we check whether $\text{cept}(1 \dots 4 \dots 8) + \text{cost}(2, 3) + \text{cost}(5, 8)$ gives a better estimate than $\text{cost}(1, 8)$. This additional check is performed at every step of the dynamic programming algorithm. This modification may work as long as gappy cepts don't interleave. Assume there was another cept $0 \dots 5$, that gives the best cost of covering the indexes 0 and

5. Our modification can not capture that the best cost is obtained through $cept(1 \dots 4 \dots 8) + cept(0 \dots 5) + cost(2, 3) + cost(6, 7)$.

Computing an accurate future cost estimate in presence of gappy units efficiently is a non-trivial problem. Using inaccurate estimates makes the search problem for our model even harder.

Using Future Cost during Search

The use of future cost estimates during decoding is done in the same way as in a phrase-based system. After the accumulative cost of a hypothesis is calculated, its coverage vector is updated. A future cost estimate is added to that cost before pruning, to make a fair comparison with other hypotheses that might be covering a different set of German words.

Future Cost with Gaps and Jumps To tighten the estimate, we add the cost of any forward and backward jumps that we are certain about to the future cost estimate. For example, if a certain hypothesis has “K” open gaps, it will do at least K many “Jump Backward (X)” operations and $0 - to - K$ “Jump Forward” operations depending upon the value of j and Z . We add the unigram probabilities of the “Jump Forward” and “Jump Backward(1)” operations, as estimated by the n-gram language model, to the future cost estimate.

Using Future Cost with Gappy Cepts Another problem using future cost with discontinuous cepts appears during decoding. If none of the words of the gappy cept is covered but words between start and end are covered, we get a wrong estimate. For a coverage vector $\{000100000\}$, the future cost estimate would be $cost(0, 2) + cost(4, 8)$. There is no efficient way to capture the information on gappy-unit $1 \dots 4 \dots 8$ during decoding if a word occurring within this cept has been covered.

4.5.4 Recombination and Pruning

Because the stack space grows exponentially, the search is intractable. To keep the search space manageable and time complexity polynomial we apply recombination and pruning. Recombination is performed on hypotheses having the same coverage vector, monolingual language model context, and operation model context.

We do histogram-based pruning, maintaining the 500 best hypotheses for each stack. We need a higher beam size to produce translation units similar to the phrase-based systems. For example, the phrase-based system can learn the phrase pair “nach meiner meinung – in my opinion” and generate it in a single step placing it directly into the stack three words to the right. Our system generates this example with three separate tuple translations “nach – in”, “meiner – my” and “meinung – opinion” in three adjacent stacks. Because “nach” is usually translated to “after” or “to” and “nach – in” is not a frequent translation unit, it will be ranked quite low in the first stack until the tuple “meiner – my” appears in the second stack. See Figure 4.21 for illustration. This drawback is inherent from using cepts i.e., minimal units for translation.

Another reason for using a higher stack size is that we do not apply any hard limit on reordering unlike state-of-the-art Phrase-based systems (which use a default distortion limit of 6). Both of these factors make the search problem of our model more difficult than that of the phrase-based model and the N-gram model. A final reason for using a higher stack size is that our model has much worse future cost estimates (discussed in the last section) than that of the phrase-based model.

4.5.5 Algorithmic Complexity

The overall complexity of the stack-based decoding with histogram-based pruning is $O(S \times T \times L)$, where S = size of the stack, T = Number of translation options and L = Length of the sentence. Because the stack size is constant and number of translation options is linear with the sentence length, the complexity can be simplified to $O(L^2)$. By applying the distortion limit the complexity

4 A Joint Sequence Translation Model with Integrated Reordering

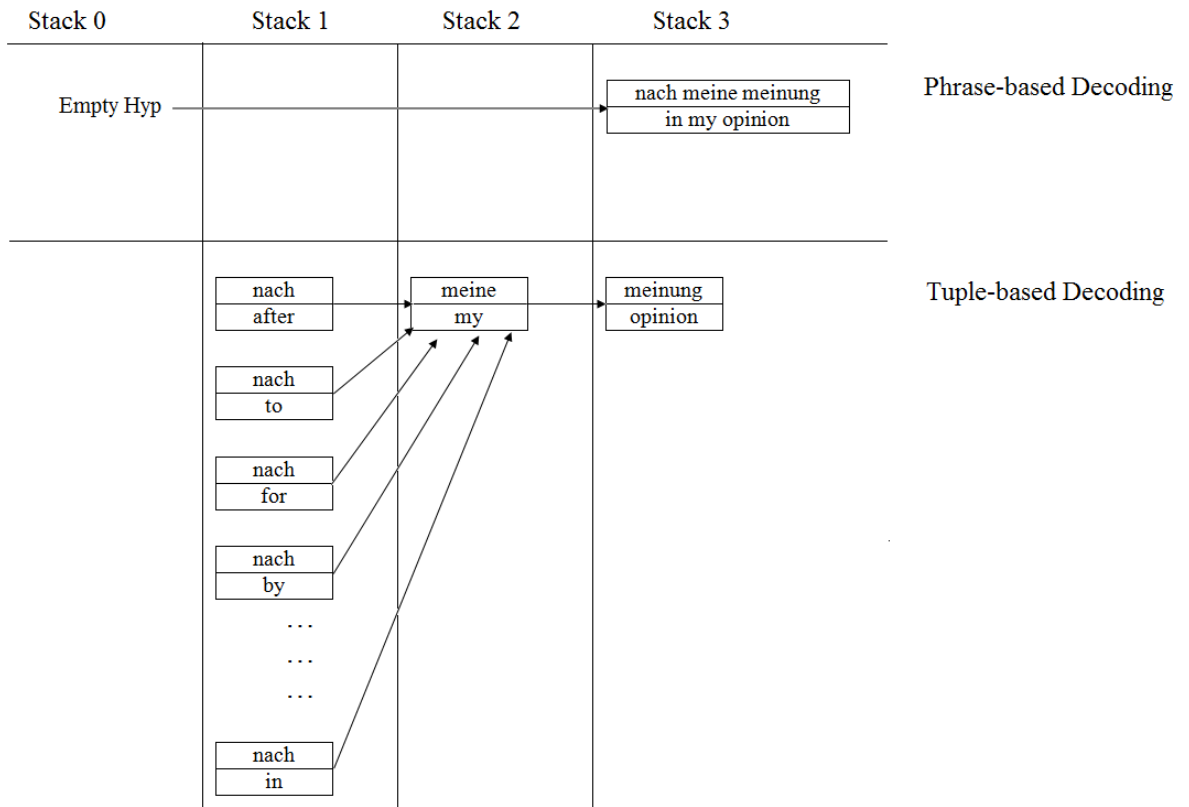


Figure 4.21: Comparison of Search - Phrases vs. Tuple Placement on Stack

can be further reduced to be linear (Koehn, 2010). This analysis holds for our decoder which is based on the same beam-search algorithm as used by the standard phrase-based decoder.

4.6 Training

The training of our model includes the following steps:

- Generating symmetrized alignments
- Post-editing of the alignments
- Extraction of the tuple corpus
- Generation of the operation sequence
- Estimation of the n-gram language models

The first step is to generate word alignments by running GIZA++ twice and symmetrizing with the grow-diag-final-and heuristic. The post-editing step removes target-side discontinuities and unaligned target words from the word alignments. See Section 4.4 for details. The next step is to extract a tuple corpus from the modified word alignments. Tuples are extracted with three probabilities, (i) lexical probability $p(e|f)$ (ii) Lexical Probability $p(f|e)$ and (iii) a joint probability $p(e,f)$. For the computation of lexical probabilities see Section 4.3.3. The English translations of a cept in the tuple corpus are ranked using the joint probability $p(e, f)$. It is used by the decoder as a filtering criterion to select the n-best translations of a source cept during search. The probability is estimated as:

$$p(e, f) = \frac{\text{count}(e, f)}{\text{Total number of tuples}}$$

Then we apply Algorithm 1 (See Section 4.2.6 for details) to convert the preprocessed aligned corpus into a sequence of translation operations. The

resulting operation corpus contains one sequence of operations per sentence pair.

In the final training step, the three language models are trained using the SRILM Toolkit. The operation model is estimated from the operation corpus. The prior probability model is estimated from the target side part of the bilingual corpus. The monolingual language model is estimated from the target side of the bilingual corpus and additional monolingual data.

4.7 Evaluation

We evaluated the system on three data sets with German-to-English, Spanish-to-English and French-to-English news translations. We used data from the 4th version of the *Europarl Corpus* and the *News Commentary* which was made available for the translation task of the *Fourth Workshop on Statistical Machine Translation*.¹² The bilingual data is obtained by concatenating the entire news commentary ($\approx 74\text{K}$ sentences) and Europarl, for the estimation of the translation model. The monolingual data comprises the English part of the bilingual data and the news commentary. The experiments are done with different bilingual and monolingual data sizes, mentioned explicitly in each section. All data is lower cased, and tokenized through the Moses tokenizer. The dev and dev-test sets are news-dev2009a and news-dev2009b which contain 1025 and 1026 parallel sentences. The feature weights of our system are tuned with Z-MERT (Zaidan, 2009). To compare our system against the baselines, we used the official evaluation data (news-test sets) from the Statistical Machine Translation Workshops 2007-2011 for all three language pairs (German, Spanish and French). All the systems are tuned using the dev set news-dev2009a. The converged vector is then used to decode the 5 test sets.

¹²<http://www.statmt.org/wmt09/translation-task.html>

4.7.1 Baseline Systems

We compare our system with three baseline systems: i) Moses, ii) Phrasal and iii) Ncode. A brief system description of each baseline along with its features is given below:

Phrase-based Systems

Two of our baselines are the state-of-the-art phrase-based systems Moses and Phrasal. Phrasal provides two main extensions to Moses: i) Hierarchical reordering model ii) Discontinuous source and target phrases. The common features used by both in the default configuration are:

1. Phrase translation model
 - Direct phrase-translation probability $p(e|f)$
 - Inverse phrase translation probability $p(f|e)$
 - Direct lexical weighting $p_{lex}(e|f)$
 - Inverse lexical weighting $p_{lex}(f|e)$
 - Phrase penalty
2. Target-side language model
3. Word Penalty
4. Reordering model
 - Distance-based reordering model
 - Lexicalized Reordering (msd-fe-bidirectional)¹³

The default configuration of Moses uses a word-based reordering model with phrase-based decoding, for the lexicalized reordering models.

¹³Using the three orientations: Monotonic, Swap and Discontinuous conditioned on both source and target languages. Both backward and forward models are used.

Phrasal uses a hierarchical reordering model (Galley and Manning, 2008). The additional features (Galley and Manning, 2010) used in Phrasal to enable discontinuous phrases are:

1. Source Gap Penalty: For a phrase having discontinuous source-side $F = f_1, \dots, f_L$, the source gap penalty calculates as:

$$d(F) = \sum_{i=2}^L f_i - f_{i-1} - 1$$

The default setting uses a hard limit of 15 which prevents the decoder from hypothesizing phrases that have source gap penalty beyond 15 words. The source gap penalty feature sums the length of all source gaps.

2. Target Gap Penalty: For a phrase having discontinuous target-side $E = e_1, \dots, e_L$, the source gap penalty calculates as:

$$d(E) = \sum_{i=2}^L e_i - e_{i-1} - 1$$

The default setting uses a hard limit of 7 which prevents the decoder from hypothesizing phrases that have target gap penalty beyond 7 words. The target gap penalty feature sums the length of all target gaps.

3. Source Gap Count: Counts the number of phrases with discontinuous source-side used, when translating a source sentence F .
4. Target Gap Count: Counts the number of phrases with discontinuous target-side used, when translating a source sentence F .
5. Crossing Alignments: This feature is a count of discontinuous phrases that have crossing alignments in the format cross-serial DTU and “bon-bon” (See Figure 4.22 for illustration).

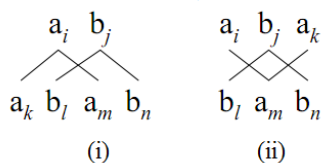


Figure 4.22: Discontinuous Phrases Configurations (i) Cross-serial DTU (ii) Bonbon

Among other defaults a stack size of 100 is used for Moses and 200 for Phrasal. A 5-gram English language model (the same as in our system) is used. Both systems use 20-best phrases for translation. A hard distortion limit of 6 is used in the default configuration of both systems. The optimization of the weight vector is done using MERT.

N-gram-based system

Our final baseline system is Ncode, the state-of-the-art N-gram-based system. The features used in the default configuration¹⁴ are:

1. Bilingual language model of tuples (linearized)
2. Target-side language model
3. Tuple bonus model
4. Target word bonus model
5. Tuple (unigram) model
 - Direct tuple-translation probability $p(e|f) = \frac{\text{count}(e,f)}{\text{count}(f)}$
 - Inverse tuple translation probability $p(f|e) = \frac{\text{count}(e,f)}{\text{count}(e)}$
 - Direct lexical weighting $p_{lex}(e|f)$

¹⁴<http://perso.limsi.fr/Individu/jmcrego/bincoder/>

- Inverse lexical weighting $p_{lex}(f|e)$

6. Reordering Models

- Lexicalized Reordering Model
- Distortion Penalty $|position(j) - position(j - 1)| - 1$

A list of 25-best English translations for each tuple is used. The decoding is done maintaining a stack size of 25. For a j word foreign sentence 2^j stacks are used. POS-based rewrite rules are extracted using a rule length of 6 tags. Source and target sentences in the bilingual corpus are POS-tagged using tree-tagger (Schmid, 1995). In order to make a fair comparison with our model, we removed the lexicalized reordering model feature from the Ncode baseline. Although we found that adding this feature mostly helps only a little (See Section 4.7.2 for the results of NCode with all its features). In all our experiments unless mentioned explicitly, we do not use the lexical reordering feature for Ncode. The optimization of the weight vector is done using MERT.

4.7.2 Results

In this section we will present the results, discussing different aspects of our model in comparison with the three baseline systems. The evaluation is done using BLEU (uncased) (Papineni et al., 2002).

Alignments

Word alignments for the experiments are generated with GIZA++ (Och and Ney, 2003). In order to obtain the best alignment quality, the alignment task is performed on the entire parallel data. We tried three symmetrization heuristics namely, intersection, union and grow-diag-final-and (Koehn et al., 2005). We choose, Phrasal (with continuous phrases and the hierarchical reordering model) as a representative of all the phrase-based baseline systems. Tables 4.3–4.5 indicate, that the alignments obtained by using the grow-diag-final-and (GDFA) heuristic give the best results for both the phrase-based and

N-gram-based systems as well as for our system. The systems with suffix “-PP” means applying the post-processing heuristic (discussed in Section 4.4) to the symmetrized alignments. See the next section for details.

Bilingual Data: 200K Sentences
 Monolingual Data: 500K Sentences

System	Union		Intersect		GDFA	
	Dev	Test	Dev	Test	Dev	Test
Ncode	17.05	17.44	16.24	17.10	18.15	18.70
Ncode-PP	17.76	18.19	17.31	17.81	18.20	19.02
Phrasal	17.60	18.02	17.52	18.26	18.43	18.84
Phrasal-PP	17.81	18.37	17.73	18.39	18.23	19.04
Our System	18.26	18.37	17.69	17.84	18.97	19.19

Table 4.3: Comparison of Symmetrization Heuristics + post-processing heuristic (PP) – German-to-English

Post Processing of Alignments

In Section 4.4, we discussed the heuristics that we apply in order to eliminate the unaligned target words and target-side gaps from the symmetrized word alignments. Without applying the post processing step we can not generate the operation sequence for a bilingual sentence, because Algorithm 1 does not support discontinuous target units. In this section we will present experiments to probe whether the post-processing heuristic is useful or hurts the performance of the baseline systems. We carried out experiments applying the post-processing heuristics to the translation task for all three language pairs and all the symmetrization heuristics.

Refer to Tables 4.3–4.5. The results using symmetrized alignments are shown in the baseline rows Ncode and Phrasal. The results after applying the post-processing heuristics to the symmetrized alignments in the baseline systems are shown as Ncode-PP and Phrasal-PP.

Most of our results indicate that our post editing of alignments is helpful in improving the results. The most notable improvement is observed in the case

System	Union		Intersect		G DFA	
	Dev	Test	Dev	Test	Dev	Test
Ncode	21.35	21.27	20.59	20.67	21.54	21.85
Ncode-PP	21.28	21.66	20.96	21.57	21.66	22.08
Phrasal	21.73	21.69	21.63	21.86	22.04	22.06
Phrasal-PP	21.48	21.68	21.69	21.83	21.98	21.91
Our System	21.35	21.66	20.95	21.89	21.86	22.36

Table 4.4: Comparison of Symmetrization Heuristics + post-processing heuristic (PP) – Spanish-to-English

System	Union		Intersect		G DFA	
	Dev	Test	Dev	Test	Dev	Test
Ncode	20.81	20.16	20.40	19.75	21.08	20.86
Ncode-PP	20.70	20.41	20.53	19.96	21.10	20.62
Phrasal	20.44	20.81	20.57	19.95	20.93	20.94
Phrasal-PP	20.35	20.49	20.53	20.86	20.95	20.73
Our System	20.73	20.64	20.62	20.70	21.59	21.05

Table 4.5: Comparison of Symmetrization Heuristics + post-processing heuristic (PP) – French-to-English

of the German-to-English (G-E) language pair. This can be explained by the fact that the words in the discontinuous English cepts in the G-E bilingual corpus are more distanced apart than in the other two language pairs. This results in a loss of useful translation information.

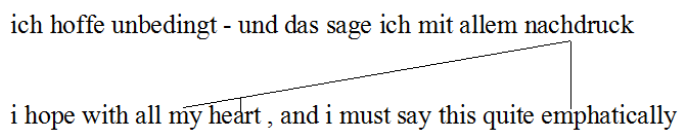


Figure 4.23: Long Distance Gappy Cept

See the example in Figure 4.23. The German word “nachdruck” is wrongly aligned with a gappy unit “my heart...emphatically”. To learn the translation of “nachdruck”, both the phrase-based and the N-gram-based system has to learn

a phrase/tuple “und das sage ich mit allem nachdruck – with all my heart , and i must say this quite emphatically”. Because only a phrase with 6 or less words is learned, the information describing how to translate “nachdruck” can not be learned from this example. The scenario is worse for the N-gram model. Recall Section 3.2.3, the embedded tuples¹⁵ (on the target-side) with long distance discontinuities are collapsed to form a single tuple. The N-gram model would not be able to learn any of the intervening tuples (“und – and”, “das – that”, “sage – must say”) in this example. The post-processing heuristic preserves these translation units by retaining the more valuable (sparse) link. For this example, the heuristic removes the links “nachdruck – my” and “nachdruck – heart” (because “my” and “heart” are more frequent vocabulary words than “emphatically”).

Figure 4.24 shows the distribution of target-side discontinuous units in the three language pairs. The X-axis shows the width of a target gap and the Y-axis shows the frequency of tuples have that width. As can be seen, the number of tuples having gap width of 5+ in German is twice as much as that of Spanish and French.

Another reason for this result is the higher number of unaligned target-side words in German-to-English bilingual data, which is more than twice that in French-English and Spanish-English parallel data. Figure 4.25 shows statistics on unaligned target words in each of the data sets.

The results also improve, although slightly, in case of the Spanish-to-English (S-E) task and drop in the case of the French-to-English task. For all the reported experiments in the next sections we use the GDFA symmetrization heuristic with post-processed alignments for all the baseline systems, except for the configuration of Phrasal that uses discontinuous source and target phrases. For the configuration of Phrasal that does not use discontinuous phrases we apply post-processing heuristic before feeding the alignments into the system.

¹⁵A solution to this problem as proposed in (Crego and Yvon, 2009) is to use split rules (See Section 3.2.3 on Page 106) . The Ncode baseline system, however, does not support this feature.

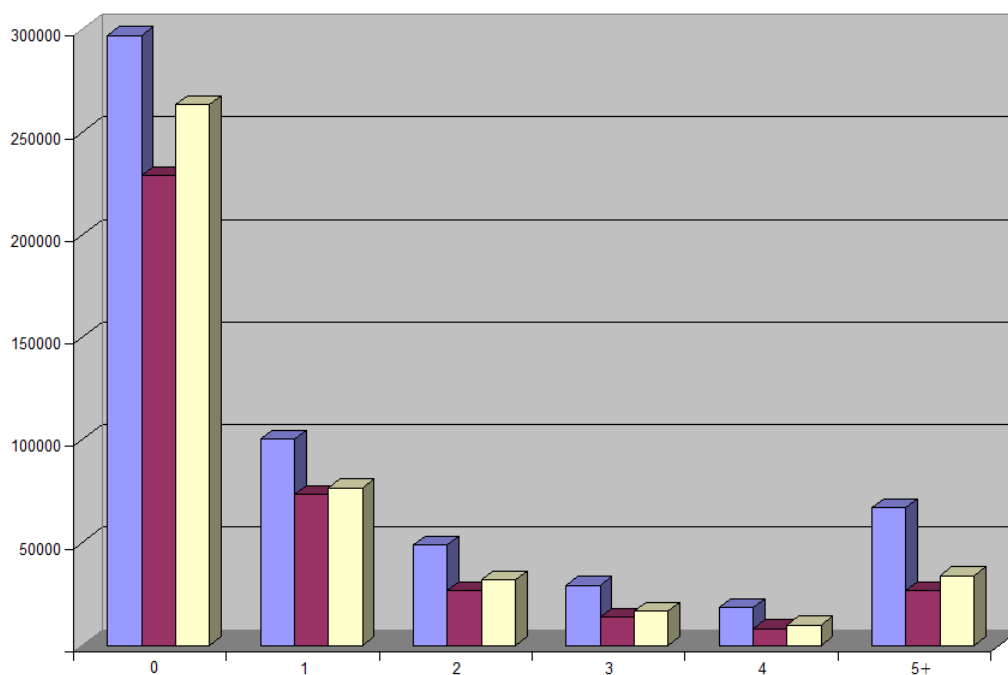


Figure 4.24: Target-Side Discontinuous Units – X-Axis = Width of Target Gap, Y-Axis = Frequency of Tuples having that Width – Series 1 = German, Series 2 = Spanish, Series 3 = French – Stats collected from 1000K Parallel Data

Experiments Comparing Our Systems

In our initial results we compared the two variants of our system. Our first system \mathbf{OS}_{ag} uses all the features discussed in Section 4.3.3. In our second system \mathbf{OS}_0 we eliminate the translation units with discontinuous source cepts from the list of 10-best English translations, and disable the source gap penalty feature. Both our systems do not apply any hard limit on reordering and also do not put any hard constraint on the size of the discontinuous source cepts in the system \mathbf{OS}_{ag} .

From the results in Table 4.6, we found that our system with gappy cepts \mathbf{OS}_{ag} does not help improve performance over its counter part which does not

4 A Joint Sequence Translation Model with Integrated Reordering

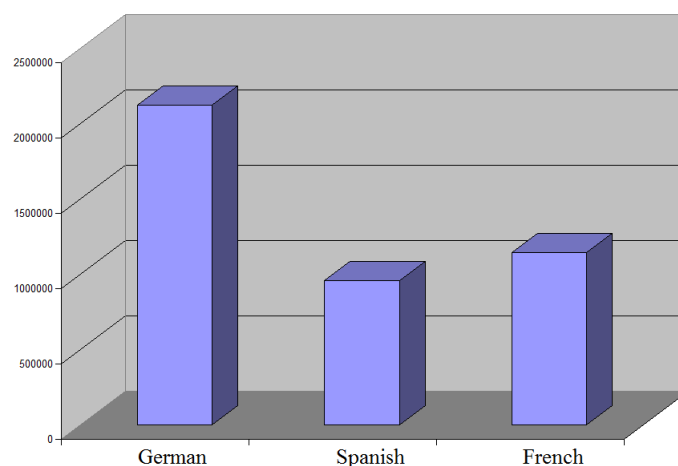


Figure 4.25: Target-Side unaligned Units – Statistics collected from 1000K Parallel Data

	OS_{ag}		OS_0	
	Dev	Test	Dev	Test
Bilingual Data: 200K Sentences				
Monolingual Data: 500K Sentences				
DE	18.26	18.81	18.97	19.19
ES	21.72	22.10	21.86	22.36
FR	21.56	21.09	21.59	21.05

Table 4.6: Comparison of Our Systems – DE = German, ES = Spanish, FR = French

use discontinuous translation units. The accuracy of our system OS_{ag} drops in the G-E and S-E translation results, and stays the same in the case of F-E task. In an analysis of the output we found the following reasons for this result:

- Using tuples with source gaps increases the list of extracted n-best translation tuples multiple times making the search problem even more difficult. Table 4.7 shows the number of tuples (with and without gaps) extracted when decoding the test file with 10-best translations.
- We observed that many of the tuples with gappy source cepts were wrong alignments. For example “der...die – which” has appeared more than

Source	German	Spanish	French
OS_{ag}	965515	1705156	1473798
OS₀	256992	313690	343220

Table 4.7: 10-best Translation Options With & Without Gaps and using our Heuristic

400 times in the tuple corpus. With the help of the source gap penalty we are able to deal with such tuples. However, given the poor future cost estimate our model has, it is hard to completely eliminate these during stack pruning. They consume the stack space and prune potential hypotheses that, if allowed to continue, may turn out to be the best hypothesis in the end. Table 4.8 shows the MERT tuned vector and the count of the source gap penalty feature (SG) obtained in dev and test sets. The value of **SG** feature penalizes more in case of the French-to-English task, which helps in eliminating the gappy units effectively from the search space, giving the same translation accuracy as the system that uses no gappy units. In comparison the value of the **SG** feature is less penalizing in case of S-E and even less in the G-E task, resulting in a higher number of search errors, which causes a drop in the translation accuracy.

- The future cost is poorly estimated in the case of tuples with gappy source cepts, causing search errors. The dynamic programming approach of calculating the future cost for bigger spans gives erroneous results when gappy cepts can interleave. Refer back to Section 4.5.3 for the discussion of future cost estimation with discontinuous units.

Comparison with the Baseline Systems

In this section we compare our system with only continuous cepts (**OS₀**) with the three baseline systems. In order to make a fair comparison, we provide

4 A Joint Sequence Translation Model with Integrated Reordering

	Final Vector {LM, OP, PM, Lex-F, Lex-E, LB, DP, GP, OG, RP, GD, SG }	SGP	
		Dev	Test
DE	{1.0 0.90 0.079 0.93 0.54 2.15 -2.14 -0.02 -0.07 -0.02 -0.02 -0.05 }	218	192
ES	{1.0 0.65 -0.37 0.58 0.93 0.78 0.88 -0.53 0.24 0.34 -0.73 -0.67 }	37	67
FR	{1.0 0.65 -0.17 0.68 0.79 1.1 0.82 0.086 0.68 0.34 -0.97 -2.59 }	13	11

Table 4.8: MERT-tuned Feature Vector for **OS_{ag}** Systems and Source Gap Penalty Count (SGP) – LM = Target-Side Language Model, OP = Operation Model, PM = Prior Probability Model, Lex-F = $\log(p_{lex}(e|f))$, Lex-E = $\log(p_{lex}(f|e))$, LB = Length Bonus, DP = Deletion Penalty, GP = Gap Penalty, OG = Open Gap Penalty, RP = Reordering Penalty, GD = Gap Distance Penalty, SG = Source Gap Penalty

the baseline systems with the same alignments as ours i.e., applying G DFA symmetrization and post-processing heuristics. For the configuration of Phrasal that uses discontinuous phrases, we do not apply the post-processing heuristic, because having the ability to learn discontinuous phrases on both source and target-side, enables Phrasal to avoid data sparsity and loss of information, discussed in the previous section.

The experiments are done on three different data scales. In the small data configuration we use 200K sentences of parallel data and 500K sentences of monolingual English data. In the medium data configuration we use 500K sentences of parallel data and 1M sentences of monolingual English data. In the large data configuration we use 1M sentences of parallel data and 2M sentences of monolingual English data. In all the configurations, the monolingual data contains the English part of the bilingual data + additional monolingual data from Europarl.

We used Kevin Gimpel’s tester¹⁶ which uses bootstrap resampling (Koehn, 2004b) to test which of our results are significantly better over which of the

¹⁶<http://www.ark.cs.cmu.edu/MT/>

4 A Joint Sequence Translation Model with Integrated Reordering

	Moses		Phrasal		Phrasal _d		Ncode		OS ₀	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
S	18.36*	18.97	18.23*	19.04	18.43*	18.97	18.20*	19.02	18.97	19.19
M	18.89*	19.11*	19.13	19.29*	19.17	19.88	18.64*	19.28*	19.32	20.11
L	19.79	20.15	19.76	19.92*	20.05	20.45	19.16*	19.77*	19.64	20.36

Table 4.9: Comparison of German-to-English on (S)mall, (M)edium and (L)arge Scaled Data – Small = 200K Parallel Data, 500K Monolingual English Data – Medium = 500K Parallel Data, 1000K Monolingual English Data – Large = 1000K Parallel Data, 1000K Monolingual English Data

	Sentences	Moses	Phrasal	Phrasal _d	Ncode	OS ₀
MT07	2002	24.26*	24.79	25.02	24.27*	24.85
MT08	2039	19.54	19.37	19.64	19.01*	19.53
MT09	2987	18.73*	18.45*	19.00	18.36*	19.18
MT10	2470	18.58*	18.37*	18.96	18.86	19.08
MT11	2975	17.38*	17.12*	17.58	17.39*	17.84

Table 4.10: Comparison of German-to-English on 5-Test Sets (Large Scaled Data)

baseline results. We mark a baseline “*” representing that our model shows a significantly better improvement over this baseline result with a confidence of $p < 0.05$. We use 1000 samples during bootstrap resampling.

Tables 4.9 and 4.10 show that our German-to-English results are better than all the baseline systems, except Phrasal with discontinuous phrases (*phrasal_d*) where the performance of our system is similar to that of Phrasal. All our shared-task results are significantly better than the Ncode baseline results showing that our model is better able to handle language pairs with high reordering. In Figure 4.26, for example, the verb final “investiert – invested” is successfully reordered to its correct position in our system that applies a reordering pattern “haben investiert – have invested”. The Ncode system is unable to trigger a POS rule to hypothesize “investiert – invested” after

Source	Beide Länder haben Millionen von Dollar in die Untersuchungen investiert
Ref	Both countries invested millions of dollars into surveying.
Moses	Both countries have millions of dollars in the investigation.
Phrasal	Both countries have invested millions of dollars in the investigation.
Ncode	Both countries have millions of dollars invested in the investigations.
OS₀	Both countries have invested millions of dollars in the investigation.

Figure 4.26: Example-1 from Test MT09 – Demonstrating the Better Reordering Mechanism

Source	Ihre Tanzgrundlagen haben mir in Flamenco viel geholfen.
Ref	Her dances ' titles suggested me some elements of Flamenco.
Moses	I have their Tanzgrundlagen in Flamenco much help.
Phrasal	I have been in their Tanzgrundlagen Flamenco much help.
Phrasal_d	Their Tanzgrundlagen have helped a lot in Flamenco me.
Ncode	I have their Tanzgrundlagen in Flamenco much help.
OS₀	Their Tanzgrundlagen have helped me a great deal in Flamenco.

Figure 4.27: Example-2 from Test MT09 – Demonstrating Better Reordering Mechanism

“haben – have” is inserted. The output in Moses drops the main verb, an error that we frequently observed in the output of Moses. Phrasal is able to apply a discontinuous phrase “haben X investiert – have invested” to produce the correct output just as our system did.

Figure 4.27 depicts another example where the verb “geholfen – helped” is correctly reordered after auxiliary “haben – have”, only by our system because of its strong ability to represent the dependency between “haben – have” and “geholfen – helped”. Strong reliance on the language model to guide reordering hampers the performance of other systems in this case because of the presence of two unknown words “Flamenco” and “Tanzgrundlagen” in the source sentence. The language model is unable to compensate for the dis-preference of the translation model for non-local reordering and its strong bias towards the phrasal unit “viel geholfen – much help”.

Source	die USA haben bereits signalisiert, dass sie über konkrete Zahlen nicht verhandeln wollen
Ref	The United States has let it be known that it is unwilling to negotiate precise numbers
Moses	The United States has already indicated that they have concrete figures do not want to negotiate
Phrasal	The US have already indicated that they do not want to negotiate on specific figures
Ncode	The United States have already indicated that they have concrete figures do not want to negotiate
OS₀	The US have already indicated that they do not want to negotiate on specific figures

Figure 4.28: Example-3 from Test MT08 – Demonstrating Ability to Memorize Phrases

Our model exhibits complex reordering in this example. After the insertion of “haben – have”, the model must jump to the end of the sentence to cover the verb final “geholfen – helped”. A gap is inserted for the skipped sequence of words “mir in flamenco viel” which consists of the object of the sentence “mir – me”, an adverbial phrase (AP) “viel – a great deal” and a prepositional phrase (PP) “in Flamenco – in Flamenco”. After generating “geholfen – helped” the translator jumps back and generates the object “mir – me” and then inserts another gap (for the prepositional phrase) and generates the adverbial phrase. The translator finally jumps back to the open gap and generate “in Flamenco”.

The discontinuous phrase-based system is able to capture the dependency between “haben – have” and “geholfen – helped” through a discontinuous phrase “haben X geholfen – have helped”, however it is not able to displace the object “mir – tuple” to its correct position. The lexicalized reordering model is unable to justify the jump from “geholfen” to “mir”.

In Figure 4.28, we demonstrate the ability of our model to memorize and displace phrasal units just as the phrase-based system. Notice that although our model is using smaller translation units “nicht – do not”, “verhandeln –

4 A Joint Sequence Translation Model with Integrated Reordering

negotiate” and “wollen – want to”, it is able to memorize the phrase translation “nicht verhandeln wollen – do not want to negotiate” as a sequence of translation and reordering operations. And it is able to successfully displace the whole unit over the prepositional phrase “über konkrete Zahlen – on specific figures” capturing both local and non-local dependencies.

	Moses		Phrasal		Phrasal_d		Ncode		OS₀	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
S	21.88	21.97*	21.98	21.91*	22.01	21.93*	21.66	22.08	21.86	22.36
M	22.29	22.89	22.51	22.71	22.83	23.17	22.60	22.84	22.23	22.98
L	23.10	23.38	23.41	23.81	23.62	23.76	22.96	23.30	22.90	23.47

Table 4.11: Comparison of Spanish-to-English on (S)mall, (M)edium and (L)arge Scaled Data

	Sentences	Moses	Phrasal	Phrasal_d	Ncode	OS₀
MT07	1997	34.81	35.46	35.05	34.21	34.41
MT08	2035	22.87	23.12	23.17	22.75	22.82
MT09	2935	24.38	24.59	24.63	24.71	24.39
MT10	2470	25.55	25.78	25.66	25.72	25.66
MT11	2960	25.72*	26.14	26.17	26.32	26.25

Table 4.12: Comparison of Spanish-to-English on 5-Test Sets (Large Scaled Data)

Our results from the Spanish-to-English translation task are shown in Tables 4.11 and 4.12 and the French-to-English translation task are given in Tables 4.14 and 4.15.

Our Spanish-to-English system has roughly the same translation quality as the other baseline systems. Our results do not show statistically significant improvements over baselines. In some cases our results are worse than the other systems.

In three out of six test-sets (dev-test, MT07 and MT08), our results are slightly better than NCode. In the other three the NCode system is better. Compared to Moses our results are slightly better on three test-sets (dev-test,

4 A Joint Sequence Translation Model with Integrated Reordering

JumpSize	German	Spanish	French
0	21005591	23925136	24982196
1	2950802	5208091	4648931
2	2001122	2034345	1980326
3	1358259	1183803	1187276
4	951518	677759	696869
5	707524	414637	461838
6+	3031509	1463074	1816495

Table 4.13: Statistics on Reordering Distances in German, Spanish and French calculated from Bilingual Training Corpus

MT10 and MT11). On the test set MT07 Moses shows a statistically significant improvement over our results.

	Moses		Phrasal		Phrasal _d		Ncode		OS ₀	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
S	20.86*	20.60*	20.95*	20.73	21.00*	20.82	21.10*	20.62*	21.59	21.05
M	21.33*	21.37	21.62*	21.64	21.85	21.79	21.64	21.55	22.00	21.51
L	21.78	21.96	22.09	21.89	22.20	21.98	22.08	21.67	21.98	21.81

Table 4.14: Comparison of French-to-English on (S)mall, (M)edium and (L)arge Scaled Data

The results from both continuous and discontinuous Phrasal systems (which use Galley’s hierarchical reordering model) show best results on the Spanish-to-English task. In five out of six test-sets Phrasal shows better results than ours. However, a noteworthy point is that using discontinuous phrases does not yield any better results than using only the contiguous phrases on the Spanish-to-English task.¹⁷ Using discontinuous phrases showed statistically significant improvements over using continuous phrases, in some cases (MT08 and MT09) in the German-to-English translation task. These results can be

¹⁷We used the default source and target gap sizes of 15 and 7 respectively. Perhaps tuning these for the Spanish-to-English task yield better results.

4 A Joint Sequence Translation Model with Integrated Reordering

	Sentences	Moses	Phrasal	Phrasal_d	Ncode	OS₀
MT07	1994	28.54	28.97	29.28	28.33	28.25
MT08	2031	21.19	21.29	21.28	21.25	21.30
MT09	2970	24.61*	24.51*	24.73*	24.28*	25.06
MT10	2460	23.69*	22.74*	23.09*	23.96	24.04
MT11	2957	25.17*	25.15*	25.55	24.92*	25.58

Table 4.15: Comparison of French-to-English on 5-Test Sets (Large Scaled Data)

explained by the fact that Spanish and English have relatively similar word order and require less reordering. In comparison the German–English language pair exhibits a lot of reordering. Short distance reorderings of 3-4 words can be effectively captured inside phrases, however the translation model in the continuous phrase-based system can not capture long distance reordering. These reorderings have to be justified by the reordering model and language model scores. The discontinuous phrase-based system is able to represent these long distance dependencies inside of the translation model.

In order to measure the amount of reordering in the three language pairs, we stored the jump sizes (reordering distances) between two tuples along with their frequencies in the bilingual training corpus. See Table 4.13 for statistics. A jump size of 0 indicates that the next tuple is generated monotonically. As evident from the statistics, French and Spanish are more monotonic than German. The number of cases with a jump distance of 6+ in Spanish is half than in German. French is closer to Spanish but has more cases of long-range reordering. The number of cases with a jump distance of 3, 4 and 5 words is also higher in the case of German-English bilingual data.

Our French-to-English results show significantly better improvement over the Moses and Phrasal baselines for three test-sets (MT09, MT10 and MT11) and for two test-sets (MT10 and MT11) for the Ncode baseline. On the MT07 baseline both Moses and Phrasal show better performance.

N-Gram System with additional Features

	Ncode	Ncode_l	Ncode_{l+p}	OS₀
Dev	19.16*	19.26*	19.15*	19.64
Dev-Test	19.77*	19.71*	20.01	20.36
MT07	24.27*	24.21*	24.34*	24.85
MT08	19.01*	19.12*	19.18*	19.53
MT09	18.36*	18.37*	18.65*	19.18
MT10	18.86	18.64*	18.94	19.08
MT11	17.39*	17.49*	17.72	17.84

Table 4.16: Comparison to Full N-gram System l = Lexical Reordering, $l+p$ = Lexical Reordering + POS-based bilingual Model – German-to-English

In Section 4.7.2, we presented the results for the N-gram system by disabling the lexical reordering feature. Our purpose was to directly compare the two reordering mechanisms. In this section we load the N-gram model with additional features namely lexicalized reordering and adding a POS-based model trained on bilingual tuple corpus of source and target POS-tags, as presented in Crego and Yvon (2010).

	Ncode	Ncode_l	Ncode_{l+p}	OS₀
Dev	22.96	23.17	23.20	22.90
Dev-Test	23.30	23.78	23.48	23.47
MT07	34.21	34.75	34.82	34.41
MT08	22.75	23.05	22.87	22.82
MT09	24.71	24.72	24.61	24.39
MT10	25.72	25.87	25.77	25.66
MT11	26.32	26.36	26.48	26.25

Table 4.17: Comparison to Full N-gram System l = Lexical Reordering, $l+p$ = Lexical Reordering + POS-based bilingual Model – Spanish-to-English

Table 4.16 shows that adding the lexicalized reordering model to Ncode does not yield statistically significant improvements. Our gains on the German-to-English task remain significantly better over Ncode with the lexicalized

4 A Joint Sequence Translation Model with Integrated Reordering

reordering model for all the test-sets. Adding a POS-based tuple model however shows improvements over the baseline Ncode system in three test-sets (Dev-test, MT10 and MT11). But the performance of our system is still better than Ncode system with lexicalized reordering and POS-based tuple model.

Using the lexicalized reordering model however, shows statistically significant improvements for the Spanish-to-English task in some test-sets (Dev-test and MT07). Using these additional features the Ncode model gives better performance than our system for all the test-sets.

	Ncode	Ncode_l	Ncode_{l+p}	OS₀
Dev	22.08	22.17	22.19	21.98
Dev-Test	21.67	21.75	21.40	21.81
MT07	28.33	28.01	28.46	28.25
MT08	21.25	21.34	21.10	21.30
MT09	24.28*	24.63*	24.05*	25.06
MT10	23.96	24.02	23.81*	24.04
MT11	24.92*	25.17*	24.68*	25.58

Table 4.18: Comparison to Full N-gram System l = Lexical Reordering, l+p = Lexical Reordering + POS-based bilingual Model – French-to-English

In case of the French-to-English task, the lexicalized reordering feature significantly improves the performance on MT09 and gives slight gains on the other test-sets. However, the performance of our system is still significantly better over NCode with lex and lex + POS system on MT09 and MT11.

Experiments on Distortion Limit

	Moses_{dl=6}		Moses_{no-dl}		OS_{gd=6}		OS₀	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
200K	18.36	18.97	17.01	17.61	18.79	19.13	18.97	19.19
500K	18.89	19.11	17.72	18.47	19.32	19.94	19.32	20.11
1000K	19.99	20.15	17.44	18.67	19.80	20.14	19.64	20.36

Table 4.19: Experiments on Distortion Limit – German-to-English

4 A Joint Sequence Translation Model with Integrated Reordering

	Moses_{dl=6}		Moses_{no-dl}		OS_{gd=6}		OS₀	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
200K	21.88	21.97	20.23	20.02	21.86	21.87	21.86	22.36
500K	22.29	22.89	20.47	21.33	22.29	22.92	22.23	22.98
1000K	23.10	23.38	20.94	21.56	22.70	23.37	22.90	23.47

Table 4.20: Experiments on Distortion Limit – Spanish-to-English

	Moses_{dl=6}		Moses_{no-dl}		OS_{gd=6}		OS₀	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
200K	20.86	20.60	19.32	19.49	21.21	20.98	21.59	21.05
500K	21.33	21.37	19.36	19.62	21.99	21.51	22.00	21.51
1000K	21.78	21.96	20.49	20.72	21.84	21.86	21.98	21.81

Table 4.21: Experiments on Distortion Limit – French-to-English

In the tables 4.19–4.21 we do a comparison between our system and the phrase-based system without hard reordering limit. We also present a variation of our system (**OS_{gd=6}**) where, like the phrase-based systems, we apply a hard constraint which limits reordering to no more than 6 positions. Specifically, we do not extend hypotheses that are more than 6 words apart from the first word of the left-most gap during decoding. In other words we do not extend hypotheses that have a gap distance penalty of more than 6 words. For the baseline experiments, we choose Moses as a representative of the other baseline system, Phrasal¹⁸.

Tables 4.19–4.21 show that the result for the phrase-based system drop by more than 1 BLEU point without the hard reordering limit. In comparison our system (**OS₀**) without any hard reordering limit shows the same performance as that of the system with hard reordering limit (**OS_{gd=6}**). In fact the results in case of German-to-English translation task are better for the system with

¹⁸The assumption is that the underlying model and features in Phrasal are the same as the Moses and that Phrasal will have the same problem as Moses. Moreover running Phrasal without a hard reordering limit will require a lot of days to run MERT training.

Source	Die EZB ist bestrebt, die Inflationsrate unter zwei Prozent, oder zumindest knapp an der zwei-Prozent-Marke zu halten.
Ref	The ECB wants to hold inflation to under two percent, or somewhere in that vicinity.
Moses	The ECB is endeavouring, the rate of inflation below two per cent, or at least less than in the zwei-Prozent-Marke.
Phrasal	The ECB is striving to the rate of inflation below two per cent, or at least in the short zwei-Prozent-Marke.
Ncode	The ECB is striving to inflation below 2%, or at least to keep running out of the zwei-Prozent-Marke.
OS₀	The ECB is anxious to keep inflation under 2% , or at least close to the zwei-Prozent-Marke.

Figure 4.29: Example-4 from Test MT08 – Where No Reordering Limit Helps

Source	Anderson nahm 32 erstklassige Wickets mit je 33 Runs, um eine Position in dieser Tour zu verdienen.
Ref	Anderson took 32 first-class wickets at 33 runs each to earn a place on this tour.
Moses	Anderson, 32 first-rate wickets with the 33 runs, a position in this tour to earn .
Phrasal	Anderson , 32 first-rate wickets with ever 33 runs, a position on this tour .
Ncode	Anderson took wickets 32 first-rate with ever 33 runs, in order to earn a position in this way.
OS₀	Anderson took 32 first-rate wickets with ever 33 runs, in order to earn a position on this tour.

Figure 4.30: Example-5 from Test MT08 – Where No Reordering Limit Helps

no reordering limit.

Figure 4.29 demonstrates an example where having no distortion limit is helpful. Our system without a reordering limit moves the verb-phrase “zu halten” by jumping over 12 English tokens to generate the English verb-phrase “to keep” at its correct position after generating “Die EZB ist bestrebt, – The ECB is anxious”.

Another example is shown in Figure 4.30, where our system is able to displace the verb phrase “zu verdienen” over 7 words to place the English verb phrase “to earn” in its correct position. The hard distortion limit in Moses and Phrasal restricts them from generating the hypothesis “to earn” in its correct place.

Contribution of Feature Functions

In this section we study the contribution of each feature in the overall performance of the system. We removed the features from our main system \mathbf{OS}_0 in groups and report the accuracy without the removed group of features. We tested our system without prior probability model (\mathbf{OS}_{pm}), without lexical probability models (\mathbf{OS}_{lex}), without gap and open gap features ($\mathbf{OS}_{\text{gp-og}}$) and without the distance-based penalty features i.e., reordering distance and gap distance ($\mathbf{OS}_{\text{rd-gd}}$). The effect of removing source word deletion model and its related deletion penalty feature is also studied. The role of the language model in our system and the phrase-based system is reported at the end.

Bilingual Data: 500K Sentences
 Monolingual Data: 1000K Sentences

	\mathbf{OS}_{pm}		\mathbf{OS}_{lex}		$\mathbf{OS}_{\text{gp-og}}$		$\mathbf{OS}_{\text{rd-gd}}$		\mathbf{OS}_0	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
DE	19.50	20.10	17.08	17.76	19.28	19.88	18.30	18.52	19.32	20.11
ES	22.11	22.81	18.53	18.95	21.99	22.81	20.12	20.46	22.23	22.98
FR	21.94	21.55	21.30	20.78	21.49	21.04	20.11	19.70	22.00	21.51

Table 4.22: Contribution of Different Feature Functions

4 A Joint Sequence Translation Model with Integrated Reordering

Table 4.22 shows the contribution of each set of features in the three language pairs. Removing the prior probability model yields roughly the same results as our full model. All the other set of features contribute towards the BLEU score. Removing the IBM-1 lexical probability models results in a drop of 2.35 BLEU points in the German-to-English task, 4 BLEU points in Spanish-to-English task and 0.73 BLEU points in the French-to-English task. The reordering-based gap and open gap penalties help to control the unjustified long-distance reorderings, evidence of which is not obtained from the operation and monolingual language model scores. Removing these two features consistently drop the translation quality. The distance-based penalties are important to enable no hard distortion limit during decoding. Removing these features results in a drop of more than 1.5 BLEU points in our system with no hard reordering limit. However, if we use the hard-reordering limit, removing these features does not hurt. Refer back to our system $OS_{gd=6}$ in the Tables 4.19–4.21, which does not use the gap distance and reordering distance penalties as soft-constraint. Using a window of 6 or less jumps, dramatically reduces decoding complexity and cuts the search space by multiple folds. This results in a lower number of search errors. Not using gap and reordering distance penalties as a soft constraint in our system OS_0 causes massive search errors, and a drop in the translation quality.

Bilingual Data: 1000K Sentences
 Monolingual Data: 2000K Sentences

	$OS_{gd=6}$		$OS_{gd=6} - PP$	
	Dev	Test	Dev	Test
DE	19.80	20.14	19.75	20.22
ES	22.70	23.37	22.64	23.53
FR	21.84	21.86	21.66	21.64

Table 4.23: Removing Source Word Deletion Model

Source Word Deletion: In order to evaluate the effect of the source word deletion model in our generative story and the deletion penalty feature, we

4 A Joint Sequence Translation Model with Integrated Reordering

removed the source-unaligned tuples (where a source word is aligned to null). Such tuples are represented through **Generate Source Only(X)** operation in our operation corpus. To remove this operation from the operation corpus we modified the alignments by applying a post-processing step that removes the source-unaligned tuples (where a source word is aligned to null). To achieve this, we flip the alignments and apply the procedure discussed in Section 4.4 and then flip the alignments again before generating the operation corpus. The post-processing step removes all source and target unaligned words by connecting them to the left or right tuples based on the counts collected in a preprocessing step. Table 4.23 gives results (**OS_{gd=6} – PP**) after applying the post-processing heuristic on the source-side of the bilingual corpus. The results on all language pairs are consistent and show that not using a source word deletion model and deletion penalty does not hurt. All the results stay roughly the same with a difference of +/- 0.25 BLEU points.

Bilingual Data: 1000K Sentences
 Monolingual Data: 2000K Sentences

	OS_{gd=6}-LM		Moses_{LM}		NCode_{LM}	
	Dev	Test	Dev	Test	Dev	Test
DE	17.93	18.00	16.44	16.61	17.20	17.70
ES	20.80	21.48	19.94	19.87	20.50	20.66
FR	19.70	19.69	17.86	17.82	19.47	19.13

Table 4.24: Removing Monolingual Language Model Features from Moses, NCode and Our System

Monolingual Language: In another experiment we removed the monolingual language model feature from both our system and Moses. We also removed the prior probability model feature from our system to ensure that no assistance, to judge the fluency of English translations, is provided to both systems other than their translation models. The purpose of this experiment was to measure the strength of the translation models, and their reliance on the monolingual language model, in the two frameworks.

Table 4.24 shows that the results of the baseline systems and our system without the language model feature. All results drop significantly as compared to their counterparts where the language model feature was used. However, the results in the phrase-based system (**Moses**–**LM**), without the language model are much worse than our system (**OS_{gd=6}**–**LM**) without the language model and prior probability model features. This shows the weakness of lexicalized reordering models, as used in phrase-based SMT. The model only learns how a phrase was translated with respect to their previous and next phrase, and makes independence assumptions over previously translated phrases. It does not take into account how the previous words were translated and reordered. The reordering model relies heavily on the language model to get the correct word order. In comparison, our system models the dependencies better by taking into account how previous tuples were translated and reordered.

System Settings

In this section, we study the behavior of our system by tweaking different parameters such as language model orders, stack sizes and number of translation options for each source cept.

Bilingual Data: 500K Sentences
 Monolingual Data: 1000K Sentences

Lang	Operation Model Order : Monolingual Language Model Order							
	2:2	2:3	3:3	4:4	3:5	5:5	7:5	9:5
DE	19.25	19.72	20.17	20.12	20.19	20.11	20.11	20.11
ES			22.07	22.27	22.06	22.16		22.23
FR				21.90	21.92	21.93	21.87	22.00

Table 4.25: Contribution of Different Feature Functions

Language Model Orders: Table 4.25 shows our findings from running experiments with different combinations of orders of monolingual language model and operation sequence model. From the results we found that using bilingual

4 A Joint Sequence Translation Model with Integrated Reordering

language model order for both the models hurts performance. We tried to use a tri-gram model for the monolingual language model as commonly used and a bigram model for the operation sequence model. This however, gives significantly worse results than our baseline system (OS_0). We found that using a trigram language model for German-to-English gives equally good results as our baseline system for German-to-English. For the Spanish-to-English task the results are slightly worse though and require 4-gram models. In case of French-to-English the results oscillate with a difference of +/- 0.1 BLEU point using 3-to-9gram order for the operation sequence model. These results are inconclusive apart from the fact that going below trigram model worsens the results. We therefore persisted with using 5-gram as the order of the monolingual language model and 9-gram as the order of the operation sequence model.

Bilingual Data: 1000K Sentences
 Monolingual Data: 2000K Sentences

Stack	German			Spanish			French		
Size	OS_0	Moses	Phrasal	OS_0	Moses	Phrasal	OS_0	Moses	Phrasal
50	19.86	19.99	19.88	22.79	23.33	23.76	21.62	21.92	21.88
100	20.06	20.15	19.90	23.05	23.38	23.81	21.78	21.96	21.89
250	20.20	20.17	19.92	23.34	23.33	23.79	21.79	21.96	21.89
500	20.36	20.17	19.94	23.47	23.32	23.78	21.81	21.95	21.89

Table 4.26: Translation Quality of Our System and Moses with different Stack Sizes

Stack Size: Table 4.26 shows results from running experiments with different stack sizes. Our findings are consistent with Costa-jussà et al. (2007) who report a drop in the performance of N-gram-based SMT when smaller beam sizes are used. Using higher stack size is essential when decoding with minimal units in order to produce larger phrasal units. Our results from running Moses and Phrasal with different stack sizes is consistent with (Koehn and Haddow,

2009) and other shared task papers that show that a stack size of 200 and above does not have a significant effect on translation into English.

		German	Spanish	French	
Bilingual Data:	1000K Sentences	5	19.78	23.13	21.78
Monolingual Data:	2000K Sentences	10	20.36	23.47	21.81
		20	20.13	23.35	21.82
		30	19.99	23.36	21.82

Table 4.27: Translation Quality of Our System Varying the Number of Translation Options

Translation Options: Table 4.27 shows our findings from running experiments by varying the number of translation options for a source cept. In the German-to-English translation task using a higher number of translation units worsens the translation score. In comparison the results remain stable in case of the Spanish and French-to-English translation tasks. This can be explained by the large amount of reordering required in German-to-English which makes the search space much more populated than that in French and Spanish where the source sentence is generated more or less monotonically.

4.8 Chapter Summary

In this chapter we comprehensively presented a novel model for statistical SMT which can be used as an alternative to phrase-based translation. The model inherits its fundamentals from N-gram-based SMT, in the sense that it is also a joint probability model and uses minimum translation units. However, we propose a better reordering mechanism that handles long distance reordering more effectively than the phrase-based and its predecessor N-gram-based model. Our model tightly couples translation and reordering into a single generative story. The generative story also provides a mechanism to handle discontinuous source-side cepts and unaligned source words through a source word deletion model. Our system, however can not handle target-side discontinuous cepts

and unaligned target units. We proposed a heuristic to overcome this problem. With the help of distance-based features, our model is able to correctly reorder words across large distances beyond a window of 6 words. Our system is based on minimal units but it can memorize frequent phrasal translations including their reordering as probable operations sequences. In the evaluation we found that our model outperforms the state-of-the art phrase-based and N-gram-based systems on the German-to-English translation task and achieves comparable results for Spanish-to-English and French-to-English.

5 Conclusion

In this chapter we present the contributions of the thesis. We revisit the pros and cons of the phrase-based and N-gram-based system discussing how our model is able to overcome the drawbacks of the previous models. At the end we present a section on shortcomings and future work.

5.1 Contributions of this Work

In this section we will list the contributions of this work, making a comparison with the phrase-based (Moses) and N-gram (Ncode) models.

5.1.1 Comparison with the Phrase-based System

First we will compare our system with the phrase-based. Let us begin by reiterating the drawbacks of the phrase-based model. See sections 2.3 for the details.

1. Dependencies across phrases are not directly represented in the translation model
2. Discontinuous phrases cannot be represented and used
3. Context independence assumption are made
4. The reordering model is not designed to handle long range reorderings
5. A hard reordering limit is applied during decoding

6. The presence of many different equivalent segmentations increases the search space
7. Source word deletion and target word insertion outside phrases is not allowed during decoding

Modeling of Non-Local Dependencies

Problem: Traditional phrase-based SMT models dependencies between words and their translations inside of a phrase well. However, dependencies across phrase boundaries are largely ignored due to the strong phrasal independence assumption. Recall the example shown in Figure 5.1. The reordering of the verb “abstimmen – vote” is internal to the phrase and therefore handled conveniently. However, the phrase-based system fails while translating the sentence “die menschen würden gegen meine außenpolitik abstimmen”. The phrase “gegen meine außenpolitik abstimmen – vote against my foreign policy” may not be available due to data sparsity. In this case, the phrase-based model is forced to fall back to word-based translation. The sentence will therefore be translated as “people would against my foreign policy vote” unless the language model provides strong enough evidence for a different ordering.

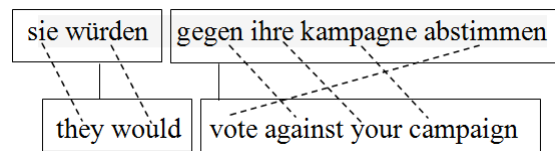


Figure 5.1: Handling of Local Dependencies – Dotted lines = Word Alignments

Contribution of this Work: The generation of this sentence in our model starts with generating “sie – they”, “würden – would”. Then a gap is inserted on the German side, followed by the generation of “abstimmen – vote”. At this point, the (partial) German and English sentences look as follows:

sie würden abstimmen

they would vote

We jump back to the gap on the German side and fill it by generating “gegen – against”, “ihre – your” “kampagne – campaign”, for the first example and generating “gegen – against”, “meine – my”, “außenpolitik – foreign policy” for the second example, thus handling both short and long distance reordering in a unified manner. Learning the pattern “würden abstimmen – would vote” helps us to generalize to the second example with unseen context, without relying on the monolingual language model.

Modeling of Gappy Units

Problem: A traditional phrase-based system can only use continuous phrases. If a discontinuous cept appears within a phrase, it can be learned and reproduced during decoding. However, such dependencies can not be handled across phrases. Recall the example shown in Figure 5.2. The phrase-based system fails to reproduce the translation unit “hat gelesen – read” (Figure 5.2(b)) although it has observed it during training (Figure 5.2(a)).

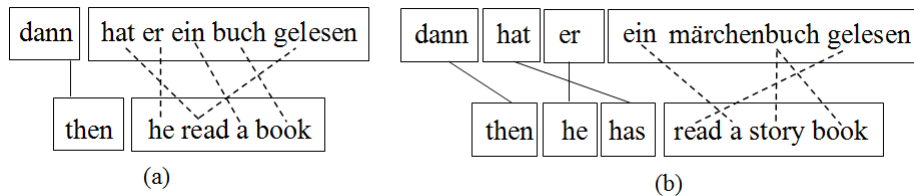


Figure 5.2: Handling of Gaps – Dotted lines = Word Alignments – (a) Learned Phrase (b) Unseen Context

Contribution of this Work: Our model can also use tuples with source-side discontinuities. The above sentence would be generated by the following

sequence of operations: (i) Generate(dann, then) (ii) **Insert Gap** (iii) **Generate(er, he)** (iv) **Jump Back(1)** (v) **Generate(hat gelesen, read)** (only “hat” and “read” are added to the sentences yet) (vi) Jump Forward (to the right-most source word so far generated) (vii) **Insert Gap** (viii) **Continue Source Cept** (“gelesen” is inserted now) (ix) **Jump Back(1)** (x) Generate(ein, a) (xi) Generate(buch, book).

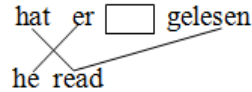


Figure 5.3: Learned Pattern

From this operation sequence, the model learns a pattern (Figure 5.3) which allows it to generalize to the example in Figure 5.2(b). The open gap represented by serves a similar purpose as the non-terminal categories in a hierarchical phrase-based system such as Hiero and Phrasal (discontinuous PBSMT). Thus it generalizes to translate “ein märchenbuch gelesen” in exactly the same way as “ein buch”.

Context Independence Assumption

Problem: Phrase-based SMT makes an independence assumption during translation. Each phrase is translated independently of others. Assume that the phrases “er hat – he has” and “hat gegessen – has eaten” appear in the bilingual corpus. The translation of a German sentence “er hat gegessen – he has eaten” is carried out through the phrase sequence “er hat – he has” and “gegessen – eaten” or through the phrase sequence “er – he” and “hat gegessen – has eaten”. In the former case the dependency of “gegessen – eaten” upon “hat – has” is not captured. In the latter case, the dependency that models subject verb relation is not captured. This problem also occurs in the discontinuous phrase-based SMT. Consider translating the sentence “er hat eine Pizza gegessen – he has eaten a pizza”. The translation could be done through two phrases “er hat X gegessen – he has eaten” and “eine Pizza – a pizza”. The dependency between the verb “gegessen – eaten” and the object “eine Pizza – a pizza” can not

be captured. This problem is somewhat reduced with the help of the language model but the model is only limited to what can be captured by the target-side LM.

Contribution of this Work: Our model like the N-gram model does not make any context independence assumptions other than the language model order. The model represents source and target words in operations:

Phrase 1 (Learned Bigram): Generate(er, he) Generate(hat, has)
 Phrase 2 (Learned Bigram): Generate(hat, has) Generate(gegessen, eaten)

During the decoding our model is able to capture both the dependencies, because our model can retrieve the bigram probabilities from the learned n-gram model.

Weak Reordering Model

Problem: The lexicalized reordering models used by the phrase-based system remain weak at modeling the long distance jumps and heavily rely on the language model to select the right word order. The model only learns how a phrase is reordered according to the last word of the previous phrase (Koehn et al., 2005) or the block of phrases (Galley and Manning, 2008). Recall the discussion and an example from Section 2.3.3 (Page 73). The orientation models learned from the following example are “sie würden – they would” (m,d)¹ and “abstimmen – against(d,d)”.

sie würden gegen ihre kampagne abstimmen

they would vote against your campaign

With the help of the learned orientations we can not distinguish the bad hypotheses from the good hypotheses shown in Table 2.8 (Page 76), when translating the German sentence given below.

¹Orientations m = monotonic, s = swap, d = discontinuous

5 Conclusion

sie würden für die Legalisierung der Abtreibung in Kanada abstimmen
they would for the legalization of abortion in Canada vote

The monolingual language model is required to break the tie. See Section 4.7.2 for comparison of our model with the phrase-based models without the language model feature. For longer sentences however, the monolingual language model is no longer able to compensate the dis-preference of the translation model for long distance reordering. The further to the right the word “abstimmen” is in the sentence the more difficult it is for the lexicalized reordering model to move it to the right position.

Contribution of this Work: The drawback of the lexicalized reordering model is that it does not model the connection between the words “würden – would” and “abstimmen – vote”. Our model on the other hand learns this relationship through the operation sequence:

sie würden abstimmen

they would vote

Generate(sie, they) Generate(würden, would) Insert Gap Generate(abstimmen, vote)

The reordering of “abstimmen – vote” is necessary to move the second part of the German complex verb to its correct position therefore gap insertion is a probable operation after the generation of “würden – would”. Having observed the above pattern in the training, the hypothesis that translates “sie würden...abstimmen” to “they would vote” would be more probable according to the operation model than any other hypothesis shown in Table 2.8 (Page 76). Our model takes more information into account than the lexicalized reordering model. The operation model strongly couples translation and reordering oper-

ations such that reordering operations are directly influenced by the preceding translation and reordering operations and vice versa.

5.1.2 Hard Reordering Limit

Problem: Phrase-based systems apply a hard reordering limit, allowing only a jump of 6 words. Using a higher distortion limit not only increases the decoding time but also increases the search errors resulting in a drop of translation accuracy.

Contribution of this Work: We do not apply any hard reordering limit during search. Our model with the help of reordering distance and gap distance penalties is able to differentiate the good hypotheses and filter out the bad ones. Using no reordering limit does not cause a decrease in translation accuracy in our system. On the contrary we observe slight gains in the German-to-English translation task which requires long distance jumps. See Section 4.7.2 for the translation accuracy of the phrase-based system without distortion limit in comparison to our system and for the discussion of results.

Spurious Phrasal Segmentation

Problem: Because phrasal SMT does not use minimal translation units during decoding, different compositions of the same phrasal units are hypothesized making the decoding slow. In practice spurious phrasal segmentation is not a problem because of the hard reordering limit, applied during decoding and data sparsity, which forces the decoder to use only smaller phrases during test. However, the removal of the hard reordering limit, which is ultimately desired, will cause a massive increase in the decoding time because of the multiple segmentations of the same phrasal unit being tried repeatedly with different hypothesis extensions. See Sections 2.3.5 and 3.5.1 for details.

Contribution of this Work: Our model, like the N-gram model, uses minimal translation units. The tuples, unlike phrases, do not overlap. Each bilin-

5 Conclusion

System	Phrasal	Phrasal_d	OS₀	OS_{ag}
German				
Test₁	1278661	6333837	130369	169026
Test₂	1286757	6494811	130410	169608
Spanish				
Test₁	1173039	4563726	140371	178968
Test₂	1174748	4648271	141800	181272
French				
Test₁	1360973	5305723	155850	205307
Test₂	1371261	5336184	157115	207627

Table 5.1: Total Number of Extracted Translation Units (Types) in Phrasal and Our System with both Continuous and Discontinuous Units

gual sentence, pair given the alignment, has exactly one possible operation sequence, thus a single segmentation. The number of translation units extracted by phrase-based SMT is many times more than that extracted by our model and the N-gram-based SMT. See Table 5.1 for a comparison of statistics on the extracted translation units in both systems from the two test sets of 1025 sentences. 20-best English translations are extracted for each source-side cept or phrase. The reported statistics are obtained from the translation model trained on 1M bilingual sentences.

Using minimal units, however presents a much more difficult search problem. 1) A potential hypothesis that would win in the later stages of decoding might be ranked quite low during the earlier stages of decoding and would be pruned out. 2) The future cost estimates are much worse using the minimal units, contributing towards the search errors. See Sections 3.5.3, 4.5 and 5.2.2 for details. A higher stack size is required to overcome this disadvantage. The efficiency gain thus obtained by using minimal units is therefore neutralized. See Section 4.7.2 for the search performance of our system on different stack sizes.

Source Word Deletion

Problem: Phrase-based systems handle deletion and insertion of words inside of phrases but do not allow these operations outside of phrases. See Section 2.3.6 (Page 79) for the details.

Contribution of this Work: Our model like the N-gram model provides tuple translations such as “ja – null”² where the source word “ja” gets deleted during decoding. This is done through the “Generate Source Only(X)” operation. The decoder can hypothesize a translation “Generate Source Only (ja)” when translating the German sentence “das Haus ist ja klein – the house is small”³.

The phrase-based model however has an advantage over our model. Although our generative story provides a “Generate Target Only(Y)” operation which can be useful for inserting target words during decoding, this operation is not supported by the decoder. On the other hand the phrase-based system can hypothesize phrases with target word insertions. Our system must use an attachment heuristic to achieve this effect. See Section 4.4 for details.

5.1.3 Comparison with the N-gram Model

In this section we will briefly go through the drawbacks of the N-gram-based SMT, making a comparison with our model. Below we enumerate the drawbacks of the N-gram model. The first three are related to the reordering framework used in the N-gram model. In this section we will discuss them in detail. Our model shares the last three drawbacks of the N-gram-based SMT so these are not discussed here. Refer to Section 3.5.4 for details.

1. Only the pre-calculated orderings are hypothesized during decoding.
2. The N-gram model can not use lexical triggers.

²“ja” is sometimes used for emphasis, and does not need to be translated in those cases.

³This example has been borrowed from Koehn (2010).

5 Conclusion

3. Long distance reorderings can not be performed.
4. Unaligned target words can not be handled without an attachment heuristic.
5. Target-side discontinuities are not represented
6. Using tuples presents a more difficult search problem than that in phrase-based SMT.

Pre-calculated Reorderings

The reordering framework of the N-gram model is triggered by source linearization and rewrite rules. The search graph used by the decoder is not built dynamically but based on the POS-based rewrite rules learned during training. The search is therefore performed only on the orderings pre-calculated on the source side independently of the target side. Often, the evidence for the correct ordering is provided by the target-side language model (LM). In the N-gram approach, the LM only plays a role in selecting between the pre-calculated orderings.

Inability to use Lexical Triggers

A related problem of the N-gram model is its inability to use lexical triggers. To hypothesize a reordering a rule must trigger. There can be a scenario where the linearized bilingual language model has the evidence of the reordering but there is no rewrite rule to hypothesize that reordering. Assume the training sentence:

Ich (I) habe (have) mit (with) ihm (him) gesprochen (talked)

I have talked to him

The linearization process changes the German word order to “Ich habe gesprochen mit ihm” and learns the reordering rule, IN PRP VBN \rightarrow 2 0 1. Now

consider a test sentence “Ich habe mit allen meinen freunden gesprochen”. The reordering of “gesprochen – talked” can only be hypothesized if a reordering rule “IN DT PRP NNS VBN \rightarrow 4 0 1 2 3” exists in the rule inventory. Although the bilingual corpus learns a tuple tri-gram “<Ich – I> <habe – have> <gesprochen – talked>” from the training example, the decoder does not hypothesize it for the test sentence because the rule is unavailable. Although the N-gram model uses POS tags, data sparsity still limits the usability of the rules.

Contribution of this Work: In comparison to the N-gram-based model, our model performs search on all possible reorderings. Our model has the ability to learn lexical triggers and apply them to the unseen contexts. For the above sentence the model learns a lexical trigger:

Ich habe gesprochen

I have talked

The model can jump back to the open gap and insert “mit ihm – to him” for the first example and “mit allen meinen freunden – with all my friends” in the test sentence. The inserted gap acts as a place holder for any number of words giving a greater flexibility to our model. The POS-based rewrite rules require exactly the same sequence of POS tags to appear in the training data.

Long Distance Reordering

The N-gram-based system uses a rule length of 6 or less POS tags. This prevents the N-gram model from hypothesizing long range reorderings which require larger jumps. Consider translating the German sentence shown in Figure 5.4. To move the German clause-final verb “stimmen – vote” to its correct position behind the auxiliary “would”, it needs to jump over 15 German words.

5 Conclusion

74% würden gegen die Studiengebühren, 79% gegen die Praxisgebühr, und 84% gegen das Krankenhaus-Taggeld stimmen

74% would vote against the tuition fee, 79% against the clinical practice, and 84% against the hospital daily allowance

Figure 5.4: Long Distance Reordering

Contribution of this Work: Our model does not apply any hard reordering constraint. We consider all possible reorderings comprising both long and short jumps. With the help of distance-based penalties our model is able to penalize the bad long distance reorderings.

5.2 Shortcomings and Future Work

5.2.1 Decoding

One of the major drawbacks of our model is the more difficult search problem. This problem is inherent to using minimal translation units during the decoding process. The phrase-based system can learn the phrase pair “wie heißt du – what is your name” and generate it in a single step placing it directly into the stack three words to the right. Our system generates this example with three separate tuple translations “wie – what is”, “du – your” and “heißt – name” in three adjacent stacks. In order to generate this phrasal unit, our system must use a higher stack size. Because “what is” is not the best candidate translation for “wie”, it will be ranked quite low in the first stack until the tuple “du – your” appears in the second stack. Higher stack size increases the decoding complexity.

Future Work

An interesting idea for future is to use phrase-based decoding instead of tuple-based decoding with our model. One simple idea could be to use the ngram model estimated from the operation sequence corpus as a feature in the discriminative model of phrase-based systems such as Phrasal or Moses. All the relevant features discussed in Section 4.3.3 can be added. Alternatively the existing decoder could be modified to use phrases instead of tuple units. Combining the ideas from the two models may yield fruitful results.

5.2.2 Future Cost Estimation

A related problem, caused by using smaller translation units, is the inaccurate future cost estimation. Using larger phrases helps phrasal SMT to capture dependencies such as language model, local reorderings etc. during the estimation of future cost. In comparison the future cost for tuples mostly comprises unigram probabilities. The future cost estimate for the phrase pair “wie heißt du – what is your name” is estimated by calculating the cost of each feature. The language model cost for example is estimated as:

$$p_{tm} = p(\text{what}) * p(\text{is} | \text{what}) * p(\text{your} | \text{what is}) * p(\text{name} | \text{what is your})$$

Translation model cost is estimated as:

$$p_{tm} = p(\text{what is your name} | \text{wie heißt du})$$

Phrasal SMT is aware during the preprocessing step that the words “wie heißt du” will be translated as a phrase. This is helpful for estimating a more accurate future cost because the context is already available. The same is not true for our model, to which only minimal units are available. Our model does not have the information that “wie heißt du” will be translated as a phrase during decoding. The future cost estimate available to the our model

5 Conclusion

for the span covering “wie heißt du” will have unigram probabilities for both the translation and language model.

$$p_{tm} = p(\text{what}) * p(\text{is} | \text{what}) * p(\text{your}) * p(\text{name})$$

The translation model cost is estimated as:

$$p_{tm} = p(\text{Generate}(\text{wie, what is}) * p(\text{Generate}(\text{heißt, name}) * p(\text{Generate}(\text{du, your}))$$

An accurate future cost estimation for our translation model cost would be:

$$\begin{aligned} p_{tm} = & p(\text{Generate}(\text{wie, what is})) * p(\text{Insert Gap} | \text{Generate}(\text{wie, what is})) \\ & * p(\text{Generate}(\text{du, your}) | \text{Generate}(\text{wie, what is}) - \text{Insert Gap}) \\ & * p(\text{Jump Back}(1) | \text{Generate}(\text{wie, what is}) - \text{Insert Gap} - \text{Generate}(\text{du, your})) \\ & * p(\text{Generate}(\text{heißt, name}) | \text{Generate}(\text{wie, what is}) - \text{Insert Gap} \\ & \text{Generate}(\text{du, your}) - \text{Jump Back}(1)) \end{aligned}$$

Thus the upper bound estimate given by the future cost estimate in our model is much worse as compared to that of the phrase-based model. The poor future cost estimation leads to search errors subsequently causing a drop in the translation quality.

Future Work

Phrase-based Decoding: Using phrase-based decoding on top of the joint sequence model, also improves future cost estimates. Notice that having to know that the sequence of words “wie heißt du” will be translated together help us to accurately estimate the cost of the language model and the operation model. However, using phrase-based decoding we reintroduce the spurious phrasal segmentation problem. The ambiguity however in our case, will only appear in the decoding process and not in the model (in contrast to PBSMT).

Spurious segmentation ambiguities make decoding more complex because more segmentations exist, but also simpler because the beam size can be reduced. A future research will unveil whether combining phrase-based decoding with operation sequence model is beneficial.

Multi-pass Decoding: Another idea to improve the future cost estimates is to use multi-pass decoding. In this framework the future cost is estimated from the states resulting from the initial pass, these estimates are then used in the next pass of decoding. The process can be repeated iteratively until the model scores improve. Multi-pass decoding with MERT training, however, can be computationally expensive.

5.2.3 Reordering Constraints

In this work, we used beam-search decoding with histogram pruning to constrain the search space. While it is a commonly used decoding mechanism to restrict the exponentially growing hypotheses, it is worth while trying other mechanisms to restrict the search space.

A future work can be to study the impact of applying restrictions such as ITG (inversion transduction grammars) constraints (Wu, 1997) or IBM constraints (Berger et al., 1996) to our model. The ITG grammars generate the input sentence as a sequence of blocks that can be merged monotonically or with a swap. IBM constraints, restrict the reordering to only one of the next k uncovered positions. These reordering restrictions can be applied by putting constraints on Jump Forward and Jump Back (N) operations in the model. The number of reorderings can also be controlled by putting hard limits on the total number of gaps and open gaps at a time.

5.2.4 Removing Target-Side Discontinuities

One of the drawbacks of our generative story is its inability to model discontinuous target-side cepts. Although our model can handle discontinuous source-side cepts such as “habe...gemacht – made” by inserting gaps and through

5 Conclusion

“Continue Source Cept” operation, the same can not be done on the target-side. The reason for this discrepancy is the assumption that English will be generated from left-to-right in continuous fashion. Our model can not represent alignment such as the one shown in Figure 4.15.

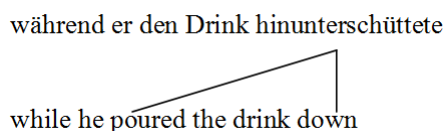


Figure 5.5: Target Side Discontinuities

Both continuous phrase-based and N-gram-based SMT shares this problem with our model. However, a phrase-based system can learn small phrases, such as “Drink hinunterschüttete – poured the drink down”, without applying the attachment heuristic.

In Section 4.4, we proposed a heuristic-based solution to this problem. We apply a post-processing that edits such alignments to make them consistent with our model. For example in this case we delete the alignment link “hinunterschüttete – down” and only retain “hinunterschüttete – poured”. However, a more elegant solution that modifies the generative story to represent target-side gaps is desirable.

Future Work

Idea-1: One possible solution to handle target-side discontinuities is to introduce an operation like `Generate(hinunterschüttete, poured [down])`, that will immediately generate “hinunterschüttete – poured” and place “down” on the queue. A follow up operation “Continue Target Cept” will dequeue “down” from the queue. This mechanism of handling discontinuous target-side gaps is used in Galley and Manning (2010). A potential drawback of this approach is that the conditioning of “Continue Target Cept” on its matching operation `Generate(hinunterschüttete, poured [down])` is weak, in the case where the dis-

tance between “poured” and “down” is more than 2 or 3 words, due to data sparsity.

Using Split Rules: A possible future work is to go along the lines of Crego and Yvon (2009) and to use split tokens on the target-side. We will therefore split a source-side cept X aligned with a discontinuous target-side cept into X_1 and X_2 , each aligning to one of the components of the discontinuous target cept. Figure 5.5 can therefore be transformed into Figure 5.6. A split rule such as “hinunterschüttete \rightarrow hinunterschüttete₁ hinunterschüttete₂ can be used during decoding.

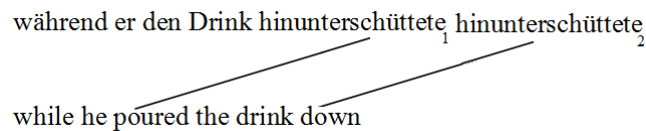


Figure 5.6: Target Side Discontinuities – Split Rules

Introducing Gaps on Target-Side: A more elegant solution would be to enable the generative story to handle this by removing the assumption of left-to-right generation and by introducing gaps and jumps on target-sides. This may however, overly complicate the generative story and will require parsing-based decoding.

5.2.5 Unaligned Target-Side Words

Another drawback in our work is that our system can not handle unaligned target words. Although our generative story can model unaligned target-side words through the “Generate Target Only (Y)” operation, the problem of how to hypothesize these words during decoding without increasing decoding complexity is a non-trivial problem. Both phrase-based and N-gram-based SMT share this problem with our model. However, phrase-based SMT can learn insertions of unaligned words inside of phrases. During decoding if such a word

appears in the same left or right context as observed during training, the phrase-based SMT can successfully hypothesize such units.

Future Work

Using Context for Clues: The idea is to enable “Generate Target Only (Y)” operation in the operation sequence model and use phrasal clues during decoding to hypothesize candidates of insertions. Currently when we see a case such as:

$$\langle A - a \rangle \langle NULL - b \rangle \langle C - c \rangle$$

we try to attach “b” to right or left tuple to form a new tuple $\langle A - ab \rangle$ or $\langle C - bc \rangle$ and learn an operation sequence such as “Generate(A,ab)” or “Generate(C,bc)”. Instead of doing this we now learn two phrases from this example $\langle A - ab \rangle$ and $\langle C - bc \rangle$ while maintaining this information that b is unaligned in these phrases. During decoding when A or C appears we build two hypotheses, one generating $\langle A - a \rangle$ with a “Generate (A,a)” operation and second generating $\langle A - ab \rangle$ with the operations “Generate (A,a) – Generate Target Only (b)”. This is equivalent to hypothesizing phrases “A – a” and “A – ab” in the phrase-based system. This method will work as long as A or C appear in the source sentence. A more sophisticated mechanism is required to insert target-side words with no contextual clues. It is arguable whether spuriously inserting target-side words with no contextual evidence may be a useful thing.

5.2.6 Removing Deficiency and Derivational Ambiguities

The model in its current form has two problems i) it is deficient and ii) it has derivational ambiguity. In this section we will briefly highlight these problems and abstractly discuss how these can be addressed.

Deficiency: One problem with the operation sequence model is that it assigns probability mass to some operation sequences that can not occur in practice. Following are some events where the model shows deficiency:

- The model can jump back with a non-zero probability even when there are no open gaps. Similarly model will assign probabilities to Jump Back(2) or Jump Back(3) although there is only one open gap.
- The model also assigns a probability mass to any “Continue Source Cept” operation that does not follow “Generate (X,Y)” operation with a multi-word source cept.

A reverse of this problem is caused due not having an explicit “Stop” operation when the generation ends. This leads to the probability of all operation sequences summing up to more than 1.

Derivational Ambiguity: A related problem is that several operation sequences can generate the same aligned sentence pair which results in a derivational ambiguity. The corpus conversion algorithm mentioned in Section 4.2.6 (Page 143) converts each bilingual sentence pair, given its alignment into a unique operation sequence. However, there is a many-to-one relationship on the reverse side i.e. more than one sequence of operations can be mapped to the same bilingual sentence pair. Secondly not every operation sequence can be mapped to a sentence pair. For example “wie heißt du - what is your name” can be generated with the operation sequence:

Generate(wie, what is) → Insert Gap → Generate (du, your) → Jump Back (1) → Generate (heißt, name)

but also with the following operation sequence:

Insert Gap → Jump Back (1) → Generate(wie, what is) → Insert Gap → Generate (du, your) → Jump Back (1) → Insert Gap → Jump Forward → Jump Back (1) → Generate (heißt, name)

Clearly the second operation sequence has an additional number of unnecessary jumps but the model currently does not have parameters to forbid such sequences.

5 Conclusion

Another example of such spurious generation can occur due to null-aligned words. Consider the example “kommen Sie mit – come with me” where “Sie” aligns with nothing. This example can be generated with the following operation sequence:

Generate (kommen, come) → Generate Source Only (Sie) → Generate (mit, with me)

but also with the following operation sequence:

Generate (kommen, come) → Insert Gap → Generate (mit, with me) → Jump Back (1) → Generate Source Only (Sie)

The model does not explicitly represent the information of when the null-aligned source word “Sie” has to be generated. Going in the other direction from alignments to operation sequence this restriction is encoded in the conversion algorithm which allows generation of “Sie” immediately after its previous German word is generated. There is a need for the elimination of many-to-one mappings from operation sequence to aligned sentence pairs.

Future Work

In order to remove the problem of deficiency and spurious derivational ambiguity, new parameters should be added to the model. These parameters would keep track of any open gaps. A “Stop” operation is required which would be disallowed if any “Continue Source Cept” operation is pending or if there are any open gaps that are required to be filled.

A future work will address the following open questions:

- What is the minimal set of restrictions required to eliminate inconsistent and non-canonical operation sequence?
- How does the operation sequence model need to be enriched (with additional context information) in order to be able to enforce these restrictions?

While having a model free from the problems mentioned in this section is desirable in theory, it is less likely to cause any improvements in machine

translation quality. An argument that can be used to support this notion would be the transition from IBM Model 4 to Model 5 (mentioned in Chapter 2, Pages 34 and 36). Model 4 is much more deficient than our model but Och (2002) has shown that Model 5, which solves the problem of deficiency in Model 4, does not improve the word-alignment or translation accuracies.

Secondly our model has a much richer conditioning than Model 5. In order to memorize lexical and reordering triggers, we have to remember previous $n - 1$ operations. Adding other parameters to the generative story will raise sparsity concerns.

5.2.7 Using POS-based and Lemma-based Operation Sequence Model

The inclusion of linguistic information such as morphology and syntax inside of machine translation has been shown to be useful. A tight integration of these into SMT has several benefits (i) Using POS-tags and lemmas are helpful to overcome data sparsity. (ii) Many phenomena in language translation can be best explained with morphological and syntactic evidence (Koehn and Hoang, 2007).

In this section we discuss some ideas about how to further improve the operation sequence model. One first step could be a POS-based extension of the model which allows the system to condition the probability of the next operation either on the sequence of preceding operations (as before), or on a generalized sequence of operations with words replaced by POS tags. The generalization of the operation sequence allows the system to consider a wider syntactic context where this is appropriate. A further extension would be to enable factored-based machine translation with our model. Below we sketch a possible road map for these ideas.

The lexical trigger learned from the bilingual sentence “er hat eine Pizza gegessen – he has eaten a pizza” is “hat gegessen – has eaten”. This pattern fails to apply on the test sentence “Wir haben eine Pizza fabriziert”. However, learning a reordering pattern such as “AUX VBN” based on POS-tags can

generalize to the test sentence “Wir haben eine Pizza fabriziert”. A lexically driven operation is conditioned on the k preceding operations with word pairs. The POS driven operation will be conditioned on k preceding operations with POS-tags. The n-gram language model for the POS-driven operations allows the generator to look further back, which will improve reordering.

The model estimated from the operation sequence of POS-pairs can be used as a feature in the discriminative framework. Alternatively it can be interpolated with the operation sequence model of lexical forms during decoding. A POS-driven translation bonus/penalty could be added which would effectively model the prior probability of POS-driven translation verses lexically driven translation.

Seeing the translation “auto – car” in the training data is not sufficient to produce the translation “autos – cars”. In order to generalize, we can switch to translation with lemmas and POS-tags. The generation can now be split into two steps. The first step source and target lemmas (auto,car). The probability of generating a lemma pair $p(\text{lemma}_{src}, \text{lemma}_{tgt})$ is conditioned on k preceding operations like before. The second step generates source and target words. The probability of the second step $p(\text{stem}_{src}, \text{stem}_{tgt})$ can be calculated by the method of maximum-likelihood estimates.

5.2.8 Additional Features

Linearized Source-Side Monolingual Language Model: Other supportive features could be added to the discriminative model to improve the reordering decisions made by the operation sequence model. One such feature can be a source-side monolingual language model. The idea is to linearize the source-side so that the word order is the same as on the target-side. Adding a linearized German-side language model in a phrase-based system has been shown to help (Feng et al., 2010).

Source-Side Syntax Feature: The operation sequence model is better than phrase-based SMT at long-distance reordering, but still fails to capture some

5 Conclusion

syntactic restrictions which are relevant for reordering. In order to improve the translator, source-side syntax information could be integrated. The source sentence is parsed. Whenever a jump (or sequence of jumps) occurs during translation, we compute the path between the start and end position of the jump in the parse tree. The path consists of the POS-tags of the source and target words, the direction of the jump, and the sequence of parse tree labels encountered on the traversal from the start to the end position of the jump. Other plausible ways of defining such jump features can be explored.

Bibliography

- Douglas Arnold, Lorna Balkan, Siety Meijer, R. Lee Humphreys, and Louisa Sadler. *Machine Translation: An Introductory Guide*. Blackwells-NCC, London, Great Britain, 1994.
- Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, and José B. Mariño. Statistical Machine Translation of Euparl Data by using Bilingual N-grams. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 133–136, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W/W05/W05-0823>.
- Satanjeev Banerjee and Alon Lavie. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43th Annual Meeting of the Association of Computational Linguistics (ACL-2005)*, Ann Arbor, Michigan, 2005.
- Srinivas Bangalore and Giuseppe Riccardi. Stochastic Finite-State Models for Spoken Language Machine. In *In Proceedings of the Workshop on Embedded Machine Translation Systems*, pages 52–59, 2000.
- Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, John R. Gillett, A. S. Kehler, and R. L. Mercer. Language translation apparatus and method of using context-based translation models. United States Patent, No. 5510981, 1996.
- Alexandra Birch, Chris Callison-Burch, and Miles Osborne. Constraining the Phrase-Based, Joint Probability Statistical Translation Model. In *Proceedings of the 7th Conference of the Association for Machine Translation of the Americas*, pages 10–18, 2006.
- Peter F. Brown, J. Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, F. Jelinek, John D. Lafferty, R. L. Mercer, and P. S. Roossin. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, 1990.

Bibliography

- Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.
- Ralf D. Brown. Example-Based Machine Translation in the Pangloss System. In *Proceedings of COLING*, Copenhagen, Denmark, 1996.
- Chris Callison-Burch, Colin Bannard, and Josh Schroeder. Scaling Phrase-based Statistical Machine Translation to Larger Corpora and Longer Phrases. In *In Proceedings of ACL*, pages 255–262, 2005.
- Daniel Cer, Michel Galley, Daniel Jurafsky, and Christopher D. Manning. Phrasal: A Statistical Machine Translation Toolkit for Exploring New model Features. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, pages 9–12, Los Angeles, California, June 2010. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/N10-2003>.
- David Chiang. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *Proceedings of ACL*, pages 263–270, Ann Arbor, MI, 2005.
- David Chiang. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228, 2007. URL <http://www.mitpressjournals.org/doi/pdf/10.1162/coli.2007.33.2.201>.
- Michael Collins, Philipp Koehn, and Ivona Kucerova. Clause Restructuring for Statistical Machine Translation. In *ACL05*, pages 531–540, Ann Arbor, MI, 2005. URL <http://www.aclweb.org/anthology/P/P05/P05-1066>.
- Marta R. Costa-jussà, Josep M. Crego, David Vilar, José A.R. R. Fonollosa, José B. Mariño, and Hermann Ney. Analysis and System Combination of Phrase- and N-Gram-Based Statistical Machine Translation Systems. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 137–140, Rochester, New York, April 2007. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/N/N07/N07-2035>.
- Josep M. Crego and José B. Mariño. Reordering Experiments for N-gram-based SMT. In *1st IEEE/ACL International Workshop on Spoken Language Technology (SLT'06)*, 2006.
- Josep M. Crego and José B. Mariño. Improving Statistical MT by Coupling Reordering and Decoding. *Machine Translation*, 20(3):199–215, 2006.

Bibliography

- Josep M. Crego and José B. Mariño. Syntax-Enhanced N-gram-Based SMT. In *Proceedings of the 11th Machine Translation Summit, MTsummitXI*, pages 111–118, 2007.
- Josep M. Crego and François Yvon. Gappy Translation Units under Left-to-Right SMT Decoding. In *Proceedings of the Meeting of the European Association for Machine Translation (EAMT)*, pages 66–73, Barcelona, Spain, 2009.
- Josep M. Crego and François Yvon. Improving Reordering with Linguistically Informed Bilingual N-Grams. In *Coling 2010: Posters*, pages 197–205, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL <http://www.aclweb.org/anthology/C10-2023>.
- Josep M. Crego, Marta R. Costa-Jussà, José B. Mariño, and José A. R. Fonollosa. Ngram-Based Versus Phrase-Based Statistical Machine Translation. In *Proceedings of the International Workshop on Spoken Language Technology (IWSLT05)*, pages 177–184, 2005a.
- Josep M. Crego, Adrià de Gispert, and José B. Mariño. The TALP Ngram-based SMT System for IWSLT’05. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, 2005b.
- Josep M. Crego, José B. Mariño, and Adrià de Gispert. Reordered Search and Unfolding Tuples for N-Gram-Based SMT. In *Proceedings of the 10th Machine Translation Summit (MT Summit X)*, pages 283–289, Phuket, Thailand, 2005c.
- Josep M. Crego, François Yvon, and José B. Mariño. Ncode: an Open Source Bilingual N-gram SMT Toolkit. *The Prague Bulletin of Mathematical Linguistics*, (96):49–58, 2011.
- Adrià de Gispert and José B. Mariño. TALP: Xgram-Based Spoken Language Translation System. In *Proc. of the International Workshop on Spoken Language Translation*, pages 85–90, Kyoto, Japan, 2004.
- John DeNero and Dan Klein. The Complexity of Phrase Alignment Problems. In *Proceedings of ACL-08: HLT, Short Papers*, pages 25–28, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P/P08/P08-2007>.

Bibliography

- George Doddington. Automatic Evaluation of Machine Translation Quality Using N-Gram Co-occurrence Statistics. In *Proceedings of the second international conference on Human Language Technology Research, HLT '02*, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. URL <http://dl.acm.org/citation.cfm?id=1289189.1289273>.
- Nadir Durrani, Hassan Sajjad, Alexander Fraser, and Helmut Schmid. Hindi-to-Urdu Machine Translation through Transliteration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 465–474, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P10-1048>.
- Minwei Feng, Arne Mauser, and Hermann Ney. A Source-side Decoding Sequence Model for Statistical Machine Translation. In *Conference of the Association for Machine Translation in the Americas 2010*, Denver, Colorado, USA, October 2010.
- Alexander Fraser and Daniel Marcu. Getting the Structure Right for Word Alignment: LEAF. In *Proceedings of EMNLP-CONLL*, pages 51–60, Prague, Czech Republic, 2007.
- Michel Galley and Christopher D. Manning. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/D08-1089>.
- Michel Galley and Christopher D. Manning. Accurate Non-Hierarchical Phrase-Based Translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California, June 2010. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/N10-1140>.
- Michel Galley, Spence Green, Daniel Cer, Pi chuan Chang, and Christopher D. Manning. Stanford University’s Arabic-to-English Statistical Machine Translation System for the 2009 NIST MT Open Evaluation, 2009.
- Adrià Gispert and José B. Mariño. Linguistic Tuple Segmentation in N-Gram-Based Statistical Machine Translation. In *INTERSPEECH*, 2006.

Bibliography

- Cyril Goutte, Kenji Yamada, and Eric Gaussier. Aligning Words Using Matrix Factorisation. In *Proceedings of ACL*, pages 502–509, Barcelona, Spain, 2004.
- Spence Green, Michel Galley, and Christopher D. Manning. Improved Models of Distortion Cost for Statistical Machine Translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 867–875, Los Angeles, California, June 2010. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/N10-1129>.
- W. John Hutchins and Harold L. Somers. *An Introduction to Machine Translation*. Academic Press, Cambridge, MA, 1992.
- Stephan Kanthak, David Vilar, Evgeny Matusov, Richard Zens, and Hermann Ney. Novel Reordering Approaches in Phrase-Based Statistical Machine Translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, Ann Arbor, MI, 2005.
- Kevin Knight. Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*, 25(4):607–615, 1999.
- Kevin Knight and Yaser Al-Onaizan. Translation with Finite-State Devices. In *AMTA*, pages 421–437, 1998.
- Philipp Koehn. Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *AMTA*, pages 115–124, 2004a.
- Philipp Koehn. Statistical Significance Tests for Machine Translation Evaluation . In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July 2004b. Association for Computational Linguistics.
- Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, The Edinburgh Building, Shaftsbury Road, Cambridge, 2010.
- Philipp Koehn and Barry Haddow. Edinburgh’s Submission to all Tracks of the WMT 2009 Shared Task with Reordering and Speed Improvements to Moses. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 160–164, Athens, Greece, March 2009. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W/W09/W09-0429.pdf>.

Bibliography

- Philipp Koehn and Hieu Hoang. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/D/D07/D07-1091>.
- Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*, pages 127–133, Edmonton, Canada, 2003.
- Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *International Workshop on Spoken Language Translation*, 2005.
- Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL 2007 Demonstrations*, Prague, Czech Republic, 2007.
- Chi-Ho Li, Hailei Zhang, Dongdong Zhang, Mu Li, and Ming Zhou. An Empirical Study in Source Word Deletion for Phrase-Based Statistical Machine Translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 1–8, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W/W08/W08-0301>.
- Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. Demonstration of Joshua: An Open Source Toolkit for Parsing-based Machine Translation. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 25–28, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P/P09/P09-4007>.
- Percy Liang, Ben Taskar, and Dan Klein. Alignment by Agreement. In *Proceedings of HLT-NAACL*, New York, 2006.

Bibliography

- Daniel Marcu and William Wong. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Proceedings of EMNLP*, pages 133–139, Philadelphia, PA, 2002.
- José B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert., Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-jussà. N-gram-Based Machine Translation. *Computational Linguistics*, 32(4):527–549, 2006. ISSN 0891-2017. doi: <http://dx.doi.org/10.1162/coli.2006.32.4.527>.
- I. Dan Melamed. Statistical Machine Translation by Parsing. In *Proceedings of ACL*, Barcelona, Spain, 2004.
- Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto, and Kazuteru Ohashi. A Clustered Global Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 713–720, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220265. URL <http://www.aclweb.org/anthology/P06-1090>.
- John A. Nelder and Roger Mead. A Simplex Method for Function Minimization. *Computer Journal*, 7:308–313, 1965.
- Franz J. Och. GIZA++: Training of Statistical Translation Models, 2000. <http://www-i6.informatik.rwth-aachen.de/~och/software/GIZA++.html>.
- Franz J. Och. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. PhD thesis, Computer Science Department, RWTH Aachen, Germany, 2002.
- Franz J. Och. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL*, pages 160–167, Sapporo, Japan, 2003.
- Franz J. Och and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003.
- Franz J. Och and Hermann Ney. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(1):417–449, 2004.
- Kazuteru Ohashi, Kazuhide Yamamoto, Kuniko Saito, and Masaaki Nagata. NUT-NTT Statistical Machine Translation System for IWSLT 2005. In *Proc.*

Bibliography

of the International Workshop on Spoken Language Translation, October 2005.

- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02*, pages 311–318, Morristown, NJ, USA, 2002. Association for Computational Linguistics.
- Helmut Schmid. Improvements in Part-of-Speech Tagging with an Application to German. In Feldweg, Hinrichs, Feldweg, and Hinrichs, editors, *Lexikon und Text*, pages 47–50. 1995.
- Jung H. Shin, Young S. Han, and Key-Sun Choi. Bilingual Knowledge Acquisition from Korean-English Parallel Corpus Using Alignment Method (Korean-English Alignment at Word and Phrase Level). In *Proceedings of COLING*, pages 230–235, 1996.
- Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. Translating with Non-contiguous Phrases. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 755–762, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/H/H05/H05-1095>.
- Anders Søgaard and Jonas Kuhn. Empirical Lower Bounds on Alignment Error Rates in Syntax-Based Machine Translation. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 19–27, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W09-2303>.
- Anders Søgaard and Dekai Wu. Empirical Lower Bounds on Translation Unit Error Rate for the Full Class of Inversion Transduction Grammars. In *IWPT*, pages 33–36, 2009.
- Andreas Stolcke. SRILM - An Extensible Language Modeling Toolkit. In *Intl. Conf. Spoken Language Processing*, Denver, Colorado, 2002.
- Christoph Tillman. A Unigram Orientation Model for Statistical Machine Translation. In *HLT-NAACL 2004: Short Papers*, pages 101–104, Boston, Massachusetts, 2004.

Bibliography

- Christoph Tillmann. A Projection Extension Algorithm for Statistical Machine Translation. In Michael Collins and Mark Steedman, editors, *2003 Conf. on Empirical Methods in Natural Language Processing*, pages 1–8, 2003. URL <http://www.aclweb.org/anthology/W03-1001.pdf>.
- Christoph Tillmann and Hermann Ney. Word Reordering and a Dynamic Programming Beam Search Algorithm for Statistical Machine Translation. *Computational Linguistics*, 29(1):97–133, 2003.
- Christoph Tillmann and Fei Xia. A Phrase-Based Unigram Model for Statistical Machine Translation. In *Proc. of the HLT-NAACL 2003 Conference*, pages 106–108, 2003.
- Christoph Tillmann and Tong Zhang. A Localized Prediction Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 557–564, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219909. URL <http://www.aclweb.org/anthology/P05-1069>.
- Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-Based Word Alignment in Statistical Translation. In *Proceedings of COLING*, pages 836–841, Copenhagen, Denmark, 1996.
- Warren Weaver. Translation. In W. N. Locke and A. D. Booth, editors, *Machine Translation of Languages: Fourteen Essays*, pages 15–23. MIT Press, Cambridge, MA, 1955.
- Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997.
- Omar F. Zaidan. Z-MERT: A Fully Configurable Open Source Tool for Minimum Error Rate Training of Machine Translation Systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88, 2009.
- Ying Zhang and Stephan Vogel. An Efficient Phrase-to-Phrase Alignment Model for Arbitrarily Long Phrase and Large Corpora. In *In Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT-05)*, pages 30–31, 2005.
- Ying Zhang, Stephan Vogel, and Alex Waibel. Integrated Phrase Segmentation and Alignment Algorithm for Statistical Machine Translation. In *Proceedings*

Bibliography

of International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE'03), Beijing, China, October 2003.

Bing Zhao and Stephan Vogel. A Generalized Alignment-Free Phrase Extraction. In *Proceedings of the ACL Workshop on Building and using Parallel Texts*, pages 141–144, 2005.