Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5B
D-70569 Stuttgart

Master thesis

Plug-and-Play Domain Adaptation for Neural Machine Translation

Emīls Kadiķis

Studiengang: M.Sc. Computational Linguistics

Prüfer*innen: Herr Prof. Dr. Roman Klinger
Herr Prof. Dr. Sebastian Padó

Betreuer: Yarik Menchaca Resendiz

Beginn der Arbeit: 23.03.2023
Ende der Arbeit: 23.09.2023

Erklärung (Statement of Authorship)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet. Die eingereichte Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Sie ist weder vollständig noch in Teilen bereits veröffentlicht. Die beigefügte elektronische Version stimmt mit dem Druckexemplar überein.[1]

(Emīls Kadiķis)

[1]Non-binding translation for convenience: This thesis is the result of my own independent work, and any material from work of others which is used either verbatim or indirectly in the text is credited to the author including details about the exact source in the text. This work has not been part of any other previous examination, neither completely nor in parts. It has neither completely nor partially been published before. The submitted electronic version is identical to this print version.

Acknowledgements

The journey of the past three years has been a rollercoaster with its ups and downs, endings and beginnings, learnings and challenges.

A big thanks goes to my supervisor Yarik Menchaca Resendiz and Roman Klinger for all the advice, suggestions and free reign I was given to explore this topic.

I couldn't have gotten here without the support my amazing friends and family. Thank you VB, Sof and Sven for the inspiration, discussions and all the sausages grilled together. Most of all, thank you Zeina for your unwavering support and pulling me up whenever I was down.

Abstract

Neural machine translation has emerged as a powerful tool, yet its performance heavily relies on training data. In a fast-changing world, dealing with out-of-domain data remains a challenge, prompting the need for adaptable translation systems. While fine-tuning is a proven effective adaptation method, it is not always feasible due to data availability, memory, and computational constraints.

This thesis introduces a dynamic plug-and-play method inspired by controllable text generation to enhance machine translation across various domains without fine-tuning. This method, called Plug-and-Play Neural Machine Translation (PPNMT), uses a mono-lingual domain-specific bag-of-words to push the hidden state of the decoder through backrpopogation, making the output more in-domain.

The method is tested on two types of domains: formality, gender (where the source language does not make a distinction between these aspects, but the target language does), and fine-grained technical domains (which are more based on topic inherent in the text on both the source and target sides).

The method performs reasonably well for adapting the translation to different formality levels and, to a lesser extent, grammatical genders, even with an incredibly simple bag-of-words. However, it struggles with adapting the model to technical domains, and a fine-tuning baseline outperforms the proposed method in anything but very low few-shot settings in all tried domains.

Despite that, the method shows some interesting behaviour, adapting to the formality on a level that goes beyond just using formal pronouns.

Contents

1 Introduction

Machine translation (MT) is bringing the world closer together. Although neural approaches have achieved excellent performance for specific language pairs, it highly depends on the amount of available training data (Fadaee et al., 2017). This is especially evident when dealing with texts outside the distribution of the training data that the MT model has seen in training (Koehn and Knowles, 2017), or put more succinctly when dealing with out-of-domain data.

This thesis explores a new method for improving machine translation performance on various domains with a dynamic plug-and-play approach that requires no fine-tuning.

1.1 Motivation

Human translators are masters of context. They use cues from various sources, modalities, and outside knowledge to pick the most appropriate translation for any given sentence. While MT systems have come a long way, they usually lack these context cues since end-users often interact with them through short, out-of-context sentences from websites, social media, or text chats (Vieira et al., 2023). Because of this, MT systems can make wrong guesses about the appropriate formality level to use or how to translate terms that have different meanings across different domains due to a lack of context.

While the most common MT use-case is translating whole documents at once (as reported by around 68% of users in the study by Vieira et al. (2023)), even with this context, the translation quality is dependent on whether the model has seen examples from this particular domain when training.

It is, therefore, necessary to adapt machine translation systems to new domains in a flexible way that is, ideally, adjustable by the end-users themselves since they usually possess the required context.

1.2 Problems with Domain Adaptation of Machine Translation Output

Various domain adaptation methods can be used to improve machine translation performance on specific domains. The most straightforward is fine-tuning the model on in-domain data (Luong and Manning, 2015; Xu et al., 2019). While this generally does achieve a good result, this approach has some issues.

1. **Fine-tuning is very data dependent** (Koehn and Knowles, 2017). For many language pairs (outside of the most common ones, e.g. English-French), the performance of machine translation models is constrained by the amount of available parallel data, with much effort put towards increasing available training data through data augmentation and unsupervised data mining (NLLB Team et al., 2022). Adding the extra constraint of the parallel data having to be of a certain domain only exacerbates the issue.

2. **It requires fine-tuning of the usually large MT model**. While fine-tuning is much less expensive than training the model from scratch, it is considerably more expensive than using methods that perform the domain adaptation at inference, such as the methods by Hasler et al. (2018) and Michon et al. (2020), which enforce terminology constraints during decoding.

3. **Fine-tuning can lead to catastrophic forgetting**, where the model forgets how to translate texts that are not within the domain of the data used in fine-tuning (Li et al., 2022). This can especially be an issue in a multi-domain setting (Saunders, 2022).

4. **Fine-tuning is not flexible**. When a user wishes to translate a domain-specific text, if the model has not already been fine-tuned on that domain, it is impossible to adapt to the user's needs quickly.

Some examples of MT applications that might require domain adaptation are translating scientific texts or technical specifications due to the specific language and terminology used in such texts differing from the average.

1.3 Style Control of Machine Translation Output

A related problem is controlling the style of the MT output. Some examples of style relevant to machine translation could be spoken language, formal/informal text, or detoxified text, with a particularly tricky example being poetry (Genzel et al., 2010) due to having to translate the rhyming scheme.

In contrast to domain adaptation, changing the style from the input to the output can be useful. For example, detoxifying the translation or changing the formality level while still keeping the core meaning of the translation intact. When translating from a language without different polite/impolite pronouns (such as English, which has the pronoun ”you”) into a language with different pronouns (such as German, which has the pronouns ”Sie” and ”du”), it is helpful to be able to control the desired formality level. This is the task of formality-sensitive machine translation, introduced by Niu et al. (2017).

1.4 Domain Adaptation and Style Control as a Combined Task

Domain and style are terms that are often used in different contexts. However, the two concepts are fundamentally linked. For instance, scientific text (domain) is usually formal (style). Because of this, this thesis will consider domain adaptation and style control as a single task, using the same method for both. All aspects that make some text have a different probability distribution than the training data will be referred to simply as ”domain”. Mino et al. (2020) gives credence to this kind of loose definition for practical purposes by including aspects such as data noisiness as one of the domains being adapted to.

Nevertheless, there is a useful distinction to make; in some cases, the source text is in a particular domain, and through domain adaptation, we wish to preserve the characteristics of this domain through the translation process. The domain is intrinsic to the meaning of the text. This is relevant for domains that deal more with the topic of the text (e.g. biomedical, engineering, finance), where we would want to

translate technical terms differently depending on the field they stem from. However, in other cases, we might wish to add some domain characteristics to the translation output that were not already present in the source text. The domain is extrinsic to the text. This is relevant for domains that deal more with style and register and linguistic features that differ between the source and target languages (e.g. choosing whether to translate an English sentence into formal or informal German).

While both cases are tackled with a single method in this thesis, they require slightly different approaches.

1.5 Proposed Method: Plug-and-Play Domain Adaptation

This thesis proposes a novel approach to this combined domain adaptation and style control task for MT. Instead of fine-tuning, a pre-trained neural translation model is used as-is, without any weight changes, and adapted to a specific domain in the decoding step through a small domain discriminator that guides the decoding process. In essence, the domain discriminator is used during the decoding step to nudge the pre-trained neural machine translation model's output closer to the desired domain.

This approach is inspired by controllable text generation methods, especially Dathathri et al. (2020), allowing greater control over the output since the domain discriminators can be switched out, combined, and weighted in a dynamic plug-and-play way. To keep it adjustable by the end user, we focus on straightforward domain models that define a domain with a bag-of-words and no trainable parameters.

This approach also circumvents the four aforementioned issues of fine-tuning. It does not require much data to create a domain discriminator, with the user being able to create the bag-of-words either manually or with minimal example data through methods such as tf-idf, nor does this data need to be parallel (since the adaptation happens purely during decoding, within the target language). Furthermore, since no fine-tuning is involved, catastrophic forgetting is avoided.

1.6 Research Questions

The key research questions build upon each other in a complementary manner. They are as follows:

- **Can the proposed plug-and-play method improve machine translation output on specific domains?**

  First, the efficacy of the proposed domain adaptation method must be established for different domains. This will be verified via automatic MT evaluation metrics and domain-specific approaches where viable.

- **How does the performance of this kind of plug-and-play domain adaptation method compare to traditional fine-tuning methods?**

  Second, the efficacy of the proposed method will be compared to a fine-tuning baseline, in which the same base model used for the plug-and-play adaptation will be fine-tuned on the available in-domain examples.

- **How does this method compare to fine-tuning when dealing with complex topic domains like technical literature?**

  While it is expected for the proposed method to work reasonably well on domains which deal more with style, it might struggle with complex topic-based domains, which could be hard to define with a bag-of-words effectively. The limitations of the proposed method will be investigated.

- **How does it compare to fine-tuning in low-data settings?**

  It is known that more training data leads to better fine-tuning performance. The proposed method, however, should be much less sensitive to the amount of training data. It will be evaluated at what amount of training data the fine-tuning approach starts outperforming the proposed method.

2 Background

This section lays the groundwork necessary to understand the task and methods used in this thesis. The more specific research this thesis builds upon, and the research directly adjacent to it can be found in section 3.

Modelling language is at the core of natural language processing (NLP). In fact, it is the implicit end goal of NLP because, with a good enough model of language, any NLP task that a human can do could be accomplished computationally. Because of this, this section begins with an overview of language models, starting from simple statistical methods to cutting-edge neural models. Their general architecture and specific use cases are discussed.

Next, the various methods of generating text from language models are considered, focusing especially on ways in which the output can be controlled to exhibit some desired characteristics, both in terms of the topic of the text and the style.

Lastly, machine translation is considered, starting from early rule-based methods and ending with neural models, which are commonly modelled similarly to language models. This thesis leverages this similarity between machine translation models and language models and uses methods created for controlling language model output to control the output of a translation model.

2.1 Language Models

Language models are a fundamental component in the field of NLP. They are used in various tasks ranging from text generation and machine translation to speech recognition. There are many different approaches to building language models, ranging from simple statistical methods to sophisticated neural approaches. Despite this, a language model at its core simply predicts the probability of a given sequence of tokens appearing in natural language.

This section starts by establishing what exactly they model, then explaining various types of simple mathematical models used for modelling language.

2.1.1 The Task of Language Modelling

Language models play a crucial role in NLP. Modelling the intricacies of how humans communicate through language is challenging since it involves many layers, such as morphology, syntax, semantics and pragmatics. Instead of trying to model all of these things explicitly, the task of language modelling is usually expressed as simply estimating the likelihood of the next word in a sentence (Jurafsky and Martin, 2023):

$$P(w_t|w_1, w_2, ..., w_{t-1}) \tag{1}$$

This can be expressed slightly differently to estimating the likelihood of a complete sentence appearing in natural language:

$$P(w_1, w_2, ..., w_t) \tag{2}$$

Both of these definitions are effectively equivalent. Which one to use is more of a question of framing. For example, the first definition is more appropriate when generating text word-by-word. However, the second one is more appropriate when re-ranking possible translations from a statistical machine translation model.

While this simple definition does not truly capture the intricacies of human language, it is shown to be practical for many use cases such as machine translation (Brants et al., 2007) and speech recognition (Jelinek, 1985) among others.

2.1.2 Tokens

Language models work with discreet units of language called tokens. There are different ways of splitting up text into tokens with different advantages and disadvantages.

Whole words as tokens In the simplest case, one token could equal one full word. This approach is straightforward, but it has some drawbacks.

- There needs to be a token for each variation of a word. This is especially inefficient for highly agglutinative languages, where many variations of the same root word exist.
- The model does not know anything about words that are not in the vocabulary, even if they are derived from words that it does know.
- Rare words can be under-represented in the data from which the model is created, leading to imprecise probability estimations for sequences with these rare words.
- For languages with no spaces between words, it can be hard to determine what to count as a whole word. One example is Chinese, for which many segmentation systems have been developed even recently (Huang et al., 2020; Ma et al., 2018; Tian et al., 2020).

Single characters as tokens On the other extreme of the spectrum, each separate character could be counted as a token.

This approach has some advantages:

- The vocabulary is much smaller (since there are orders of magnitude more words in a language than characters)
- Out-of-vocabulary words are practically impossible (besides text where special symbols are used or there is code-switching to a language which uses a different writing system). Even misspellings are handled.

However, there are some critical disadvantages:

- Sequence length explodes even for short sentences. This can pose a problem for keeping long-term consistency in the output.

- Each token usually does not carry enough meaning on its own. This complicates text generation because the model needs more foresight to predict the meaning of a longer sequence of tokens.

Despite this, there are languages where a character-based tokenization scheme is effective. For example, Chinese has thousands of different characters, each carrying much meaning on its own.

Sub-word units as tokens While whole-word and single-character tokenization approaches have their merits, they can fall short in capturing the nuances of languages with complex morphologies or when dealing with out-of-vocabulary words. A common approach is sub-word tokenization, which breaks words down into smaller units, offering a more granular representation that strikes a balance between word and character-level information.

There are many ways in which a word may be split into sub-word units. Expert linguistic knowledge may be used. For example, prefixes and suffixes could be split off as their own sub-word units. Compound words can be split into separate word roots. Words could be split into morphemes.

More commonly, however, a data-driven approach is used. This introduces the concept of open-vocabulary compared to closed-vocabulary. Closed-vocabulary systems rely on expert-defined word lists, limiting the model's ability to handle out-of-vocabulary words effectively. In contrast, open-vocabulary approaches are data-driven, allowing the model to create subword units dynamically based on the training data, thereby accommodating a more comprehensive range of words and expressions.

However, the choice between larger and smaller vocabulary sizes involves a trade-off. A more extensive vocabulary enhances the model's capacity to represent a diverse range of words and handle variations as well as misspellings. However, it comes at the cost of increased computational complexity and memory requirements. In contrast, a smaller vocabulary simplifies computations but may struggle with handling infrequent or out-of-vocabulary words. Striking the right balance between vocabu-

lary size and computational efficiency is a crucial consideration in designing effective language models (Xu et al., 2021).

One popular technique for subword tokenization is **Byte Pair Encoding (BPE)**. It is a compression algorithm (Gage, 1994) that was first applied to text tokenization by Sennrich et al. (2016c). BPE operates on the principle of iteratively merging the most frequent character pairs in a corpus to create new subword units. Initially, each character in the vocabulary is treated as a subword unit. Through a process of successive mergers, the most frequent character pairs are combined, forming new subword units that can represent frequently occurring sequences of characters. This can be repeated until a desired vocabulary size is reached. This method effectively captures both common words and the constituents of longer words, making it ideal for languages with agglutinative structures or complex morphologies. Instead of relying on linguistic knowledge, the algorithm is able to capture linguistic patterns from the data.

A similar technique is **WordPiece tokenization** (Schuster and Nakajima, 2012), which, instead of merging the most frequent token pairs, only merges two tokens if doing so would increase the likelihood of the training data.

Both of these methods have a drawback, however. They require that the text is split into words beforehand, making them not applicable to languages such as Chinese. **SentencePiece** (Kudo and Richardson, 2018) solves this with a fully end-to-end system which treats the input as a continuous stream of characters with no pre-tokenization step required. It is universally applicable to all languages, making it a popular choice for tokenization in the context of NLP.

2.1.3 Statistical Language Models

While language models made with hand-crafted rules do exist, they are generally confined to specialized use cases (speech recognition (Kaufmann, 2009), spoken dialogue systems (Williams et al., 2010)) and not general language modelling, since natural language is too complex and too creative for such a limited approach to be practical for general language modelling. As an alternative, a significant proportion

of language models are constructed using statistical approaches, leveraging massive text corpora for training.

Early models took a pragmatic approach by employing a Markov assumption, which assumes that the probability of a word solely depends on a limited number of preceding words. This assumption simplifies the task and makes it computationally feasible. However, it comes at the cost of ignoring long-range dependencies that are crucial in capturing the intricacies of language.

While later neural approaches are also statistical at their core, this section deals only with the early methods. Due to the importance of neural methods, a separate section is devoted to them.

N-gram models The simplest language models are n-gram models. They are a class of statistical language models that are based on token occurrence. While any of the tokenization schemes discussed in the preceding section could be used, for simplicity, all examples will assume that one token is equal to one word. The "n" in n-gram refers to the number of words in a sequence that the model considers, and the model calculates the probability of encountering a particular word given the previous (n-1) words.

The simplest case with n=1 is a unigram model, which looks at the probabilities of individual tokens. It assumes that each word is entirely independent of the context, thereby neglecting any information carried by the words that precede or follow it. This model is rudimentary yet forms a foundation for more complex n-gram models. The probability of a sequence of tokens would then be the multiplication of the probabilities of each separate token.

$$P(W_1, W_2 ... W_k) \approx \prod_{i=1}^{k} P(W_i) \tag{3}$$

The probability of each token can be approximated by using word occurrence counts through a maximum likelihood estimate. If a text corpus has N words in

total, then to determine the unigram probability of some word w, we just need to divide the count of times w appears by the total word count N:

$$P(W) = \frac{C(w)}{N} \tag{4}$$

This has the obvious drawback of completely ignoring word order. It would give the same probability to the sentences "the fox jumped over the dog" and "jumped dog the over fox the".

With n=2, a bigram model is formed, which takes into account one preceding word. The probability of a sequence of words is then approximated as follows:

$$P(W_1, W_2 ... W_k) \approx \prod_{i=2}^{k} P(W_i | W_{i-1}) \tag{5}$$

This definition can be extended to any n. The more preceding words are taken into account, the more accurate the probability estimate becomes. However, the size of the corpus needed to extract a meaningful amount of n-gram counts increases exponentially the higher n is. The largest n-gram dataset is the Google Web 1T 5-gram (Web1T5) corpus from Brants and Franz (2006) created from a trillion tokens of web content and containing over a billion five-grams. However, even that is not enough to cover all reasonable five-grams that could conceivably appear in natural language.

Skip-gram Language Models Skip-gram language models are an extension of n-gram models built to deal with data sparsity. Instead of assuming that all n-grams must be contiguous tokens, it allows some distance between the words. More specifically, a k-skip-n-gram model would work with sequences of length n where the tokens can be at most k tokens away from each other.

Dealing with unseen n-grams When faced with n-grams that do not appear in the corpus used to construct the model, a basic n-gram model assigns a probability

of 0. This is not desirable behaviour; since the probability of the whole sequence is defined as the multiplication of the probabilities of each separate n-gram, if one n-gram gets assigned a probability of 0, the probability of the whole sequence becomes 0. Different strategies exist to deal with this.

A common technique employed in n-gram models to address this is smoothing. Generally speaking, smoothing methods shift some of the probability mass from common n-grams to unseen n-grams, ensuring all n-grams get assigned a non-zero probability. These methods increase the overall robustness and reliability of n-gram language models, enabling them to handle more varied ranges of sequences.

The simplest smoothing method is **Laplace (add-one) smoothing**, which adds a constant value (usually 1) to all n-gram counts, making sure that every n-gram is seen at least once. To ensure a valid probability distribution, this constant is also added to the denominator multiplied by the n-ram vocabulary size V:

$$P_{Laplace}(w) = \frac{C(w) + 1}{N + V} \tag{6}$$

To tackle the issue of unseen n-grams, several widely used smoothing techniques have been developed. Laplace (add-one) smoothing involves adding a constant value (usually 1) to both the numerator and denominator of the n-gram probabilities. This ensures that each n-gram has a non-zero probability, albeit at the cost of redistributing probability mass from observed n-grams.

The generalization of Laplace smoothing, which adds some other small value to the n-gram counts, is called **add-k smoothing**. This provides more flexibility, as the degree of smoothing can be adjusted by changing k.

These kinds of smoothing methods are blunt tools that significantly shift the probability distribution. **Kneser-Ney smoothing** (Kneser and Ney, 1995) is a more sophisticated approach, which considers the context in which n-grams occur. It involves calculating a discounting factor that accounts for the frequency of the context in which an n-gram appears. This discounting factor adjusts the probability estimates for unseen n-grams based on the prevalence of their contexts. By

incorporating contextual information, Kneser-Ney smoothing achieves more accurate probability estimates and captures more nuanced language patterns.

Another type of smoothing method is to combine multiple different n-gram models. For instance, a **Katz back-off** model (Katz, 1987) backs off to a lower-order n-gram model if a particular n-gram is unseen. For example, if a trigram is unseen, it would use the bigram probability instead, and if the bigram is unseen, the unigram probability.

A similar approach, **interpolation smoothing**, always uses all different n-gram models to calculate a weighted average of probabilities from lower and higher-order n-grams. Both of these methods ensure that all n-grams are given a non-zero probability because, in the worst case, the unigram probability will be used. With a closed vocabulary, all unigrams have some non-zero count (and with an open vocabulary, unseen words can be replaced with some special "$\langle unknown \rangle$" token).

2.2 Neural Language Models

The advent of deep neural networks has revolutionized many different fields, including NLP. While the concept of neural networks, which was based on a rudimentary model of human neuronal function, originated in 1943 (McCulloch and Pitts, 1943), it has held the attention of researchers only intermittently, falling in and out of popularity. Recently, the method has had a massive resurgence with the advances in parallel computation hardware, allowing for the training of deep neural networks with many layers.

This section will explain the basics of neural networks, how they can be structured and how they are trained. Then, their various usages in language modelling will be explained.

2.2.1 Neural Network Basics

The basic unit at the core of all neural networks is a single "neuron" which takes some inputs $\mathbf{x}$ in the form of a vector, calculates the dot product with a weight

vector $\mathbf{w}$, adds a scalar bias b and transforms the output via the use of a non-linear activation function f.

$$y = f(\mathbf{x} \cdot \mathbf{w} + b) \tag{7}$$

Layers of Neurons These neurons are usually grouped and arranged in distinct layers, each passing on their outputs to the next layer as inputs. The specifics of how the layers are connected change depending on the type of architecture being used. However, there is always some entry point into which the initial input is fed and an ultimate output. In a sense, this process can be likened to the way human perceptions about the world are processed by the brain. The first layer receives input from the sense organs. Each subsequent layer refines the initial input, gradually extracting more abstract and complex features. Finally, the output of the last layer encapsulates the network's understanding and transformation of the input, yielding task-specific results.

Input and Output The input and output are numerical due to the nature of the underlying operations the neural network performs. This means that some feature engineering is required to extract numerical features from data of different modalities. Depending on the data, different kinds of input features could be used: handcrafted features, RGB values of the pixels in an image, samples from a waveform, etc. Similarly, the output will also vary based on the nature of the task. For regression tasks, the output might be a continuous value, predicting quantities like stock prices or temperature. For classification tasks, the output could be a probability distribution over each of the possible classes, indicating the likelihood of the input belonging to each class. In sequence-to-sequence tasks, such as machine translation or text generation, the output could be a sequence of words. The power of neural networks lies in their capacity to automatically learn how to map from the input to the output. Through training, these networks learn to extract relevant features and patterns from the data, eliminating the need for explicit feature engineering and allowing them to generalize to unseen examples.

Activation Function The non-linear activation function is a critical component. Without it, neural networks would only be able to calculate linear operations, and a network with multiple layers would have the same expressive power as a single-layered network. There exist many activation functions. Some of the more popular ones are sigmoid, tanh, and ReLU (rectified linear unit). The non-linearity they introduce allows the network to learn arbitrarily complex functions, making them a universal approximator. (Cybenko, 1989)

Trainable Parameters The weights and bias are the part of each neuron that gets adjusted during training, thereby representing the model's learned knowledge. Generally speaking, the more trainable parameters the model has, the more complex knowledge it is able to capture. On the other hand, more parameters also mean more effort is required to train the model.

Training Neural Networks The knowledge of neural networks is hidden in the weights of the individual neurons. These weights are extrapolated from data through a training process. The goal of training at its simplest is, given pairs of input and expected output, to minimize the difference between the expected output and what the model actually outputs given the input. The observed difference is used to nudge the weights in a direction which minimizes the difference. This process is called supervised learning.

This is accomplished via a loss function, which quantifies the difference between the expected and actual output, or in other words, the error. This error is back-propagated (Rumelhart et al., 1986) to the weights via partial derivatives to calculate the gradient of each weight with regard to the loss function. Effectively, the gradient of each weight quantifies how large of an impact that particular weight had on the final loss. The weights are then updated using the gradient. This process is repeated until the loss converges.

2.2.2 Embeddings

In the previous section, the concept of input features was introduced. The data that NLP deals with is almost entirely textual, so in order to be used in a neural network, it needs to be converted into numerical values in some way. Going back to the concept of tokens introduced in section 2.1.2, one possibility is to assign an integer value to each token. However, representing words merely as integers does not capture any of the inherent semantic relationships. Two synonyms, while interchangeable in the input sentence, would have different integers assigned to them, so the model would process them as entirely different words.

This limitation can be solved via embeddings. Embeddings are a way to represent words in a continuous vector space, where semantically related words are positioned closer to each other. A model using these embeddings as input would then have more semantically meaningful features readily available to them and would see two synonyms as very similar due to their similar embeddings.

The vector spaces used for embedding are high-dimensional, often reaching hundreds or even over a thousand dimensions. Conceptually, each dimension is its own input feature, which could be interpreted as some semantic property of the word, for example, how ”male” or ”female” something is. The words ”king” and ”queen” would then have very similar representations, differing only in this one dimension. In practice, embeddings are usually learned from data computationally, so these dimensions are not so easily interpretable. Nevertheless, some interesting operations are possible in these vector spaces (Ethayarajh et al., 2019). A famous example is the following:

$$\vec{king} - \vec{man} + \vec{woman} \approx \vec{queen} \tag{8}$$

There are different approaches to creating word embeddings, which will be looked at in more detail in the following paragraphs.

Embeddings Through Global Matrix Factorization An early approach by Deerwester et al. (1990) uses singular-value decomposition to produce a semantic space where terms and documents that are closely associated are placed near one another. Broadly speaking, this kind of method starts with a term/document incidence matrix and decomposes it into three separate matrices, which produce the original incidence matrix when multiplied together. One of these matrices contains the calculated word embeddings.

The authors use this method to improve information retrieval performance by allowing users to find documents by querying for semantically similar terms that do not appear in the document.

Distributional Word Embeddings Distributional semantics is at the core of many word embedding approaches. It is the idea that words that appear in similar contexts have similar meanings. Using this idea, it is possible to extract semantically meaningful embeddings purely based on what other words appear within a limited context window around words. With a large enough corpus of text, meaningful embeddings may be extracted. With this approach, words that appear in similar contexts have similar embeddings in vector space.

Bengio et al. (2003) propose having an embedding layer in their feedforward language model (discussed in section 2.2.3), which learns how to represent each word as a vector jointly with the task of language modelling.

One of the more well-known later attempts at creating distributional word embeddings through machine learning is Word2Vec, proposed by Mikolov et al. (2013). They propose two different architectures that work with a limited context window: a continuous bag-of-words model, which takes one word as input and learns to predict what words might appear around it in the window, and a continuous skip-gram model, which takes the context words as input and learns to predict the word that might appear in that context. This approach is computationally effective enough to allow the authors to train the embeddings on a substantial amount of text, at the time achieving state-of-the-art performance on a syntactic and semantic word similarity task.

A later approach by Pennington et al. (2014), instead of using a limited context window, uses global word co-occurrence. This increased context size helps the embeddings capture more complicated dependencies.

The weights learned for each word by these models can be extracted as word embeddings and re-used in many different NLP tasks. This frees up task-specific models from having to learn their own word embeddings and allows them to focus on just the specific task at hand.

Contextual Word Embeddings The previously mentioned embedding methods have a crucial flaw. They all give a single embedding vector for each word, mixing together all the possible senses and contexts in which the same word might be used differently.

Contextual word embedding models such as Elmo (Peters et al., 2018), Flair (Akbik et al., 2018), and context2vec (Melamud et al., 2016) all employ bi-directional recurrent models in different ways, taking in the entire sentence as input and returning embeddings for words in that specific context. The later transformer-based (discussed in section 2.2.8) embedding model by Devlin et al. (2019) represents a state-of-the-art method for embedding language that is still used today.

These kinds of embeddings prove to be more effective for various downstream tasks. Moreover, as noted by Smith (2020), in hindsight, calculating contextual embeddings is also a more straightforward task because we no longer need to capture all possible contexts in which a word might appear.

2.2.3 Feed-Forward Neural Networks

The most basic arrangement is a feedforward network where all neurons from one layer are connected to all the neurons on the next layer. The first and the last layers are called the input and output layers, respectively, with the intermediate layers referred to as hidden layers. In a feedforward network, information flows unidirectionally from the input layer to the output layer.

Feed-Forward Language Models The simplest way to use neural networks for language modelling is inspired directly by n-gram models. Feedforward language models take the preceding $n-1$ tokens as input and calculate the probabilities of each of the tokens in the vocabulary to be the next one in the sequence. (Bengio et al., 2003)

Just like n-gram models, these neural models use minimal context to calculate the next word. As the name implies, they are built with feedforward networks and do not use any sort of recurrence. They are, however, an improvement over simple n-gram models as they work with word embeddings, and so handle out-of-vocabulary words and semantically related words much more effectively.

2.2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of neural network especially well-suited for tasks with grid-like data, such as computer vision. CNNs, unlike feedforward networks, do not connect all neurons of one layer to all neurons of the next layer. Instead, a smaller two-dimensional kernel with shared weights is used to calculate the output over localized areas of the input data. This kernel's sliding window operation, also known as convolution, captures local patterns and features with significantly fewer parameters than feedforward networks (LeCun et al., 1995).

Convolutional Language Models Although CNNs are primarily used in computer vision, they have shown some promise for NLP as well, especially in tasks which only require a limited contextual window. In these models, the convolution operation is often applied not on two-dimensional input but on a one-dimensional sequence of tokens. The shared weights in the convolutional kernels allow the model to efficiently learn and recognize patterns across different parts of the input sequence using fewer parameters compared to traditional feedforward neural networks.

Collobert and Weston (2008) create a CNN which outputs many different language processing predictions such as part-of-speech tags, chunks, named entity tags,

semantic roles, semantically similar words and the likelihood that the sentence makes sense (language modelling).

A notable advantage of convolutional language models is their ability to capture hierarchical features at different scales. By stacking convolutional layers with varying kernel sizes, the model can look both at fine-grained details and broader contextual information, allowing different layers to focus on different aspects of language (Pham et al., 2016).

Furthermore, in comparison to recurrent approaches (which are introduced in the next section), CNN-based language models are more parallelizable and so are faster to train Gehring et al. (2017).

Time-Delay Neural Networks An even earlier approach, which came before CNN's but shares striking similarities with the way they use convolutions to have different layers look at different time scales, is time-delay neural networks by Waibel et al. (1989). They use this method for phoneme recognition, where some surrounding context is helpful for disambiguation, but no long history is required.

2.2.5 Recurrent Neural Networks

Both of the previously mentioned network types are not well suited for dealing with time-series data. This is because they lack a crucial element: memory of past inputs. In many real-world tasks, including natural language processing, the sequence of events is vital to interpreting the data correctly. This memory can be implemented through recurrence. There are different approaches to achieving this.

Base Recurrent Neural Networks Recurrent neural networks (RNNs) are specifically designed for sequential data. While the concept has earlier roots in neurology, in the context of neural networks, recurrence as we know it was first proposed by Jordan (1997). Unlike feedforward networks and convolutional networks, RNNs have a loop in their architecture. Each input is passed sequentially, with the hid-

den layer also incorporating the hidden state from the previous time step, thereby allowing the model to keep some internal memory.

This memory, however, is limited by various factors. Firstly, the size of the vector, which keeps the internal state, has a finite size. Secondly, when these models are trained through backpropagation in time, they are unrolled into what are essentially deep neural networks, with the earlier time steps being deeper layers. Deep networks exhibit what is known as a vanishing gradient problem, where the gradients used to adjust the model's weights get smaller and smaller the deeper you go. Because of this, it is difficult to learn from distant past inputs and, hence, to model long-term dependencies. (Werbos, 1990)

Long Short-Term Memory Networks Long short-term memory (LSTM) models are a type of recurrent neural network designed to deal with the vanishing gradient problem. Instead of indiscriminately passing the hidden state from one time step to the next, they introduce a memory vector that is selectively adjusted at each time step according to a forget gate (which clears values in the vector, forgetting unnecessary information) and an input gate (which adds values, remembering new information). An output is then generated using the current input and the memory vector through an output gate. (Hochreiter and Schmidhuber, 1997)

While this approach helps deal with vanishing gradients, LSTMs have more trainable parameters (each of the gates requiring its own set of weights), making them more time-consuming to train. Furthermore, their ability to model long-term dependencies is still limited by the size of the vector used as the memory.

Gated Recurrent Unit Gated recurrent units (GRUs) proposed by Cho et al. (2014a) simplify the LSTM mechanism. It has fewer parameters as it drops the output gate. This means that the output at each time step is the same as the hidden state, which imposes some limitations on this architecture.

Recurrent Language Models RNNs can be used for language modelling, generally outperforming even the largest statistical models (Mikolov et al., 2010). Unlike

feedforward language models and convolutional language models, recurrent ones have an unlimited context window. One of the main limitations of simpler RNN models is that the output size is tied to the input size. This is solved through more sophisticated architectures, such as the encoder-decoder, described in the following section.

2.2.6 Encoder-Decoder Architecture

The recurrent language models mentioned in the previous section have a limitation. For each input token, it outputs exactly one output. This is workable for tasks where the input and output are the same size, for example, for token classification tasks such as named entity recognition, part of speech tagging and so on, where exactly one label is output for each token. For most NLP tasks, the size of the output needs to be decoupled from the size of the input. In such sequence-to-sequence tasks like text summarization and question answering, this is obvious; however, even for machine translation, a single token on the source side may often require multiple tokens to express on the target side.

Encoder-decoder architecture is a solution proposed by Cho et al. (2014b) for machine translation, which decouples the input from the output. Instead of having one recurrent network that processes each input token and produces a corresponding output token, an encoder-decoder model is split into two separate RNN models: an encoder and a decoder. The encoder first processes all of the input tokens without outputting anything. This is done to calculate the hidden state vector, which, in a way, encodes the input sentence. This hidden state vector is then used to initialize the hidden state of the decoder model, which outputs tokens until a particular end token is output, signifying that the decoder is done outputting. As the first input, the decoder receives a special start token and takes its own output from the previous time step as its input at each subsequent step.

Conceptually, the encoder can be seen as a model producing sentence embeddings. The embedding space is jointly learned to capture the aspects of the input that are relevant to the task at hand and then decode from this embedding space

into an output.

This approach has a very simple limitation: the size of the hidden state vector is finite. As shown by Conneau et al. (2018), how much and what kind of information this vector captures is highly dependent on the model architecture, among other factors. This means that there is a bottleneck through which the input, no matter how long, has to be pushed through. A solution for this bottleneck, namely attention, is proposed in the following section.

2.2.7 Attention

The concept of attention finds its roots in word alignment in machine translation. Word alignment, explained in more detail in section 2.5.2, is a mapping between words in the input sequence and words in the output sequence, which indicates which words in the source sentence are the most important for translating the word in the target sentence.

This concept is not only relevant for machine translation. In any sequence-to-sequence task, we might want to focus on different parts of the input when generating different parts of the output in a dynamic way. This dynamic focus is akin to a spotlight, with the decoder paying various degrees of attention to different tokens in the source sequence, emphasizing their importance and lessening the importance of the other tokens.

Attention as we know it today first emerged as an augmentation to an RNN-based translation model by Bahdanau et al. (2014), which learned the word alignment and translation jointly. In their model, the encoder stores all of the hidden states at each time step, and then the decoder at each time step calculates a context vector by getting a weighted sum of the hidden encoder states. This context vector is concatenated to the hidden state of the decoder. By using different weights at each time step, the model can have a different context for each output token. The weights are calculated by a feedforward alignment model that is learned jointly with the translation model.

Attention mechanisms help bypass the bottleneck of the finite hidden state vector. It allows the model to have fresh information from the relevant parts of the input without having to fully "remember" the entire input at all times. Intuitively, this also aligns better with how humans would perform sequence-to-sequence tasks. Translators, instead of memorizing the entire source sentence and then writing the complete translation in one shot, often refer back to the source sentence mid-translation to make sure they capture all the details.

2.2.8 Transformer models

The Transformer model, introduced by Vaswani et al. (2017) in their paper titled "Attention is all you need", represents a groundbreaking advancement in the field of NLP, which is still at the core of most state-of-the-art models. The name of the paper hints at the primary mechanism used in transformer models, namely attention.

Transformer models change the conventional sequence-to-sequence architecture by eliminating the need for a hidden state vector passed between the encoder and decoder. Instead, they rely exclusively on an extended attention mechanism for information transfer from the encoder to the decoder. Furthermore, they introduce a novel encoder and decoder architecture. In contrast to traditional models, which use recurrent layers to process sequences incrementally while maintaining a hidden state vector for self-reference, transformer models embrace a self-attention mechanism. This self-attention calculates context at each time step by performing a weighted sum over other decoder hidden states. This allows the training of transformer models to be parallelized to a much higher degree as they no longer have strict recurrence.

Key components of the Transformer architecture include:

1. **Cross-attention**: Instead of passing a hidden state vector from the encoder to the decoder once, thereby creating a bottleneck, a transformer calculates a new context vector at each step by calculating a weighted sum over the hidden states of the encoder.

2. **Self-attention**: Instead of keeping a single internal state vector that gets incrementally updated as the sequence gets processed token-by-token, both the encoder and decoder calculate the context at each step from the other outputs. The encoder attends to all outputs from the previous layer bidirectionally, whereas the decoder only attends to the outputs of the previous layer for tokens that came before the current token to preserve the auto-regressiveness of the model.

3. **Multi-Head attention**: Instead of having just one attention model that calculates a single context vector, transformer models use multi-head attention, which calculates multiple context vectors, each based on a different set of tokens. This allows the model to simultaneously capture different long-range dependencies.

4. **Positional encoding**: Transformers lack any inherent notion of sequential order since they do away with regression. Because of this, a positional encoding is added to the input embeddings, which provides the model information about the position of tokens in the sequence.

The Transformer's ability to model long-range dependencies and its parallel processing capabilities have made it the foundation for various state-of-the-art models, including BERT (Devlin et al., 2019) for language understanding and GPT (Generative Pre-trained Transformer) for text generation (Radford and Narasimhan, 2018).

2.2.9 Large Pre-Trained Language Models

A notable recent development in the field of NLP is the creation of large language models (usually based on the transformer architecture) that are trained on a massive amount of unlabeled text. These models are trained on one or a combination of different objectives, which are meant to teach the model language nuances and are applicable to large text corpora without needing any kind of human annotation.

One early example is BERT (Devlin et al., 2019), which was trained using a masked language modelling objective (predicting one or more masked tokens in a

text) together with a next sentence prediction objective (determining whether two sentences were taken consecutively from a text). Many variations of this kind of model have been created (Liu et al., 2019; Lan et al., 2020; Yang et al., 2019).

2.3 Generating Text With Language Models

Language models calculate the probability of a sequence of tokens appearing in natural language. This can be used to generate new text by using various methods. This section provides an insight into the way text can be generated with a language model. It starts with basic principles and unconditioned generation, then explains various methods of controlling the output to make it more useful for various tasks.

2.3.1 Practical Applications

Besides generating free-form text, language models find use in various NLP subtasks. In this section, a non-exhaustive list will be given to give an insight into the wide range of tasks that these models might be used for.

For speech recognition, a language model might be employed to differentiate ambiguous phoneme sequences (Jelinek, 1985; Derouault and Merialdo, 1986).

For statistical machine translation, a language model is used as a part of the scoring function for translation hypotheses. It has been shown that larger and better language models increase the translation accuracy (Brants et al., 2007).

Many different types of classifiers can be built on top of pre-trained language models through transfer learning, e.g. for sentiment classification, entailment classification, and emotion classification (Houlsby et al., 2019).

For more text generation-oriented applications, one need not look much further than the recent explosion in popularity of AI-powered chatbots such as OpenAI's ChatGPT [2] and earlier dialogue systems (Williams et al., 2010; Wen et al., 2015). These systems may be used for a wide variety of tasks, including dialogue systems

[2] https://openai.com/blog/chatgpt

built for specific use cases and domains, as well as free-form chatbots. In fact, Brown et al. (2020) show that large language models are able to perform a wide variety of tasks in a zero-shot way, that is, without being explicitly trained on it, if they are just prompted to do so through text.

2.3.2 Sampling Methods

A language model is able to provide the probabilities of the next token given an incomplete sentence. To generate text from these probabilities, one needs to sample from the model. This can be done in different ways.

Naive approach to generating text The impractical, naive method for generating text from a language model is to just generate completely random sequences of tokens, evaluate their probabilities with the language model, and take the text with the highest probability. In theory, if this process is repeated long enough, it will result in coherent and novel text eventually. In practice, however, it is obviously grossly inefficient since a random sequence of tokens is much more likely to be nonsensical.

Greedy sampling An already much more efficient method is sampling one token at a time based on the current unfinished sentence, starting from an empty sentence. However, this method will favour sentences that consist of the most common tokens, resulting in uninteresting text. Furthermore, if the probability the model assigns to a piece of text is stable (i.e. there is no random element, so the same text always gets assigned the same probability), the same sequence will be generated every time given the same prefix.

Top-K sampling An improvement to this method is instead of taking the most likely next token to randomly choose one of the k most likely tokens, either assigning each of them the same chance of being picked or using their actual probabilities to weigh them. This is called top-k sampling.

Beam decoding In contrast to greedy decoding, beam decoding is a more sophisticated approach to generating text with language models. It addresses the limitations of greedy decoding by considering a set of alternative tokens at each step, known as the "beam". Instead of just using a single best token, the best n hypotheses are kept and expanded where n is the beam width. This method allows the model to explore multiple possible continuations simultaneously, mitigating the issue of getting stuck in suboptimal paths. Beam decoding allows the model to choose words that greedy decoding would find suboptimal in the short term in the hopes of finding a better overall output. Beam decoding is a widely used technique in natural language processing and machine translation tasks due to its ability to produce more contextually coherent and higher-quality outputs.

Top-P (nucleus) sampling Instead of truncating the search space by taking only a fixed number of the most likely tokens, a more reliable approach is to truncate based on some probability threshold, like done by Holtzman et al. (2020). This removes the unreliable part of the probability distribution and instead samples only from a dynamic "nucleus" of tokens which have the vast majority of the probability mass. The authors note that this approach helps prevent the model output from degenerating and becoming repetitive.

The role of decoding in neural language models All of the aforementioned sampling methods are applicable to all types of language models, from simple n-gram models to sophisticated neural models. However, while a good decoding method is crucial for simple models with limited contexts, it is less critical for neural models, which can use a lot more of the previous context when outputting the likelihoods of the following tokens.

Nevertheless, sampling from neural language models can still be a tricky task, with the same model being able to generate both fluent and varied text, as well as repetitive text purely based on the selected decoding method (Holtzman et al., 2020).

2.4 Controlling Language Model Output

Pre-trained language models are good at generating fluent text. They are, however, hard to control due to their black-box nature. Controllable text generation methods address the question of how to make sure the output conforms to the required constraints while still retaining its fluency. In almost all use cases of text generation, it is necessary to control the output in some way, be it making the model output about a specific topic or forcing it to contain specific information. This section provides a look into some of the methods for doing that.

The survey by Prabhumoye et al. (2020) provides a framework for how controllable text generation can be approached. They identify five parts in text generation that can be targeted: the external input, or how the model is initialized; the sequential input, or what is passed into the model as input at each time step; the generator module, or what calculations are performed on the inputs; the output module, or how the output of the model is turned into natural language; and the training objective, or how the generator is trained. Parts of this section are loosely based on the classification provided in their survey.

2.4.1 Prompting and In-Context Learning

The simplest way to have a measure of control over what the language model generates is to just start it off with some human-written text and let it generate what follows next. This is often called priming, referencing the concept of the same name in psychology, where previous stimuli ”prime” the individual to respond to future stimuli in a certain way. In language models, priming involves providing an initial input, which influences subsequent outputs, aligning them with the context or direction established by the priming text. Another term interchangeably used for this is prompting.

Practically speaking, prompting involves giving the model either incomplete text to finish or instructions on what to output, with the latter being especially relevant for LLMs who have been trained for human interaction via a chat-like interface.

This technique is gaining traction, particularly with the rise of publicly accessible large language models such as ChatGPT[3].

Despite the apparent simplicity of this method, the way the prompts are constructed can have a substantial effect on the quality of the output. This has led to a deluge of articles on how to construct good prompts [4] in order to unlock the knowledge inherent in these models.

Closely related to prompting is **in-context learning**, where a large language model is prompted by providing some examples of the desired output. This capability has been shown in models such as GPT (Brown et al., 2020). Garg et al. (2022) show that "..transformers can encode complex learning algorithms that utilize in-context examples in a far-from-trivial manner.".

2.4.2 Fine-tuning Pre-trained Language Models

Before large pre-trained language models, the typical approach to creating a model for any language-related task was to train one from scratch, as can be seen in the survey by De Mulder et al. (2015). This is a time-consuming process wherein the model ends up first needing to learn a lot of general knowledge about language.

A much more effective method is to first train a general language model as a base and then fine-tune it for the specific task (Dai and Le, 2015; Radford and Narasimhan, 2018; Howard and Ruder, 2018). While the pre-trained language models are not task-specific, Lewis et al. (2020) show that, especially for text-generation tasks, the type of pre-training matters. They show that a model trained to reconstruct corrupted text (where instead of single tokens being masked like in a masked-language modelling pre-training objective, whole sequences of words are removed at a time, with the model also having to determine how many words are missing) performs better when fine-tuned for text generation tasks.

[3]https://openai.com/blog/chatgpt

[4]For instance: https://www.plannthat.com/good-vs-bad-ai-prompts/

2.4.3 Decoder Initialization

In encoder-decoder architectures (which include transformers), the hidden state of the decoder is usually initialized with the hidden state of the encoder. However, it is possible to control the model's output by adding a control vector to the model either via concatenation or other linear operations. This corresponds to the external input module from the survey of Prabhumoye et al. (2020).

These kinds of approaches can be found in information-driven dialogue systems such as the ones by Dinan et al. (2019), Zhou et al. (2018), and Ghazvininejad et al. (2018). They all do a variation of concatenating a knowledge vector independently encoded from relevant documents to the hidden state before decoding. These passages are retrieved via different kinds of information retrieval systems beforehand from Wikipedia, Amazon reviews or other factual sources.

Other methods (Liu and Lapata, 2018; Balachandran et al., 2021) instead train the encoder in such a way that its output is decomposed into separate subspace, e.g. the first half responsible for the semantic content of the text, the second half responsible for the structure and form of the text. With a representation encoded in such a way, it is possible to switch out just the part responsible for the content or the part responsible for the form in order to control the output. This decomposition can be achieved in different ways. Liu and Lapata (2018); Balachandran et al. (2021) complement the encoder with a separate structure encoding model, whereas Romanov et al. (2019) and Wang et al. (2019) use adversarial approaches, with a discriminator controlling the latent space of the encoder.

2.4.4 Extra Input to the Decoder

The decoder receives inputs at each time step, often through an attention mechanism from the encoder. This corresponds to the sequential input model of the survey by Prabhumoye et al. (2020). Similarly to the previously mentioned approach of adding a vector containing external information to the initial hidden state of the decoder, this vector may be added at each time step to what the decoder receives from

the encoder. This kind of method has been tried for definition modelling (Noraset et al., 2017), dialogue systems (Zhou et al., 2018), and Wikipedia passage generation (Prabhumoye et al., 2019) with some limited success.

2.4.5 Controllable Model Architectures

The generator is defined by Prabhumoye et al. (2020) as the set of computations that the text generation pipeline performs on the input at each time step. In the case of neural models, this is the base unit of the model architecture, e.g. the transformer block or a recurrent neural network. There are various changes to these architectures that have been proposed to allow better control over their output.

Gan et al. (2017) modify an LSTM by factoring the usual weight matrix responsible for processing the input into three separate matrices and training the model to capture stylistic information in one of the matrices. They use this model to generate stylistically different image captions by swapping out the weight matrix responsible for the style.

Kiddon et al. (2016) augment a GRU-based language model with an extra input: an agenda with a checklist, which the model is trained to incorporate into its output through extra weight matrices which determine when to generate from the model and when to generate using one of the checklist items. They show the effectiveness of this method by generating task-specific dialogues and generating recipes. Similarly, Wen et al. (2015) add another sentence planning gate to the LSTM cell, which controls the overall semantic information contained in the output, with the usual LSTM components taking care of the fluency and language modelling aspects.

2.4.6 Guided Decoding

Neural language models output token probabilities at each time step. To generate text, this output needs to be used together with some decoding algorithm such as greedy search or beam search (looked at in more detail in section 2.3.2). This proves

to be an avenue for controlling the output of a language model that often requires no changes to the underlying pre-trained model.

A simple example of this kind of approach is to generate multiple outputs with a decoding method that generates varied output, then re-ranking them according to some extra criteria. Yi et al. (2019) train a model to re-rank the outputs for a dialogue system to favour interesting responses over generic all-purpose ones. Similarly, Krishna et al. (2022) train an output re-ranking model which evaluates the quality of generated text given a human written prefix, evaluating how good of a continuation the generated text is to the prefix.

Instead of re-ranking fully generated outputs, it is possible to use auxiliary evaluation criteria to re-rank hypotheses during decoding. Hu et al. (2019a), Anderson et al. (2017), and Hokamp and Liu (2017) use variations of beam-search with lexical constraints, which make sure that specific tokens appear in the output, either in any order in a strict sequence.

2.5 Machine Translation

At its core, machine translation is the task of finding the most likely translation into some target language given some sentence in the source language: $\text{argmax}_e P(e|f)$. This definition is an instance of a generic conditioned language model whose task is to find the most likely text e given some arbitrary attributes f.

This section starts with historical approaches to MT, such as rule-based and statistical models, going into modern neural approaches, showing the similarities in their architecture with that of neural language models. Finally, this section goes into some of the usual domain-adaptation methods and the commonly seen specific task of formality-sensitive machine translation.

2.5.1 Rule-based Machine Translation

The earliest methods for machine translation relied on hand-crafted rules on how to replace certain words in the source sentence with target language words and were

brittle and highly specific. The first such effort was a collaboration between Georgetown University and IBM, which demonstrated a rudimentary Russian-English machine translation system in 1954 [5]. While this was a limited proof of concept, it provided an impetus for further work on the tricky problem of automatic translation.

2.5.2 Statistical Machine Translation

The advent of statistical machine translation (SMT) marked a paradigm shift in the approach and mindset to language processing. Instead of relying on rules handcrafted by expert linguists, one could learn the intricacies of language automatically through statistical means. In this paradigm, data is king. The more high-quality data there is, the better the end result will be. The core idea behind SMT systems is to model the translation process as a statistical problem, estimating the probability of a source language sentence generating a target language sentence. (Brown et al., 1990)

This task was usually factored into different parts, each accomplished by a separate model. Initially, this was the translation model, ensuring the adequacy of the translation and making sure it captures the meaning as fully as possible, and a language model, ensuring the fluency and naturalness of the translation. This was expressed by Brown et al. (1993) as the fundamental equation of machine translation, where e is the target translation, f is the source sentence, $P(e)$ is the language model, and $P(e|f)$ is the translation model:

$$\hat{e} = \underset{e}{\operatorname{argmax}} P(e)P(e|f) \tag{9}$$

One of the pioneering SMT systems is the IBM Model 1 (Brown et al., 1993), which introduced the concept of word alignment probabilities that could be learned automatically from a parallel corpus that is only aligned on a sentence level. This was

[5] https://www.ibm.com/ibm/history/exhibits/701/701_translator.html

done through an iterative algorithm called expectation maximization. Their subsequent models, IBM Model 2 through 5, introduced more components and more sophisticated concepts. IBM Model 2 introduced a more sophisticated word alignment probability model that was also conditioned on the lengths of both sequences. IBM Model 3 introduced the concept of fertility, where one word could now be translated into a variable number of words. IBM Model 4 added a re-ordering model, allowing for changing word order between source and target sentences. IBM Model 5 fixed some deficiencies in the alignment algorithm.

Statistical machine translation achieved notable success in the late 1990s and early 2000s (Nießen et al., 1998; Vogel et al., 2000), demonstrating the feasibility of automated translation on a larger scale. However, it still faced challenges in handling idiomatic expressions, long-distance dependencies, and rare or out-of-vocabulary words.

2.5.3 Phrase-based Machine Translation

Word-level translation methods such as the IBM models work well with similar languages, where the grammar and the way ideas are expressed are similar. With more syntactically different languages, these simplistic alignment models tend to struggle. A more sophisticated idea first originated by Och et al. (1999) is to align whole phrases instead of individual words, capturing idiomatic expressions much more accurately. This led to steady improvements in translation model capability with various authors experimenting with different ways to extract phrase alignment, different ways to decode the final translation (Koehn et al., 2003; 2007; Li et al., 2009), and making the extracted phrases hierarchical and composeable (Chiang, 2005; Vilar et al., 2010).

Crucially, these models improve over word-based SMT models trained on the same data, showing an improvement that is attributable to the model architecture, not simply using more data. Despite this, the main factor determining the quality of the resulting model is still the amount of training data, leading to the development of more and more efficient methods (Gu et al., 2018; Och and Ney, 2000; Gao and

Vogel, 2008; Brants et al., 2007) to utilize the large corpora available with the advent of the Internet.

2.5.4 Neural Machine Translation

While SMT laid the groundwork for machine translation research, introducing concepts such as attention, neural machine translation (NMT) has revolutionized it. This transition from SMT to NMT exemplifies the broader shift in NLP towards deep learning and end-to-end approaches, where instead of factoring the task into separate smaller tasks (such as alignment, lexical or phrase translation, language modelling), a single model is able to accomplish the entire task in one pass.

The first NMT models were proposed by Bahdanau et al. (2014) and Sutskever et al. (2014), both of which use a recurrent encoder-decoder architecture. The model from Sutskever et al. (2014) is based on a multi-layered LSTM with the input to the encoder reversed so that the last token that the decoder sees is the start of the sentence, allowing the decoder to handle longer sequences without ”forgetting” the beginning of the sentence.

In contrast to that, Bahdanau et al. (2014) solve the long sequence problem differently. They augment their RNN encoder-decoder with an attention mechanism, allowing the model to learn which parts of the input to focus on at each time step. This is very similar to the concept of word alignment, however, it is improved due to being a soft alignment, allowing the model to attend to multiple source tokens with different weights.

Larger models and models trained on more data soon followed, such as Google's encoder-decoder model (Wu et al., 2016) made up of deep LSTM layers and the numerous even more recent efforts based on the transformer architecture (Vaswani et al., 2017; Bawden et al., 2020), achieving state-of-the-art performance and dominating most leader-boards of conferences and workshops focusing on machine translation from 2016 onwards (Bojar et al., 2016). Since then, NMT models have arguably achieved parity with human translators in some language pairs and some domains, such as Chinese to English translation of news texts (Hassan et al., 2018).

More recently, the focus of machine translation research has shifted to other aspects, such as improving performance on low-resource languages (NLLB Team et al., 2022), multi-lingual translation models capable of translating between many different language pairs at once (Eriguchi et al., 2022) and end-to-end translation of speech (Radford et al., 2023). With large amounts of data available, the more interesting questions seem to be about how to use the available data most effectively.

2.5.5 Domain Adaptation for Machine Translation

While translation models have achieved parity with human translators on some language pairs and domains, many domains remain unexplored either due to insufficient interest or insufficient available data. Furthermore, existing domains change over time, with a striking example being news texts, which, right after the start of the COVID-19 pandemic, were dominated by topics related to the pandemic (Anastasopoulos et al., 2020).

The overview by (Saunders, 2022) divides MT domain adaptation techniques into those revolving around data selection or generation, model architecture, parameter adaptation procedure, and inference procedure. Different types of domain adaptation methods will be looked at more in-depth in this section.

Fine-tuning The most common baseline approach for domain adaptation is to fine-tune the model on in-domain data. Luong and Manning (2015) were among the first to apply this method for neural machine translation. While this approach generally achieves good results, it has some limitations, the main one being that a suitable dataset for training does not always exist.

Data selection Domain adaptation through fine-tuning requires an in-domain dataset. If one does not exist, it may be created through various methods, either through selectively filtering existing data or generating new synthetic data.

One approach is to take large existing general-domain corpora and filter out sentences which match the desired domain. This can be done via retrieval methods and

comparisons against a small number of given in-domain examples, such as looking for sentences with an n-gram overlap (Farajian et al., 2017) or using a more fuzzy matching mechanism (Xu et al., 2019) or using sentence embeddings to find sentences which are similar to a small number of available in-domain sentences (Wang et al., 2017).

Data synthesis If a monolingual in-domain dataset is available, it can translated from the target language back to the source language with some other MT model in order to obtain a parallel corpus (Sennrich et al., 2017). Despite this kind of back-translation producing more noisy texts than human written texts, Sennrich et al. (2016b) show that training on such data can still improve a model's performance. Surprisingly, Currey et al. (2017) show that it is not even necessary to translate the sentences to achieve some improvement in in-domain performance because it teaches the model to produce in-domain vocabulary.

If the monolingual dataset is in the source language, forward translation can be used instead, where the text is translated using machine translation. If the same model that is being fine-tuned is used for this, then this can be seen as a form of self-learning, where the model uses its own output to learn. Chinea-Ríos et al. (2017) employ this for low-resource machine translation. Alternatively, a stronger teacher model may be used for this in order to train a strong yet small domain-specific translation model (Gordon and Duh, 2020).

Whenever a lexicon of domain-specific term translations is available, they may be used to improve synthetic data. For example, Hu et al. (2019b) use a word-by-word back-translation that uses the lexicon for translating domain-specific terms.

An existing in-domain dataset may be extended by generating extra synthetic data through various noising methods. Vaibhav et al. (2019) synthetically introduce such noise as spelling mistakes, grammar mistakes, emoticons, and profanity. Karpukhin et al. (2019) show that this kind of noise improves MT model robustness.

Domain tagging One way to accomplish domain adaptation is to teach the model to use a separate domain embedding as part of its input. The way this input is

structured may differ from approach to approach. Sennrich et al. (2016a) pass it in a domain marker as an artificial token at the end of the sentence. Kobus et al. (2017) and Tars and Fishel (2018) instead add extra dimensions to the word embeddings into which the domain information is encoded inline. They both find this kind of extra feature approach to slightly outperform the discrete token approach. Britz et al. (2017) adds this domain tag to the start of the target sentence instead.

A complication arises from the inherently mixed and ambiguous nature of domains. A single sentence may be said to belong to many different domains depending on the granularity of the domain labels, so a single tag may not always be possible. Stergiadis et al. (2021) solve this by tagging each sentence with multiple domains and teaching the model to effectively use a mixture of them. Mino et al. (2020) extend this idea by also tagging sentences for other aspects, such as how noisy they are. This gives credence to a more loose definition of domain for practical purposes.

Extending network architecture Domain adaptation can be achieved through changes in the model architecture. These methods usually add trainable parameters to the model, and rather than adapting an already pre-trained model, they usually are trained from scratch to have control over the domain. Saunders (2022) notes, however, that a lot of these kinds of methods could, in principle, be applied to pre-trained models without having to fully retrain them, but they are trained from scratch to simplify the technical implementation.

Parts of the network could be duplicated, training each instance for a specific domain. Pham et al. (2019) add a domain-specific part to the embedding layer, which creates domain-specific lexical embeddings. Zeng et al. (2018) have separate encoders generating a domain-specific sentence encoding and a generic, non-specific encoding. Jiang et al. (2020) train separate attention modules for each domain. Gu et al. (2019) go a step further and train wholly separate encoders and decoders for each domain, which are used together with a domain-agnostic encoder-decoder.

Instead of duplicating existing parts of the model, new ones may be added and trained separately by fine-tuning the model with only the new parameters unfrozen.

Bapna and Firat (2019) add small domain-specific adapter layers on top of the encoder and decoder. Pham et al. (2020) train a separate domain discriminator to determine which of the adapters to use. This approach has gained popularity due to its relative simplicity and because it avoids catastrophic forgetting by not changing any of the pre-trained model's weights.

Training schemes Fine-tuning a translation model on new domains can lead to forgetting previous domains. The methods described in this section change how the model is trained in order to help it better adapt to new domains and to avoid the usual fine-tuning pitfalls such as catastrophic forgetting and overfitting.

The approaches that extend the network architecture all generally avoid catastrophic forgetting by simply freezing the pre-trained model weights and only training their added parameters (Jiang et al., 2020; Gu et al., 2019; Bapna and Firat, 2019; Pham et al., 2020). Furthermore, adapter modules (Bapna and Firat, 2019; Pham et al., 2020) can have a residual connection, allowing the model to bypass it entirely. Liang et al. (2021) go even deeper into the model and selectively freeze only sets of parameters that they show are responsible for most of the model's capability, fine-tuning the under-utilized parts of the network to specific domains.

The order in which the training examples are shown to the model can have an impact on the resulting performance. This was suggested by Bengio et al. (2009), who start the model off with more straightforward examples and end with tricky examples. For domain adaptation, this idea may be used to, for example, start the model with more generic examples and end with more in-domain ones. Saunders (2022) notes that simple fine-tuning is already a type of curriculum learning. Zhang et al. (2019) and van der Wees et al. (2017) do curriculum learning to improve model performance on some specific domain identified as a subset of the general training corpus.

Instead of changing the order in which the model sees training instances, they may instead be weighted differently. Different methods for determining how to weigh each specific instance may be used (Foster et al., 2010; Joty et al., 2015), including

training a separate domain discriminator (Chen et al., 2017). The effect of this is effectively the same as curriculum learning since both methods emphasize which instances the model should pay more attention to while learning.

Inference schemes The domain may be adapted to during inference time. These kinds of methods generally avoid having to fine-tune the translation model at all. Since the adaptation method proposed in this thesis falls under this category, these methods will be looked at in section 3.1 under related work.

3 Related Work

This section focuses on the research that this thesis builds directly on top of and methods that are directly adjacent to this research. It starts with some domain-specific datasets and translation models upon which this thesis builds. Then it dives into some inference-time domain adaptation methods for machine translations (since they generally do not fine-tune the translation model or change it in a way which would require re-training), then looks at domain adaptation approaches tried for the task of formality-sensitive machine translation and finally ends with a quick look at the controllable text generation methods which this thesis adapts for machine translation.

3.1 Machine Translation Domain Adaption at Inference Time

If one wishes to adapt an existing translation model to a new domain without fine-tuning it, the adaptation must happen at inference time.

A simple scheme is to use multiple translation models that are trained on different domains in an ensemble. Their predictions may be combined in various ways. Freitag and Al-Onaizan (2016) achieve reasonable performance on unknown domains with uniform weights, Sajjad et al. (2017) do better on known domains by determining static weights from a development set. Liu et al. (2020a) determine the weights right before inference by comparing how similar the input sentence is to each domain-specific model's training data. While this kind of method is simple, it requires training multiple translation models, which means the memory and time requirements scale linearly with the number of domains the system supports directly. However, given a set of domains with minimal overlap, Saunders et al. (2019) find that ensembling methods can also work on unseen domains.

A less resource-intensive ensembling method is proposed by Khandelwal et al. (2021) and later extended by Zheng et al. (2021) who use a k-nearest-neighbour retrieval model which retrieves potential in-domain translations for NMT decoder

states. These kinds of methods have been shown to improve in-domain performance without requiring additional NMT models.

A single translation model may be used in combination with domain-specific models that re-score translation hypotheses from a generic NMT model based on how in-domain they are (Dou et al., 2019). Zhang et al. (2018) perform this re-scoring based on how much the hypothesis overlaps with a similar retrieved in-domain sentence. These kinds of methods, however, are somewhat limited by the capabilities of the underlying translation model as they do not steer the actual inference process. The model might not have a good in-domain translation within the top hypotheses on unseen or rare domains.

This drawback may be solved through constrained inference methods. Khayrallah et al. (2017) constrain the NMT model with the output from a domain-specific phrase-based statistical machine translation model. This smaller statistical model provides domain-specific adequacy while the NMT fills the gaps to ensure fluency. Hokamp and Liu (2017) and Hasler et al. (2018) propose constraining the beam-search with lexical constraints. This can be used, for instance, to ensure specific term translations are used if an in-domain dictionary is available.

A similar idea to using lexical constraints during decoding to ensure correct domain terminology translations is to use these constraints in pre-editing or post-editing. A simple approach by (Song et al., 2019) is to replace terms on the source side with the correct translation from a domain-specific dictionary, encouraging the translation model to simply copy over the term. Dinu et al. (2019) experiment with different approaches to inserting the domain terminology. While these kinds of approaches do require the model to be trained to use these inserted terms, the models are not required to generate in-domain terms themselves, making them adaptable to unseen domains at inference. However, they need an exhaustive domain-specific term dictionary to work well.

3.2 Formality-sensitive Machine Translation

A single source sentence may have many valid target language translations. This is especially evident for language pairs with linguistic differences in aspects such as formality; for example, when translating from English to German, most sentences could be translated into either formal or informal registers. Niu et al. (2017) consider this aspect and introduce the task of formality sensitive-machine translation. Nadejde et al. (2022) later introduce the CoCoA-MT dataset for this task consisting of 6 language pairs with English as the source language for all of them in 3 different domains. They also include a reference-based evaluation metric, which gives the formality accuracy. Together with usual MT evaluation metrics such as BLEU, this can be used to compare the performance of different models on this task.

Different approaches have been tried for this task. The originators of the task (Niu et al., 2017) re-rank the outputs of an NMT system based on their formality. The authors of the CoCoA-MT dataset (Nadejde et al., 2022) try a straight-forward fine-tuning approach and introduce a synthetic token on the source side, which determines the desired formality, achieving an average of 81.8% formality accuracy in-domain and 72.6% out of domain.

Zhang et al. (2022) train a post-editing model which can rewrite the translation into either formal or informal forms and use it together with a fine-tuned mBART (Liu et al., 2020b) model that does the translations. They focus on the English-Hindi and English-Japanese language pairs and achieve a perfect formality accuracy score when including augmented data in the fine-tuning step. Lee et al. (2023) perform fine-tuning with synthetic data generated by large language models through prompting.

3.3 Controllable Text Generation

The main inspiration for this proposal is the plug-and-play language models of Dathathri et al. (2020). They control the attributes of the text they generate by using a pre-trained transformer language model as-is and augmenting the decoding

process with a comparatively tiny attribute model that steers the output in the desired direction.

A similar approach by Pascual et al. (2021) makes a simplification by directly shifting the output distribution towards the semantic space of a guide word or a bag of words. Because they modify the probabilities directly and do not have to do back-propagation, their attribute model does not need to be differentiable, allowing them to adjust the weighting at each inference step to ensure that the required words appear in a specific order.

These methods are powerful and expressive, but a part of their expressiveness comes from the fact that the output is usually not too constrained. Applying these in a translation context has the additional constraint of keeping the translation correct.

4 Methods

The proposed method uses a pre-trained translation model in a plug-and-play manner, without changing its learned weights or architecture, together with a bag-of-words scoring model that determines adherence to the specific domain in a plug-and-play manner. It is based on the controllable text generation method by Dathathri et al. (2020) that they call Plug-and-Play Language Models. The proposed method will, therefore, be called Plug-and-Play Neural Machine Translation (PPNMT). The code is available on GitHub[6].

This section more formally describes PPNMT as well as the types of in-domain datasets and translation models that will be used and how the results will be evaluated.

4.1 Datasets

This thesis uses two kinds of parallel datasets with slightly different characteristics regarding the domain they cover.

1. **Intrinsic domain**: These datasets contain parallel examples where both the source and target sentences are in the same domain (e.g. scientific text). It is something intrinsic to the ideas expressed in the text and cannot be separated from it. While translating this kind of data, the goal is to preserve the domain characteristics. These domains mainly deal more with the semantic meaning of the text and the topic.

   A sound translation system should keep this semantic meaning intact.

2. **Extrinsic domain**: In these datasets, the target text contains characteristics not present in the source sentence, often due to linguistic differences between the two languages. Two examples of this are formality level, which can be indicated by lexical or grammatical markers in one language, but be absent in

[6]https://github.com/EmilsKadikis/PPNMT

another language (e.g. German and English), and gender, which can similarly be expressed as inflections on verbs (e.g. in Arabic, where the gender of the speaker and the person being spoken to can be seen from inflections).

When translating this kind of data, the goal is to add these characteristics, ideally in a controllable way.

4.2 Similarity Between Language Models and Neural Translation Models

This thesis extends a controllable text generation method for neural machine translation. Therefore, this method's applicability to NMT hinges on the conceptual and architectural similarities between language models and neural translation models. This section defines both and shows where the similarities lie.

Language model definition A language model is given a sequence of tokens $X = \{x_0, ..., x_n\}$ and calculates the probability of the sequence appearing in natural language $p(X)$. Or, expressed slightly differently, it gives the probability of the next token given the rest of the sequence:

$$p(x_{n+1}|X) \tag{10}$$

Sequence-to-sequence model A sequence-to-sequence model is essentially a language model that is conditioned on some sequence beforehand. That is, given some input sequence $X' = \{x'_0, ..., x'_m\}$ of length m it gives the probability of an output sequence $X = \{x_0, ..., x_n\}$ of length n. Or, expressed in the same token-by-token manner, it calculates the probability of the next token given an input sequence and the output sequence so far:

$$p(x_{n+1}|X, X') \tag{11}$$

In practice, this is usually implemented with two separate models: an encoder and a decoder. The encoder processes the input X', outputting some latent representation. The decoder then takes this output as conditioning and generates the target sequence token by token.

Neural machine translation model definition An NMT model takes as input a sequence of tokens in one language (the source language) and outputs a sequence of tokens in another language (the target language) that is a likely translation. For historical reasons, the source text is often denoted by f and the target translation by e (meaning French and English). This gives the following definition for an end-to-end translation model, which is just an instantiation of a sequence-to-sequence model:

$$p(e|f) \tag{12}$$

This definition can be similarly restated to a token-by-token version by introducing a target sequence $E = \{e_0, ..., e_n\}$ and a source sequence $F = \{f_0, ..., f_m\}$, which gives the following definition that is very similar to the conditioned language model definition in equation 11:

$$p(e_{n+1}|E, F) \tag{13}$$

In practice, this is also often implemented through an encoder-decoder architecture such as a transformer model.

Transformer model definition A transformer model uses an attention mechanism to model dependencies in sequences. The recursive definition provided by Dathathri et al. (2020) will be considered. The language model LM takes the previous output that it produced x_t and the hidden state of the previous time-step H_t

(which for a transformer model represents the key-value pairs of all the attention heads at each layer) as input and outputs the new hidden state H_{t+1} and logits o_{t+1}:

$$o_{t+1}, H_{t+1} = LM(x_t, H_t) \tag{14}$$

The next token x_{t+1} is then sampled from a projection of the output logits o_{t+1} onto the vocabulary of the model by multiplying them with a weight matrix W.

$$x_{t+1} \sim p_{t+1} = Softmax(Wo_{t+1}) \tag{15}$$

In an encoder-decoder transformer architecture (such as the one used by NMT models), the hidden state could be split into two parts corresponding to the key-value pairs of the self-attention H_t^{self} that the decoder has to its previous outputs, and key-value pairs of the cross-attention H_t^{cross} that the decoder has to the outputs of the encoder:

$$o_{t+1}, H_{t+1}^{self}, H_{t+1}^{cross} = LM(x_t, H_t^{self}, H_t^{cross}) \tag{16}$$

While the encoder is its own transformer model with its own self-attention, for the purpose of this thesis, it can be ignored since the cross-attention will not be changed. Only the decoder will be controlled, treating it essentially like a language model.

Compared to the previously given definition of a sequence-to-sequence model, the conditioning of the model on the input sequence is captured through the cross-attention mechanism, and the conditioning on its own previous output is captured through the self-attention mechanism. Neural machine translation is commonly implemented with this kind of model.

4.3 Proposed Method: Plug-and-Play Neural Machine Translation

This thesis extends the controllable text generation method proposed by Dathathri et al. (2020) for machine translation by exploiting the architectural similarity of modern NMT models to language models. This section describes the domain adaptation method by first explaining the original controllable text generation method and then emphasizing the considerations needed to apply it to a translation model. Lastly, various additions to improve the performance of the base method are considered.

4.3.1 High Level Overview

On a high level, this method takes the recursive transformer model definition introduced earlier and, before using the hidden state of the self-attention head, runs it through a perturbation function:

$$H'_t = perturb(H_t, a(x_{1:t})) \tag{17}$$

This function nudges the hidden state in a way that enforces some domain-specific attributes defined by a domain classifier a.

When applied to the translation model, only the self-attention is perturbed.

$$H_t'^{self} = perturb(H_t^{self}, a(x_{1:t})) \tag{18}$$

$$o_{t+1}, H_{t+1}^{self}, H_{t+1}^{cross} = LM(x_t, H_t'^{self}, H_t^{cross}) \tag{19}$$

Conceptually, this could be seen as pushing the model's hidden state as if it had already been outputting an in-domain translation, prompting it to keep outputting within the same domain.

The original method by Dathathri et al. (2020) is applied on a GPT-2 model (Radford et al., 2019), which has only a self-attention mechanism and no cross-attention. Although the translation models used in this thesis have both, only the self-attention is perturbed to preserve the alignments the model has learned (which should, in principle, not be affected by the domain of the texts).

4.3.2 Capturing Domain Attributes

The definition of the perturbation function given in equation 17 contains a domain attribute model a. This is a discriminator that takes the model's output up until the current time step and gives a numerical score on how well the output conforms to the domain. Notably, it is orders of magnitude smaller and more straightforward than the translation model.

Dathathri et al. (2020) define two kinds of attribute models: small feedforward models trained to classify the decoder's hidden state on how well it adheres to the given domain and even more straightforward bag-of-word classifiers, which base their score on the probability the model gives to the tokens contained in the bag-of-words. This thesis focuses on the bag-of-words models due to their simplicity and interpretability.

The bag-of-words attribute model defines a domain simply by a collection of words in the target language that are likely to appear in a domain text or are related to the domain in some way. Given a bag-of-words $\{w_1, ..., w_k\}$ and an output distribution p which contains the logits of the next token, Dathathri et al. (2020) define the model like so:

$$\log p(a|x) = \log \sum_{i}^{k} p[w_i] \tag{20}$$

A few additions have been made in this thesis. To improve performance for contrasting domains (such as formal/informal), two bags-of-words are used: a positive set $\{w_1^+, ..., w_k^+\}$ which contains words from the target domain, and an opposing

negative set $\{w_1^-, ..., w_j^-\}$ which contains words from the contrasting domain. The model subtracts the log-likelihoods of the negative words from the log-likelihoods of the positive words:

$$\log p(a|x) = \log \sum_{i}^{k} p[w_i^+] - \log \sum_{i}^{j} p[w_i^-] \tag{21}$$

Crucially, this does not simply push the model towards outputting the tokens in the bag-of-words. Dathathri et al. (2020) note that related words not explicitly contained in the bag are generated. The same is observed for NMT as well.

4.3.3 Procedural Overview

This perturbation process is better understood through a step-by-step procedural explanation. The source language text is first processed by the encoder like usual, and then the decoder generates a translation token-by-token with the following steps:

1. **Forward pass**: The model generates the probabilities of the next token.
2. **Domain evaluation**: The model output is passed into a domain attribute model, which scores how well the output adheres to the domain.
3. **Backward pass**: gradients are calculated from the domain attribute model with regard to the hidden state of the model (not with regard to the parameters, like would be done during fine-tuning)
4. **Hidden state update**: The hidden states of the model are updated using the gradient from the previous step. These three steps may be repeated multiple times.
5. **New forward pass**: A new forward pass is performed using the perturbed hidden state. A slightly different probability distribution over tokens is output, from which the next token is sampled.

These five steps are repeated until the model finally outputs its end token, signifying that the whole input has been translated.

Forward pass This step is performed to get the unperturbed model output (in the form of logits that form the probability distribution of the next token). Typically, this probability distribution would be used directly to sample the next token. Instead, it is used in the perturbation process.

Domain evaluation The output logits are evaluated for their domain adherence. With the bag-of-words model defined in equation 21, this effectively means summing up all the logits for the tokens in the positive bag-of-words, summing up the logits for the tokens in the negative bag-of-words and calculating the difference between the logs of these sums.

Backward pass All of the calculations described in the previous paragraph are differentiable, which means we can calculate the gradients from the domain score with regard to the current self-attention key-value pairs. This effectively determines which values in the hidden state were responsible for making the positive tokens more likely and the negative tokens less likely.

However, using only the gradients from the simplistic attribute model makes the output non-fluent and repetitive. Because of this, another gradient is used, calculated through Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) to minimize the divergence between the output distributions of the original and perturbed translation models. This allows the model to stay fluent despite the perturbations.

Hidden state update In this step, similar to fine-tuning, **step-size**, a hyperparameter akin to a learning rate, is used to update the hidden state according to the gradients. Crucially, this is not fine-tuning, and none of the trainable model parameters are affected.

Dathathri et al. (2020) make further considerations here. Firstly, only the hidden states from a limited number of previous time steps may be perturbed for computational efficiency, as defined by a **window size** parameter. This is because the more distant hidden states affect the next token less, so they may be ignored. Furthermore, linear decay is used to smooth the boundary between perturbed and unperturbed hidden states. Secondly, the gradients for each layer are normalized by the largest gradient observed on it. This normalization is scaled by a hyperparameter **gamma**.

New forward pass After the previous has been performed several times (controlled by a parameter **iteration count**, with this thesis using 6) and the gradient has been accumulated, the final forward pass is performed with the perturbed hidden state. This results in a probability distribution. However, it is not sampled directly. First, it is fused together with the unperturbed probability distribution from the initial forward pass via post-norm geometric mean fusion, which mixes the two probability distributions according to a weight $\gamma_g m$:

$$x_{t+1} \sim \tilde{p}_{t+1}^{\gamma_g m} p_{t+1}^{1-\gamma_g m} \tag{22}$$

Here $\tilde{p}_{t+1}$ is the perturbed probability distribution and p_{t+1} is the original distribution. Combining them ties the generated text to the unconditioned translation model's distribution and ensures that the text keeps fluency. If $\gamma_g m = 1$, the token is sampled just from the perturbed distribution, but if $\gamma_g m = 0$, the original unperturbed distribution is. Dathathri et al. (2020) find that values in the range $0.8 - 0.95$ work well for text generation, with 0.95 being used in this thesis.

4.3.4 Extensions to the method

The core of the method is unchanged from Dathathri et al. (2020). Some extensions have been made to address specific issues for translation. These will be listed here for completeness.

Warm-up steps As applied to text generation, the original method usually starts with a prefix. This means that as soon as perturbations start, there are some past hidden states which may be perturbed. For translation, however, there is no prefix or past hidden states to start with. This poses a problem if the first word in the translation must be adapted to the domain.

This is solved by doing a few warm-up steps at the start, effectively forcing the model to output the padding token so that there would be some past hidden states to perturb.

Negative bag-of-words To adapt to contrastive domains, a positively weighted bag-of-words from the target domain is used in conjunction with a negatively weighted bag-of-words from the distractor domain. This helps the method better capture the most essential aspect of the domain to which the model is being adapted.

4.4 Baseline domain adaptation approach

For the baseline adaptation approach, the pre-trained model will be fine-tuned on available in-domain examples. This approach generally performs well in a single-domain setting but worsens in a multi-domain adaptation setting due to catastrophic forgetting. While methods exist for mitigating this, vanilla fine-tuning will be used strictly in a single-domain setting for simplicity.

4.5 Stronger perturbation

Dathathri et al. (2020) achieve a controlled text generation with a light touch, using a rather small perturbation step size across many time steps. Translation as a task is different. Not only is the choice of each next token informed by a language modelling objective, depending on which tokens have already been output, but the text in the source language must also be kept in mind. Overcoming this second constraint requires a stronger push in the form of a larger perturbation step size.

However, over a shorter time span, or else the translation might be pushed too far into incoherence.

4.6 Evaluation

Human evaluation is the most valuable because it directly evaluates the model's fit for purpose. Human evaluation is, however, expensive and time-consuming to carry out. Because of the logistical constraints caused by looking at translations between many different language pairs, human evaluation will not be used in this thesis, and instead, automatic metrics that have been shown to correlate with human judgment will be used.

There are many automatic evaluation methods for machine translation, a lot of which overlap in terms of their inner workings or purpose. For this thesis, the following three will be used:

- **BLEU** (Papineni et al., 2002)
- **chrF** (Popović, 2015)
- **BERTScore** (Zhang* et al., 2020)

Short descriptions of these methods and the reasoning behind using these and not others will now be given.

4.6.1 BLEU

BLEU score (Papineni et al., 2002) is based on n-gram precision, i.e. how many unigrams, bigrams, trigrams, and 4-grams from the reference translations appear in the target translation that is being evaluated. This precision is modified to not count n-grams which appear more often in the target translation than in the reference translations. These four separate precision scores are then averaged with a

geometric mean. Furthermore, a brevity penalty is applied to combat short translations achieving better precision scores if the translation is much shorter than the reference translations.

Due to the nature of n-gram overlap and the limited number of reference translations compared to all possible good translations, this metric is not meaningful for single translations. Instead, it must be applied over a whole corpus and averaged. For similar reasons, the scores are also not comparable between different datasets (which will have different reference translations) and even different languages (which might have different ways of tokenizing the texts).

Possible BLEU scores are from 0 to 1, with 0 meaning no overlap with the reference translations and 1 meaning perfect overlap. The scores are often reported as percentage points of 0 to 100. This is how the BLEU scores will also be reported in this thesis. In practice, scores over 30 indicate good translations and scores over 50 indicate very high-quality translations, sometimes even better than human translations [7].

This metric has been shown to correlate with human judgements despite only looking at the surface level and not considering the actual meanings of the translations. Because of this, it is one of the de-facto machine translation evaluation methods, which is why it is used in this thesis.

4.6.2 chrF

chrF (Popović, 2015) is a character-level n-gram F-score that has gained popularity due to its wide applicability in languages for which tokenization is not simple due to non-existent word boundaries. It is generally defined as follows:

$$chrF = (1 + \beta)\frac{chrP \cdot chrR}{\beta \cdot chrP + chrR} \tag{23}$$

Here, the $chrP$ and $chrR$ are character n-gram precision and recall, respectively (arithmetically averaged over all n-grams), and β is a weighing factor determining

[7] https://cloud.google.com/translate/automl/docs/evaluate#interpretation

how much importance is assigned to precision over recall. The metric authors determined that an n of 6 correlates best with human judgments. Minor variations exist with different precision/recall weighting and whether or not to treat whitespace as characters.

When interpreting chrF scores, it is essential to note that they are not comparable across datasets and languages. Like with BLEU scores, they are too dependent on the specific reference translations and typographical differences between languages.

This method was chosen due to its universal applicability to all languages regardless of the presence of word separators such as Chinese. While metrics like BLEU are usable in Chinese, their score is highly dependent on the tokenization scheme, which is a non-trivial problem for Chinese.

4.6.3 BERTScore

Both of the previously mentioned evaluation methods work on a surface level without considering the translations' actual meaning. BERTScore (Zhang* et al., 2020) offers an alternative which uses a BERT model to embed the translation and reference translations, then do a pairwise cosine similarity comparison. This allows the metric to better deal with synonyms and alternate ways of expressing the same idea.

BERTScore was the third chosen evaluation method to cover the drawbacks of the other two. For datasets with an intrinsic domain especially, we want to ensure that the translation's semantic meaning gets preserved. BERTScore indicates how well this was accomplished.

5 Experimental Setup

The proposed method is not domain or language-specific and, as such, could be applied to any domain and language pair. This section explains the specific datasets, hyperparameters, pre-trained models, and experimental settings.

Two baselines are required to answer the proposed research questions. Namely, a pre-trained model baseline establishes the un-adapted performance that the model has, and a fine-tuned model establishes how effective the usual adaptation method would be on the same model. These are then compared against the PPNMT method.

5.1 Datasets

In section 4.1, two kinds of datasets were defined: ones with an intrinsic domain and ones with an extrinsic domain. Three different datasets are used in this thesis, two with extrinsic domains and one with an intrinsic domain. These datasets, the language pairs, and the domains they cover are explained further.

5.1.1 Formality: CoCoA-MT

The Contrastive Controlled MT by AWS dataset (CoCoA-MT) (Nadejde et al., 2022) is a multilingual dataset that contains English sentences that native speakers have translated into a formal version and an informal version into German, French, Italian, Spanish, Hindi, and Japanese.

For each language pair, there are texts in three domains: topical chat, telephony, and call centre. The training set comprises 200 instances in the topical chat domain, and 200 instances in the telephony domain, and the test set contains 200 instances from each of the three domains, with the call centre domain not appearing in the training data.

The translations are annotated, marking the words responsible for the formality/informality in each sentence. These annotations are the basis of a simple formality accuracy measure included with the dataset, which tries to predict the formality

of the translation by looking for the marked formality phrases in the translations. The result is a formality accuracy score from 0 to 1. It is important to note that this is a crude measure that does not measure the quality of the translation and can be easily gamed by just repeatedly outputting the formality phrase. Because of this, it is used in conjunction with traditional MT evaluation methods.

This dataset contains instances that are made up of multiple sentences. However, translation models are usually trained on single sentences. The chosen models would sometimes not translate parts of the other sentences. Because of this, the data was first split into sentences.

5.1.2 Gendered language: The Arabic Parallel Gender Corpus 2.0

The Arabic Parallel Gender Corpus 2.0 (Alhafni et al., 2022) (APGC v2.0) is an expanded version of the APGC v1.0 dataset containing 590K words. Arabic is a morphologically rich language with two grammatical genders indicated by gender markers for both the first and second person. The dataset explores all possible combinations of the first person and second person being male, female, or ambiguous over three splits.

While this dataset was created to explore gender bias, this thesis aims to bias the translation in a desired direction, either translating the first and second person as being both male or both female. Because of this, only the instances labelled as both being male or both being female were used.

5.1.3 Fine-grained domains: FGraDA

The Fine-Grained Domain Adaptation corpus (FGraDA) (Zhu et al., 2022) is a Chinese-English dataset consisting of 800 development instances and 3967 test instances from four technology-related domains, namely autonomous vehicles, AI education, real-time networks, and smartphone. They also include domain-specific dictionaries with around 300 different domain-specific terms and their translations and

a large number of extracted wiki pages for each domain, which cover the terms in the dictionaries.

In their baselines, Zhu et al. (2022) discover that the NMT model often mistranslates domain-specific words and common words with a domain-specific meaning. They try to remedy this by using the extracted domain term dictionaries with methods such as grid beam decoding (Hokamp and Liu, 2017)

5.2 Translation Models

As the pre-trained translation models, the Opus-MT models (Tiedemann and Thottingal, 2020) trained with the Marian framework (Junczys-Dowmunt et al., 2018) are used. They are modestly sized transformer encoder-decoder models with six layers in each and eight attention heads for each layer trained on the OPUS dataset (Tiedemann, 2012) for a vast number of language pairs. In total, they have made available 1069 bilingual models and 56 multilingual models, supporting a total of 1739 different language pairs[8].

These models have limitations, as listed on their GitHub page[9]. Firstly, the authors note that most models were trained on a maximum of 72 training hours on 1 to 4 GPUs and that not all converged before hitting this time limit. Secondly, no data augmentation or filtering methods were used. Lastly, the validation data used for early stopping was often taken from Tatoeba, which mainly features short and simple sentences. Because of these limitations, instances in the CoCoA-MT dataset had to first split into single sentences (otherwise, the model would sometimes miss translating parts of the text entirely), and any data with Japanese as the source or target language had to be dropped due to the Japanese models not functioning well enough.

The specific models used are listed in table 1 with their checkpoints from the Huggingface Hub[10].

[8]https://opus.nlpl.eu/Opus-MT/

[9]https://github.com/Helsinki-NLP/Opus-MT#known-issues

[10]https://huggingface.co/Helsinki-NLP

Language pair	Model checkpoint
English-German	Helsinki-NLP/opus-mt-en-de
English-Spanish	Helsinki-NLP/opus-mt-en-es
English-Italian	Helsinki-NLP/opus-mt-en-it
English-French	Helsinki-NLP/opus-mt-en-fr
English-Hindi	Helsinki-NLP/opus-mt-en-hi
English-Arabic	Helsinki-NLP/opus-mt-en-ar
English-Chinese	Helsinki-NLP/opus-mt-en-zh
Chinese-English	Helsinki-NLP/opus-mt-zh-en
English-Japanese	Helsinki-NLP/opus-mt-en-jap

Table 1: The translation models that were used in the thesis. The English-Japanese model was found to severely underperform, outputting an empty translation around half the time, so the English-Japanese data was not used.

5.3 Domain Attribute Models

The domain attributes were captured in a bag-of-words containing domain-relevant tokens. Multiple ways of constructing these were tried, both manually using linguistic knowledge and automatically from the training data.

1. **Manual**: It can be doable to define a small set of tokens manually for certain domains.

2. **Contrastive**: The texts from the contrasting domains (two or more) are tokenized, and token counts are calculated. Any tokens that appear in multiple domains are discarded, and the top k unique tokens from each domain are picked. The tokens from the target domain form the positive bag-of-words, while those from all the other domains form a negative bag-of-words. This works best for contrasting domains where a single source sentence has multiple translations, which only differ in the domain-specific aspect.

3. **tf-idf**: The words salient to a domain may be identified through tf-idf or similar measures. Unique domain-specific keywords can be collected if words

that overlap between the domains are removed.

The first two methods were tried for the extrinsic domains (formality and gender), and the manual and tf-idf methods were tried for the fine-grained tech domains, where the contrastive method was not reliable enough due to the non-overlapping texts contained in the four separate domains. For the manual bag-of-words for formality, the formal and informal pronouns for "you" were used, and for the fine-grained tech domains, the terms in the provided domain dictionaries were used.

5.4 First Baseline: Unchanged Pre-Trained Translation Model

The pre-trained models listed in section 5.2 were first evaluated on the three datasets listed in section 5.1 without any fine-tuning or domain adaptation. BLEU, chrF and BERTScore were used, as well as the formality accuracy measure provided with the CoCoA-MT dataset. This provides the baseline performance, which the proposed method attempts to improve.

Some concessions were made for a more direct comparison. Due to technical and time constraints, the proposed adaptation method was implemented just for greedy search. Because of this, the baseline models were also evaluated using greedy decoding. This accounts for a decrease of around 2 for the BLEU score. Despite this, the method is, in principle, also applicable to beam decoding.

5.5 Second Baseline: Fine-Tuned Translation Model

The pre-trained models were also fine-tuned on the in-domain data separately for each domain. Since the formality and fine-grained technology domain datasets have limited examples, five-fold cross-validation was used to discover the optimal amount of epochs to fine-tune for (which turned out to be 2). Beyond that, the default hyperparameter values provided by the *Seq2SeqTrainer* from Huggingface's Transformers library were used with no specific considerations for either domains or datasets.

5.6 Proposed Domain Adaptation Method: PPNMT

See section 4.3.3 for a step-by-step overview of the method. It uses multiple hyperparameters that may be used to tune the perturbation process, which will be explained in more detail in this section. The formality domain for English-German was used to determine the best hyperparameters due to the author's familiarity with the language. Different considerations needed to be taken into account for translation compared to free-form text generation.

5.6.1 Additional Considerations for Translation

When the method is applied to text generation, a long passage is usually generated starting from a short prefix. This length allows the model some time to generate a long enough history to perturb. For translation, however, there is no initial prefix or history. This poses a problem if the first word of the translation has to be in-domain (e.g. when translating "**You** are late.") as perturbations are not possible yet. To resolve this, the model is allowed to do a few warm-up steps before translating (which is effectively just adding padding tokens as the prefix).

Furthermore, it was discovered that the decoder needs to be pushed more than an unconditioned language model. This is controlled by the iteration count (i.e. how many times backpropagation is done to accumulate the gradient) and the stepsize parameter (i.e. how much the history is perturbed at each step). That also means, however, that it is easier for the model to hit some unknown threshold and start producing adversarial and repetitive text. This is countered by performing perturbations only for a certain number of time steps since, especially for domains like formality, the translation model is likely to keep the same formality level if a formality-bearing word has been output.

5.7 Zero-Shot and Few-Shot Setting

It is possible to use the proposed method zero-shot by constructing the bag-of-words manually. This is tested for the English-German language pair for formality, using

the formal and informal first-person pronouns: ”Sie” and ”du” as the positive and negative bag-of-words, respectively. The performance of this was compared to fine-tuning at various few-shot settings. Furthermore, the pre-trained baseline can be considered a zero-shot adaptation method for this comparison.

The CoCoA-MT dataset also contains sentences with no formality markers in them. These were filtered out before selecting instances for the few-shot setting.

6 Results

The adaptation results are shown separately on the three used datasets, followed by the results of the zero-shot and few-shot adaptation experiments.

6.1 Formality Adaptation

The results of formality adaptation can be seen in table 2. As expected, the baseline fine-tuning approach significantly improves all metrics over the pre-trained baseline, increasing the formality accuracy to close to 100% for all language pairs. For PPNMT, a smaller improvement over the pre-trained baseline is seen, although for Hindi, the BLEU score decreases slightly, but formality accuracy still increases.

In a full-data setting, fine-tuning vastly outperforms the proposed method. A part of the improvement in the BLEU score, chrF and BERTScore is attributable to the model additionally adapting to the spoken domain inherent to the dataset examples. In contrast, the proposed method only targets the formality level. However, the high formality accuracies that the fine-tuned baseline achieves speak to the power of fine-tuning as a domain adaptation method.

Notably, there are only small or insignificant changes in the BERTScores between the pre-trained baseline and the PPNMT model. That could indicate that there are no significant changes in the meaning of the translation unless the model is fine-tuned.

6.2 Gender Adaptation

The results of the gender adaptation can be seen in table 3. The pre-trained model achieves very low scores on the dataset. Judging by the higher score for the male texts, it can be assumed that the pre-trained model is more biased towards outputting in this gender. Although there is no automatic domain evaluation metric, we can judge the improvement in BLEU score, chrF and BERTScore. The proposed

Language	Method	Formal				Informal			
		BLEU	chrF	BERTScore	Acc.	BLEU	chrF	BERTScore	Acc.
EN-DE	pre-trained	33.5	57.1	0.874	0.519	33.0	56.9	0.877	0.481
	fine-tuned	**43.9**	**64.6**	**0.903**	**0.998**	**42.4**	**63.4**	**0.901**	**0.974**
	PPNMT	37.2	59.9	0.881	0.915	36.1	59.1	0.881	0.846
EN-ES	pre-trained	38.8	63.8	0.887	0.200	44.9	66.3	0.897	0.800
	fine-tuned	**48.3**	**68.1**	**0.904**	**0.995**	**48.1**	**68.2**	**0.904**	**0.972**
	PPNMT	40.9	65.1	0.889	0.746	43.9	65.5	0.894	0.938
EN-IT	pre-trained	36.4	63.0	0.874	0.048	44.5	66.3	0.892	0.952
	fine-tuned	**51.2**	**69.5**	**0.910**	**0.984**	**51.9**	**70.0**	**0.912**	**0.987**
	PPNMT	37.3	63.3	0.877	0.230	43.8	65.6	0.885	0.968
EN-FR	pre-trained	37.6	58.9	0.876	0.724	33.8	56.6	0.871	0.276
	fine-tuned	**43.7**	**63.9**	**0.893**	**1.000**	**42.8**	**62.5**	**0.891**	**0.931**
	PPNMT	40.7	61.8	0.881	0.993	37.0	58.5	0.874	0.749
EN-HI	pre-trained	17.1	38.6	0.835	0.785	15.1	36.4	0.828	0.215
	fine-tuned	**25.7**	**46.1**	**0.864**	**0.992**	**24.8**	**47.2**	**0.865**	**0.988**
	PPNMT	16.4	39.6	0.836	0.906	14.3	37.8	0.828	0.442

Table 2: Formality adaptation results for the two baselines and the proposed method (PPNMT) in a full-data setting. The fine-tuned baseline outperforms the proposed method.

Target Gender	Method	EN-AR		
		BLEU	chrF	BERTScore
Female	pre-trained	6.5	34.2	0.804
	fine-tuned	**14.5**	**42.1**	**0.832**
	PPNMT	6.8	35.3	0.796
Male	pre-trained	10.1	39.6	0.821
	fine-tuned	**16.9**	**42.3**	**0.835**
	PPNMT	9.1	39.3	0.806

Table 3: Arabic grammatical gender adaptation results for the two baselines and the proposed method (PPNMT) in a full-data setting. The target gender is pushed for both first and second person.

method helps the model output more female-leaning texts, but when adapting to male texts, it slightly hurts performance.

Fine-tuning, as expected, greatly improves the results and vastly outperforms the proposed method.

Domain	Method	ZH-EN			EN-ZH		
		BLEU	chrF	BERTScore	BLEU	chrF	BERTScore
Autonomous Vehicles	pre-trained	27.9	55.9	0.939	-	25.8	0.831
	fine-tuned	29.4	57.6	**0.942**	-	**32.1**	**0.855**
	PPNMT	25.1	55.5	0.936	-	24.5	0.819
AI Education	pre-trained	31.0	58.3	0.943	-	28.2	0.839
	fine-tuned	**33.3**	**59.4**	**0.945**	-	**34.4**	**0.859**
	PPNMT	28.7	58.4	0.941	-	26.9	0.825
Real-Time Networks	pre-trained	14.3	43.4	0.912	-	17.2	0.770
	fine-tuned	**15.8**	**44.9**	**0.916**	-	**19.6**	**0.788**
	PPNMT	13.0	44.1	0.909	-	16.7	0.760
Smart Phone	pre-trained	19.5	48.3	0.931	-	22.1	0.797
	fine-tuned	**22.3**	**50.6**	**0.934**	-	**27.3**	**0.825**
	PPNMT	19.6	48.6	0.929	-	21.2	0.785

Table 4: Fine-grained tech domain adaptation results for the two baselines and the proposed method (PPNMT) in a full-data setting. The BLEU score was not used for evaluating English-Chinese translations because of its unsuitability for Chinese.

6.3 Technical Domain Adaptation

The results of the domain adaptation on the fine-grained tech domains included in the FGraDA dataset can be seen in table 4. The pre-trained baseline performs poorly, especially with the Real-Time Networks and Smart Phone domains. Fine-tuning the model on the domain data achieves a moderate improvement for all domains. However, PPNMT almost universally hurts the performance, except for a negligible increase in the smartphone domain for Chinese-English translation.

Since no domain-specific evaluation is available and the domains are so similar to each other, it is impossible to judge if the adapted translations adhere better to the domain.

Method	Formal			
	BLEU	chrF	BERTScore	Acc.
pre-trained	33.5	57.1	0.874	0.519
PPNMT_0	36.7	59.4	0.879	0.905
fine-tuned_1	35.8	58.7	0.879	0.664
fine-tuned_3	37.1	59.5	0.882	0.767
fine-tuned_5	37.6	59.9	0.882	0.835
fine-tuned_{10}	38.0	60.1	0.883	0.886
fine-tuned_{20}	39.1	61.1	0.886	0.933
fine-tuned_{50}	40.1	62.3	0.890	0.983
fine-tuned_{full}	43.9	64.6	0.903	0.998

Table 5: Few-shot and zero-shot formality adaptation results. The PPNMT model here uses a very simple, manually defined bag-of-words; therefore, it is a zero-shot approach, not requiring any training data. The fine-tuned models were trained on an increasing amount of training examples.

6.4 Zero-Shot and Few-Shot Adaptation

The results for zero-shot and few-shot domain adaptation for formality can be seen in table 5. Few-shot fine-tuning achieves parity with the proposed method (using the very simple ”Sie” and ”du” as the positive and negative bags-of-words) very quickly in terms of BLEU score, chrF and BERTScore at only three training instances, although still lagging slightly behind with the formality accuracy. The formality accuracy then is surpassed with 20 training examples.

7 Discussion

Fine-tuning outperforms the proposed method in a full-data setting. This is unsurprising because fine-tuning is known to be a very effective domain-adaptation method. The proposed method, however, shows some interesting qualities.

7.1 Adapting to Extrinsic Domains vs Intrinsic Domains

The results support PPNMT being better at adapting to extrinsic domains (formality and to a lesser extent gender) rather than intrinsic domains (fine-grained tech domains), possibly due to their properties being easier to capture with something as simple as a bag-of-words.

Despite having extensive domain term dictionaries for the fine-grained tech domains (which, in theory, is the perfect bag-of-words for the domain, especially when contrasting it to the dictionaries of the other domains), PPNMT hurts all of the quality metrics. This could be partially explained by FGraDA being a challenging dataset, with all four domains being too close to each other for the method to be able to make a meaningful difference.

7.2 PPNMT does not Give the Model New Knowledge

Interestingly, for formality, the language pair with the worst performing pre-trained model (English-Hindi) has a slight decrease in BLEU, chrF, and BERTScore for PPNMT while still increasing the formality accuracy. That could indicate that the method is better at controlling more fluent models. This could be explained as the perturbation method not imbuing the model with any new knowledge (like fine-tuning would) but instead just channelling the pre-existing knowledge that it already has from pre-training in a different direction. A more performant model could, therefore, be easier to push without hurting the performance because the model already has the capabilities for outputting perfectly fluent formal or informal versions of a sentence.

A similar result can be seen with the gender adaptation, where the initial pre-trained model performs poorly (also seemingly biased towards using the male inflections due to the higher scores for the fine-tuned baseline in male texts). Pushing it toward male translations does not do anything meaningful; it just slightly hurts the performance, which could be caused by the noise that the perturbation method introduces that the model's meagre language modelling capabilities cannot overcome. However, pushing toward female translations slightly improves the results, which could be due to the perturbations helping overcome the inherent male bias that the model has.

The poor performance on the FGraDA dataset could also be explained by the used pre-trained Chinese-English and English-Chinese models not being fluent enough by themselves (as can be seen by the low chrF scores for Chinese-English and low BLEU scores for two of the domains for English-Chinese translation). If the models do not already know the domain-specific term translations, then it is possible that no amount of pushing in the decoder could make it output that translation.

The fact that the BERTScores do not change significantly from the pre-trained baseline to the PPNMT model also supports this: just using different surface-level word choices does not meaningfully affect the semantic meanings of the translations and does not give the model translation capability that it did not already possess.

7.3 Zero-shot formality adaptation

The proposed method has an edge in a zero-shot setting for formality adaptation. This edge, however, is relatively small. Even at quite low few-shot settings with 20 training instances, fine-tuning starts outperforming PPNMT in terms of domain accuracy.

The fine-tuned model adapts first to the inherent domain that is present in the CoCoA-MT dataset, namely, spoken dialogue (which is common to all three of the included subdomains of telephony, call centre and topical chat), since the BLEU score, chrF and BERTScore improve on that of the proposed method before the actual formality accuracy. This is likely because the bag-of-words attribute model

captures a very narrow and focused type of domain compared to the fine-tuning procedure, which is good at capturing all patterns in the data, including ones unrelated to formality.

However, even if this adaptation to the inherent domain is ignored, fine-tuning achieves much better domain accuracy with only a few examples. The proposed method's main advantage is that the model did not need to be fine-tuned, saving the space that storing domain-specific versions of the model would require.

7.4 Qualitative Analysis

Despite the simplicity of the bag-of-word models, quite sophisticated adaptation is achieved in some translations. Dathathri et al. (2020) already note that the bag-of-word attribute models also push the model to generate words related to words in the bag, not just the terms contained within it. The same effect is also seen in machine translation. In the zero-shot setting, even with a bag-of-words as simple as "Sie" (as the positive bag-of-words) and "du" (as the negative bag-of-words), the model correctly translates other inflections of these pronouns, as can be seen in table 6, with the model correctly translating the possessive pronoun. Since none of the words in the sentence and the bag-of-words share tokens, the model must have been pushed indirectly.

Source:	Who is your favorite Baseball team?
Unadapted:	Wer ist **dein** Lieblings-Baseball-Team?
PPNMT:	Wer ist **Ihr** Lieblings-Baseball-Team?

Table 6: "dein" is changed to "Ihr", despite the bag-of-words containing neither "dein" nor "Ihr"

The instances in the CoCoA-MT dataset focus only on changing the formality through pronouns (and the resulting grammar changes), and there are no instances where the root verb is changed. This is a result of the way the dataset was created. However, formality is more than just using the formal or informal pronouns. Different

verbs can mean the same thing but have slightly different formalities, e.g. more indirect and passive verbs tend to be more formal than their direct counterparts. In table 7 and table 8, two examples are shown where the PPNMT method also changes the verb into a more passive version, whereas the fine-tuned baseline only changes the pronoun.

Source:	Now, it's easier to **get your money back**, but back then it was a whole big thing and you know.
Unadapted:	Es ist einfacher, **dein** Geld **zurückzuholen**, aber damals war es eine große Sache und du weißt schon.
PPNMT:	Es ist einfacher, **Ihr** Geld **zurück zu bekommen**, aber damals war es eine große Sache und Sie wissen schon
Fine-tuned:	Also, es ist einfacher, **Ihr** Geld **zurückzuholen**, aber damals war es eine ganz große Sache und wissen Sie,

Table 7: The more active "zurückzuholen" (take back) is replaced by a more polite passive "zurück zu bekommen" (receive back). The fine-tuned baseline only changes the pronoun.

Source:	What do I like about my work?
Unadapted:	Was **mag ich** an meiner Arbeit?
PPNMT:	Was **gefällt mir** an meiner Arbeit?
Fine-tuned:	Was **mag ich** an meiner Arbeit?

Table 8: The more passive "gefällt mir" (is appealing to me) is used instead of the more active "mag ich" (I like), which is slightly more formal. The fine-tuned baseline does not make such a change.

These cases, however, are the exception rather than the rule. The fine-tuned baseline performs admirably when it comes to using the correct pronouns, whereas the proposed method sometimes fails to change the sentence at all from an informal

translation. This could be a question of adjusting the hyperparameters. However, it is also possible to push too far in some instances and make the model output nonsense.

7.5 Controllability of the Method

Many hyperparameters may be adjusted to change how hard the model is pushed toward a domain, such as the step size or iteration count and the weighting for post-norm fusion. Despite this, it is hard to tune the method. If the perturbations are too weak, the translations often remain unchanged. However, if the perturbations are too strong, the quality suddenly degrades, and the output becomes repetitive. Due to the categorical nature of text (either a word is translated one way or another), finding a good middle ground is challenging, and it may be impossible to have a one-size-fits-all solution.

Some things might be done to mitigate this. With a good domain classifier, the step size might be dynamically increased until some threshold is reached and the domain changes. Furthermore, if repetition is detected, the step size might be tuned back and the translation retried.

8 Conclusion and Future Work

This section summarizes the key insights from this thesis.

In summary, the proposed method can only channel knowledge that the model already has (because, unlike fine-tuning, it does not expose the model to any new unseen examples). This means that it would be expected to perform much better at adapting to linguistic aspects (formality, gender, and so on) since a model trained on a sufficiently large corpus would necessarily have seen many examples of different formality and different gendered inflections. This is observed for formality and on a smaller scale for gender.

This could be extrapolated to say that a large enough model has seen enough instances from different intrinsic domains, so a larger and better English-Chinese translation model might work better with the proposed method. However, with the used model, PPNMT only introduces noise and hurts performance.

While this method is reasonably good in a zero-shot setting, fine-tuning starts outperforming it with very few training instances. Despite this, the proposed method with formality adaptation shows some interesting emergent behaviour, sometimes changing the translation's formality on a deeper level that goes beyond pronouns.

Lastly, despite all the levers it gives the user to control the perturbation process, it is not easy to tune this method for translation. In the text-generation task, the model has a lot more time to be pushed towards a specific domain slowly. However, for translation (especially when translating individual sentences at a time), the model needs to get there much faster and be pushed more strongly.

Research questions Four research questions were put forward in this thesis. They will be restated and answered here.

- **Can the proposed plug-and-play method improve machine translation output on specific domains?**

  The results indicate that the answer to this is yes, PPNMT can improve in-domain performance over the pre-trained model, as was shown for formality.

- **How does the performance of this kind of plug-and-play domain adaptation method compare to traditional fine-tuning methods?**

  Traditional fine-tuning significantly outperforms the proposed method in terms of raw performance.

- **How does this method compare to fine-tuning when dealing with complex topic domains like technical literature?**

  The proposed method does not seem suitable for complex topic domains, at least for the dataset and model used, whereas fine-tuning shows some moderate improvements.

- **How does it compare to fine-tuning in low-data settings?**

  The zero-shot performance is one of the few aspects where PPNMT has an advantage, allowing even a very simple manually constructed domain model to show some improvement over the baseline.

Future work In future work, it would be useful to consolidate the method by reducing the number of hyperparameters to a smaller set and doing a more thorough hyperparameter search. Secondly, it would be interesting to investigate this method on larger and better translation models to see if the assumption that the method only channels what the model already knows holds. This could be tested by fine-tuning a model on multiple domains and then applying PPNMT on the same domains plus a similar unseen domain. Lastly, only bag-of-words attribute models were considered. It would be interesting to see how well this method performs with small neural networks as domain classifiers. It is possible that this approach would improve performance in domains that are not easy to capture with keywords.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

Bashar Alhafni, Nizar Habash, and Houda Bouamor. The Arabic parallel gender corpus 2.0: Extensions and analyses. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 1870–1884, Marseille, France, June 2022. European Language Resources Association. URL `https://aclanthology.org/2022.lrec-1.199`.

Antonios Anastasopoulos, Alessandro Cattelan, Zi-Yi Dou, Marcello Federico, Christian Federmann, Dmitriy Genzel, Franscisco Guzmán, Junjie Hu, Macduff Hughes, Philipp Koehn, Rosie Lazar, Will Lewis, Graham Neubig, Mengmeng Niu, Alp Öktem, Eric Paquin, Grace Tang, and Sylwia Tur. TICO-19: the translation initiative for COvid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, Online, December 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlpcovid19-2.5. URL `https://aclanthology.org/2020.nlpcovid19-2.5`.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1098. URL `https://aclanthology.org/D17-1098`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

Vidhisha Balachandran, Artidoro Pagnoni, Jay Yoon Lee, Dheeraj Rajagopal, Jaime Carbonell, and Yulia Tsvetkov. StructSum: Summarization via structured representations. In *Proceedings of the 16th Conference of the European Chapter of*

the Association for Computational Linguistics: Main Volume, pages 2575–2585, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.220. URL https://aclanthology.org/2021.eacl-main.220.

Ankur Bapna and Orhan Firat. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1538–1548, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1165. URL https://aclanthology.org/D19-1165.

Rachel Bawden, Giorgio Maria Di Nunzio, Cristian Grozea, Inigo Jauregi Unanue, Antonio Jimeno Yepes, Nancy Mah, David Martinez, Aurélie Névéol, Mariana Neves, Maite Oronoz, Olatz Perez-de Viñaspre, Massimo Piccardi, Roland Roller, Amy Siu, Philippe Thomas, Federica Vezzani, Maika Vicente Navarro, Dina Wiemann, and Lana Yeganova. Findings of the WMT 2020 biomedical translation shared task: Basque, Italian and Russian as new additional languages. In *Proceedings of the Fifth Conference on Machine Translation*, pages 660–687, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.76.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944966.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt

Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2301. URL https://aclanthology.org/W16-2301.

Thorsten Brants and Alex Franz. Web 1t 5-gram version 1 ldc2006t13. Web Download, 2006.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/D07-1090.

Denny Britz, Quoc Le, and Reid Pryzant. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4712. URL https://aclanthology.org/W17-4712.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2): 79–85, 1990. URL https://aclanthology.org/J90-2002.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993. URL https://aclanthology.org/J93-2003.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 40–46, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3205. URL `https://aclanthology.org/W17-3205`.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219873. URL `https://aclanthology.org/P05-1033`.

Mara Chinea-Ríos, Álvaro Peris, and Francisco Casacuberta. Adapting neural machine translation with parallel synthetic data. In *Proceedings of the Second Conference on Machine Translation*, pages 138–147, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4714. URL `https://aclanthology.org/W17-4714`.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014a. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL `https://aclanthology.org/W14-4012`.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014b. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390177. URL https://doi.org/10.1145/1390156.1390177.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1198. URL https://aclanthology.org/P18-1198.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4715. URL https://aclanthology.org/W17-4715.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989. ISSN 0932-4194. doi: 10.1007/BF02551274. URL http://dx.doi.org/10.1007/BF02551274.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In

C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

Wim De Mulder, Steven Bethard, and Marie-Francine Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61–98, 2015. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2014.09.005. URL https://www.sciencedirect.com/science/article/pii/S088523081400093X.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990. doi: https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

Anne-Marie Derouault and Bernard Merialdo. Natural language modeling for phoneme-to-text transcription. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):742–749, 1986. doi: 10.1109/TPAMI.1986.4767855.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of Wikipedia: Knowledge-powered conversational agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. Training neural machine translation to apply terminology constraints. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3063–3068, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1294. URL https://aclanthology.org/P19-1294.

Zi-Yi Dou, Xinyi Wang, Junjie Hu, and Graham Neubig. Domain differential adaptation for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 59–69, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5606. URL https://aclanthology.org/D19-5606.

Akiko Eriguchi, Shufang Xie, Tao Qin, and Hany Hassan. Building multilingual machine translation systems that serve arbitrary XY translations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 600–606, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.44. URL https://aclanthology.org/2022.naacl-main.44.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1315. URL https://aclanthology.org/P19-1315.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages

567–573, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2090. URL https://aclanthology.org/P17-2090.

M. Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4713. URL https://aclanthology.org/W17-4713.

George Foster, Cyril Goutte, and Roland Kuhn. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459, Cambridge, MA, October 2010. Association for Computational Linguistics. URL https://aclanthology.org/D10-1044.

Markus Freitag and Yaser Al-Onaizan. Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*, 2016.

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, feb 1994. ISSN 0898-9788.

Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and li Deng. Stylenet: Generating attractive visual captions with styles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 955–964, 07 2017. doi: 10.1109/CVPR.2017.108.

Qin Gao and Stephan Vogel. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL https://aclanthology.org/W08-0509.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *arXiv preprint*, 2022.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1243–1252. JMLR.org, 2017.

Dmitriy Genzel, Jakob Uszkoreit, and Franz Och. "poetic" statistical machine translation: Rhyme and meter. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 158–166, Cambridge, MA, October 2010. Association for Computational Linguistics. URL `https://aclanthology.org/D10-1016`.

Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. A knowledge-grounded neural conversation model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.11977. URL `https://ojs.aaai.org/index.php/AAAI/article/view/11977`.

Mitchell Gordon and Kevin Duh. Distill, adapt, distill: Training small, in-domain models for neural machine translation. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 110–118, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.ngt-1.12. URL `https://aclanthology.org/2020.ngt-1.12`.

Rong Gu, Min Chen, Wenjia Yang, Chunfeng Yuan, and Yihua Huang. Seal: Efficient training large scale statistical machine translation models on spark. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 118–125, 2018. doi: 10.1109/PADSW.2018.8644562.

Shuhao Gu, Yang Feng, and Qun Liu. Improving domain adaptation translation with domain invariant and specific information. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3081–3091, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1312. URL `https://aclanthology.org/N19-1312`.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2081. URL https://aclanthology.org/N18-2081.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567, 2018. URL http://arxiv.org/abs/1803.05567.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1141. URL https://aclanthology.org/P17-1141.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*. OpenReview.net, 2020. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2020.html#HoltzmanBDFC20.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Re-*

search, pages 2790–2799. PMLR, 2019. URL http://dblp.uni-trier.de/db/conf/icml/icml2019.html#HoulsbyGJMLGAG19.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. Improved lexically constrained decoding for translation and monolingual rewriting. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 839–850, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1090. URL https://aclanthology.org/N19-1090.

Junjie Hu, Mengzhou Xia, Graham Neubig, and Jaime Carbonell. Domain adaptation of neural machine translation by lexicon induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2989–3001, Florence, Italy, July 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1286. URL https://aclanthology.org/P19-1286.

Weipeng Huang, Xingyi Cheng, Kunlong Chen, Taifeng Wang, and Wei Chu. Towards fast and accurate neural Chinese word segmentation with multi-criteria learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2062–2072, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.186. URL https://aclanthology.org/2020.coling-main.186.

F. Jelinek. A real-time, isolated-word, speech recognition system for dictation transcription. In *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 10, pages 858–861, 1985. doi: 10.1109/ICASSP.1985.1168313.

Haoming Jiang, Chen Liang, Chong Wang, and Tuo Zhao. Multi-domain neural machine translation with word-level adaptive layer-wise domain mixing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1823–1834, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.165. URL https://aclanthology.org/2020.acl-main.165.

Michael I. Jordan. Chapter 25 - serial order: A parallel distributed processing approach. In John W. Donahoe and Vivian Packard Dorsel, editors, *Neural-Network Models of Cognition*, volume 121 of *Advances in Psychology*, pages 471–495. North-Holland, 1997. doi: https://doi.org/10.1016/S0166-4115(97)80111-2. URL https://www.sciencedirect.com/science/article/pii/S0166411597801112.

Shafiq Joty, Hassan Sajjad, Nadir Durrani, Kamla Al-Mannai, Ahmed Abdelali, and Stephan Vogel. How to avoid unwanted pregnancies: Domain adaptation using neural network models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1259–1270, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1147. URL https://aclanthology.org/D15-1147.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-4020. URL https://aclanthology.org/P18-4020.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (3rd Edition draft)*. Prentice-Hall, Inc., USA, Jan. 2023. ISBN 0131873210.

Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text*

(W-NUT 2019), pages 42–47, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5506. URL https://aclanthology.org/D19-5506.

Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoust. Speech Signal Process.*, 35: 400–401, 1987. URL https://api.semanticscholar.org/CorpusID:6555412.

Tobias Kaufmann. *A rule-based language model for speech recognition*. PhD thesis, Swiss Federal Institute of Technology, Zürich, 2009. URL https://doi.org/10.3929/ETHZ-A-006035932.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=7wCBOfJ8hJM.

Huda Khayrallah, Gaurav Kumar, Kevin Duh, Matt Post, and Philipp Koehn. Neural lattice search for domain adaptation in machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 20–25, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://aclanthology.org/I17-2004.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1032. URL https://aclanthology.org/D16-1032.

R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1, 1995. doi: 10.1109/ICASSP.1995.479394.

Catherine Kobus, Josep Crego, and Jean Senellart. Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances*

in Natural Language Processing, RANLP 2017, pages 372–378, Varna, Bulgaria, September 2017. INCOMA Ltd. doi: 10.26615/978-954-452-049-6_049. URL https://doi.org/10.26615/978-954-452-049-6_049.

Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3204. URL https://aclanthology.org/W17-3204.

Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133, 2003. URL https://aclanthology.org/N03-1017.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/P07-2045.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. RankGen: Improving text generation with large ranking models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 199–232, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.15. URL https://aclanthology.org/2022.emnlp-main.15.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018.

Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Seungjun Lee, Hyeonseok Moon, Chanjun Park, and Heuiseok Lim. Improving formality-sensitive machine translation using data-centric approaches and prompt engineering. In *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*, pages 420–432, Toronto, Canada (in-person and online), July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.iwslt-1.40. URL https://aclanthology.org/2023.iwslt-1.40.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Dingcheng Li, Zheng Chen, Eunah Cho, Jie Hao, Xiaohu Liu, Fan Xing, Chenlei Guo, and Yang Liu. Overcoming catastrophic forgetting during domain adaptation of seq2seq language generation. In *Proceedings of the 2022 Conference of the*

North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5441–5454, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.398. URL https://aclanthology.org/2022.naacl-main.398.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March 2009. Association for Computational Linguistics. URL https://aclanthology.org/W09-0424.

Jianze Liang, Chengqi Zhao, Mingxuan Wang, Xipeng Qiu, and Lei Li. Finding sparse structures for domain specific neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13333–13342, May 2021. doi: 10.1609/aaai.v35i15.17574. URL https://ojs.aaai.org/index.php/AAAI/article/view/17574.

Jin Liu, Xin Zhang, Xiaohu Tian, Jin Wang, and Arun Kumar. A novel domain adaption approach for neural machine translation. *International Journal of Computational Science and Engineering*, 22:445, 01 2020a. doi: 10.1504/IJCSE.2020.109404.

Yang Liu and Mirella Lapata. Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75, 2018. doi: 10.1162/tacl_a_00005. URL https://aclanthology.org/Q18-1005.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL http://arxiv.org/abs/1907.11692. cite arxiv:1907.11692.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training

for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020b. doi: 10.1162/tacl_a_00343. URL https://aclanthology.org/2020.tacl-1.47.

Minh-Thang Luong and Christopher Manning. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 76–79, Da Nang, Vietnam, December 3-4 2015. URL https://aclanthology.org/2015.iwslt-evaluation.11.

Ji Ma, Kuzman Ganchev, and David Weiss. State-of-the-art Chinese word segmentation with Bi-LSTMs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4908, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1529. URL https://aclanthology.org/D18-1529.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1006. URL https://aclanthology.org/K16-1006.

Elise Michon, Josep Crego, and Jean Senellart. Integrating domain terminology into neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3925–3937, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.348. URL https://aclanthology.org/2020.coling-main.348.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun,

editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proc. Interspeech 2010*, pages 1045–1048, 2010. doi: 10.21437/Interspeech.2010-343.

Hideya Mino, Hideki Tanaka, Hitoshi Ito, Isao Goto, Ichiro Yamada, and Takenobu Tokunaga. Content-equivalent translated parallel news corpus and extension of domain adaptation for NMT. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3616–3622, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.445.

Maria Nadejde, Anna Currey, Benjamin Hsu, Xing Niu, Marcello Federico, and Georgiana Dinu. CoCoA-MT: A dataset and benchmark for contrastive controlled MT with application to formality. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 616–632, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.47. URL https://aclanthology.org/2022.findings-naacl.47.

S. Nießen, S. Vogel, H. Ney, and C. Tillmann. A DP based search algorithm for statistical machine translation. In *COLING 1998 Volume 2: The 17th International Conference on Computational Linguistics*, 1998. URL https://aclanthology.org/C98-2153.

Xing Niu, Marianna Martindale, and Marine Carpuat. A study of style in machine translation: Controlling the formality of machine translation output. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2814–2819, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1299. URL https://aclanthology.org/D17-1299.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia-Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation. 07 2022. doi: 10.48550/arXiv.2207.04672.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. Definition modeling: Learning to define word embeddings in natural language. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3259–3266. AAAI Press, 2017.

Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong, October 2000. Association for Computational Linguistics. doi: 10.3115/1075218.1075274. URL `https://aclanthology.org/P00-1056`.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999. URL `https://aclanthology.org/W99-0604`.

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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://aclanthology.org/P02-1040`.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. A plug-and-play method for controlled text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3973–3997, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.334. URL https://aclanthology.org/2021.findings-emnlp.334.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Minh Quang Pham, Josep Maria Crego, François Yvon, and Jean Senellart. A study of residual adapters for multi-domain neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 617–628, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.72.

MinhQuang Pham, Josep Crego, François Yvon, and Jean Senellart. Generic and specialized word embeddings for multi-domain machine translation. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong, November 2-3 2019. Association for Computational Linguistics. URL https://aclanthology.org/2019.iwslt-1.26.

Ngoc-Quan Pham, German Kruszewski, and Gemma Boleda. Convolutional neural

network language models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1153–1162, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1123. URL https://aclanthology.org/D16-1123.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. URL https://aclanthology.org/W15-3049.

Shrimai Prabhumoye, Chris Quirk, and Michel Galley. Towards content transfer through grounded text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2622–2632, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1269. URL https://aclanthology.org/N19-1269.

Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. Exploring controllable text generation techniques. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1–14, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.1. URL https://aclanthology.org/2020.coling-main.1.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th*

International Conference on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/radford23a.html.

Alexey Romanov, Anna Rumshisky, Anna Rogers, and David Donahue. Adversarial decomposition of text representation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 815–825, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1088. URL https://aclanthology.org/N19-1088.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Yonatan Belinkov, and Stephan Vogel. Neural machine translation training in a multi-domain scenario. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 66–73, Tokyo, Japan, December 14-15 2017. International Workshop on Spoken Language Translation. URL https://aclanthology.org/2017.iwslt-1.10.

Danielle Saunders. Domain adaptation and multi-domain adaptation for neural machine translation: A survey. *Journal of Artificial Intelligence Research*, 75: 351–424, 2022.

Danielle Saunders, Felix Stahlberg, and Bill Byrne. UCAM biomedical translation at WMT19: Transfer learning multi-domain ensembles. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 169–174, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5421. URL https://aclanthology.org/W19-5421.

Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California, June 2016a. Association for Computational Linguistics. doi: 10.18653/v1/N16-1005. URL https://aclanthology.org/N16-1005.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August 2016b. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. URL https://aclanthology.org/P16-1009.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016c. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation*, pages 389–399, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4739. URL https://aclanthology.org/W17-4739.

Noah A. Smith. Contextual word representations: Putting words into computers. *Commun. ACM*, 63(6):66–74, may 2020. ISSN 0001-0782. doi: 10.1145/3347145. URL https://doi.org/10.1145/3347145.

Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. Code-switching for enhancing NMT with pre-specified translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 449–459, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1044. URL https://aclanthology.org/N19-1044.

Emmanouil Stergiadis, Satendra Kumar, Fedor Kovalev, and Pavel Levin. Multi-domain adaptation in neural machine translation through multidimensional tagging. In *Proceedings of Machine Translation Summit XVIII: Users and Providers Track*, pages 396–420, Virtual, August 2021. Association for Machine Translation in the Americas. URL https://aclanthology.org/2021.mtsummit-up.27.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Sander Tars and Mark Fishel. Multi-domain neural machine translation. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 279–288, Alicante, Spain, May 2018. URL https://aclanthology.org/2018.eamt-main.26.

Yuanhe Tian, Yan Song, Fei Xia, Tong Zhang, and Yonggang Wang. Improving Chinese word segmentation with wordhood memory networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8274–8285, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.734. URL https://aclanthology.org/2020.acl-main.734.

Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey, May 2012. European

Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf.

Jörg Tiedemann and Santhosh Thottingal. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal, November 2020. European Association for Machine Translation. URL https://aclanthology.org/2020.eamt-1.61.

Vaibhav Vaibhav, Sumeet Singh, Craig Stewart, and Graham Neubig. Improving robustness of machine translation with synthetic noise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1916–1920, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1190. URL https://aclanthology.org/N19-1190.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1147. URL https://aclanthology.org/D17-1147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Lucas Nunes Vieira, Carol O'Sullivan, Xiaochun Zhang, and Minako O'Hagan. Machine translation in society: insights from uk users. *Language Resources and Evaluation*, 57(2):893–914, Jun 2023. ISSN 1574-0218. doi: 10.1007/s10579-022-09589-1. URL https://doi.org/10.1007/s10579-022-09589-1.

David Vilar, Daniel Stein, Matthias Huck, and Hermann Ney. Jane: Open source hierarchical translation, extended with reordering and lexicon models. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 262–270, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/W10-1738.

Stephan Vogel, Franz Och, S. Nieen, Hassan Sawaf, Christoph Tillmann, and Hermann Ney. Statistical methods for machine translation. 09 2000.

A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, March 1989. ISSN 0096-3518. doi: 10.1109/29.21701.

Ke Wang, Hang Hua, and Xiaojun Wan. *Controllable Unsupervised Text Attribute Transfer via Editing Entangled Latent Representation*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 560–566, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2089. URL https://aclanthology.org/P17-2089.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1199. URL https://aclanthology.org/D15-1199.

P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. doi: 10.1109/5.58337.

Jason D. Williams, Iker Arizmendi, and Alistair Conkie. Demonstration of at&t "let's go": A production-grade statistical spoken dialog system. In *2010 IEEE Spoken Language Technology Workshop*, pages 157–158, 2010. doi: 10.1109/SLT.2010.5700839.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`.

Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. Vocabulary learning via optimal transport for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7361–7373, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.571. URL `https://aclanthology.org/2021.acl-long.571`.

Jitao Xu, Josep Crego, and Jean Senellart. Lexical micro-adaptation for neural machine translation. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong, November 2-3 2019. Association for Computational Linguistics. URL `https://aclanthology.org/2019.iwslt-1.27`.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Sanghyun Yi, Rahul Goel, Chandra Khatri, Alessandra Cervone, Tagyoung Chung, Behnam Hedayatnia, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. To-

wards coherent and engaging spoken dialog response generation using automatic conversation evaluators. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 65–75, Tokyo, Japan, October–November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8608. URL https://aclanthology.org/W19-8608.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 447–457, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1041. URL https://aclanthology.org/D18-1041.

Daniel Zhang, Jiang Yu, Pragati Verma, Ashwinkumar Ganesan, and Sarah Campbell. Improving machine translation formality control with weakly-labelled data augmentation and post editing strategies. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 351–360, Dublin, Ireland (in-person and online), May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.iwslt-1.32. URL https://aclanthology.org/2022.iwslt-1.32.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1120. URL https://aclanthology.org/N18-1120.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. Curriculum learning for domain adaptation in neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1903–1915, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1189. URL https://aclanthology.org/N19-1189.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. Adaptive nearest neighbor machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 368–374, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.47. URL https://aclanthology.org/2021.acl-short.47.

Kangyan Zhou, Shrimai Prabhumoye, and Alan W Black. A dataset for document grounded conversations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 708–713, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1076. URL https://aclanthology.org/D18-1076.

Wenhao Zhu, Shujian Huang, Tong Pu, Pingxuan Huang, Xu Zhang, Jian Yu, Wei Chen, Yanfeng Wang, and Jiajun Chen. FGraDA: A dataset and benchmark for fine-grained domain adaptation in machine translation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6719–6727, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.lrec-1.723.

A Domain-Specific Bags-of-Words

This appendix contains the bags-of-words that were used for the experiments.

A.1 Formality

The formality experiments used a contrastive method to identify a set of 30 formal and 30 informal tokens which were used as the positive and negative bags.

Formal German : [”Ihr”, ”Ihnen”, ”wissen”, ”haben”, ”Haben”, ”Ihrem”, ”Ihren”, ”Ihrer”, ”sich”, ”müssen”, ”sind”, ”n”, ”können”, ”Möge”, ”denken”, ”würden”, ”könnten”, ”mögen”, ”sollten”, ”Wissen”, ”Ihres”, ”kaufen”, ”en”, ”schauen”, ”werden”, ”brauchen”, ”waren”, ”wollen”, ”lassen”, ”benutzen”]

Informal German : [”du”, ”deine”, ”dein”, ”st”, ”dir”, ”weißt”, ”hast”, ”Du”, ”deinem”, ”Hast”, ”dich”, ”deinen”, ”deiner”, ”musst”, ”bist”, ”kannst”, ”Mag”, ”denkst”, ”würdest”, ”könntest”, ”magst”, ”solltest”, ”Hab”, ”Weißt”, ”deines”, ”kauf”, ”schau”, ”brauchst”, ”warst”, ”willst”]

Formal Italian :[”Lei”, ”sua”, ”suoi”, ”può”, ”Le”, ”deve”, ”è”, ”rsi”, ”sue”, ”vuole”, ”potrebbe”, ”r”, ”La”, ”È”, ”rebbe”, ”va”, ”dice”, ”La”, ”sé”, ”conosce”, ”vada”, ”Mi”, ”tras”, ”corre”, ”rà”, ”lo”, ”lasci”, ”potesse”, ”capisce”, ”ricorda”]

Informal Italian : [”tuo”, ”tua”, ”tu”, ”tuoi”, ”te”, ”ti”, ”o”, ”pu”, ”de”, ”rti”, ”Ti”, ”sti”, ”sei”, ”Tu”, ”vuoi”, ”tue”, ”rai”, ”Se”, ”potre”, ”sci”, ”ci”, ”po”, ”ri”, ”tene”, ”mi”, ”tra”, ”sco”, ”rri”, ”mmi”, ”dai”]

Formal Spanish : [”le”, ”tiene”, ”se”, ”ha”, ”puede”, ”Le”, ”se”, ”Ha”, ”piensa”, ”quiere”, ”Su”, ”tenga”, ”compra”, ”está”, ”Cree”, ”va”, ”era”, ”debe”, ”ve”, ”Usted”, ”Sabe”, ”Tiene”, ”Cu”, ”podría”, ”Se”, ”necesita”, ”suyo”, ”entiende”, ”quedarse”, ”Podría”]

Informal Spanish :["tu", "s", "sabes", "te", "tus", "tú", "tienes", "Te", "has", "contigo", "puedes", "ti", "Has", "ste", "piensas", "quieres", "Tu", "tengas", "compras", "estás", "Crees", "vas", "eras", "tuyo", "debes", "ves", "Tú", "Sabes", "Tienes", "podrías"]

Formal French : ["vous", "votre", "vous", "avez", "vos", "Vous", "Avez", "voyez", "savez", "pensez", "pouvez", "z", "iez", "ez", "aimez", "devez", "faites", "passez", "vôtre", "Votre", "êtes", "voulez", "étiez", "allez", "Profitez", "devriez", "pourriez", "A", "im", "Êtes"]

Informal French :["tu", "ton", "tu", "ta", "toi", "tes", "Tu", "As", "vois", "te", "penses", "t", "peux", "aimes", "dois", "fais", "ais", "passe", "es", "e", "veux", "étais", "vas", "Profit", "devrais", "Ta", "pourrais", "Ai", "mes", "Es"]

Formal Hindi : ["आपको", "हैं", "आपका", "आपकी", "आपने", "आपसे", "जाइए", "करें", "करेंगे", "दें", "डालें", "लें", "बताइए", "रहें", "चलिए", "जि", "एँ", "सोचें", "सुनिए", "कीजिए", "आज़माएँ", "होंगे", "रुकें", "रुकिए", "पाएँगे", "जोड़ें", "बताएँ", "जाएँ", "करवाए", "रखें"]

Informal Hindi : ["तुम्हें", "तुम", "हो", "तुम्हारे", "तुम्हारा", "तुम्हारी", "तुमने", "तुमसे", "करो", "जाओ", "बताओ", "करोगे", "दो", "डालो", "लो", "ओ", "चलो", "रुको", "जियो", "तुम्", "सोचो", "सुनो", "आज़मा", "रहो", "होगे", "हारे", "पाओगे", "जोड़ो", "हें", "करवा"]

A.2 Gender

Male Arabic : ["آسف", "سيدي", "متأكد", "كم", "سيد", "تعرف", "عزيزي", "أبي", "رجل", "واثق", "تكون", "سعيدا", "أيها", "اسف", "ذاهب", "قادم", "مسرور", "أسف", "ترى", "خائف", "تدعو", "أنظر", "فتى", "انظر", "معجب", "جاد", "حبيبي", "تعتقد", "تظن", "قلقا"]

Female Arabic : ["تعرفين", "أمي", ,"سيدة", "يني" "كن", "متأكدة", "سيدتي", "سعيدة", "آسفة", "تكوني", "خائفة", "قادمة", "مشغولة", "تعلمين", "ذاهبة", "ينني", "أيتها", "فخورة", "واثقة", "امرأة", "عزيزتي", "تعتقدين" "تري", "تريدين", "إله", "تظنين", "محقة", "مسرورة", "فتاة", "تدعي",]

A.3 Fine-grained tech domains

There are four domains contained in the FGraDA dataset: autonomous vehicles, AI education, real-time networks, and smartphone. For adapting to one of these domains, its bag-of-words was used to push towards and the other bags-of-words were used to push away from.

Autonomous vehicles : ["digital world", "human intervention", "human intervention", "human intervention", "AIoT", "data flow", "life cycle analysis", "training data", "data science", "IDC", "environmental factors", "distributed factories", "data set", "fuel", "parameters", "lifespan", "lidar", "automotive", "anti-shock vibration", "electromagnetic interference", "high-resolution", "vehicle-sharing", "detection distance", "laser", "silicon-based", "low-resolution", "four-line", "functionalities", "adaptive", "resolution", "target detection", "free space detection", "target classification", "localization", "vertical field", "automatic driving", "lane signs", "tires", "vehicle", "ultra-high-resolution", "ranger", "valet parking", "IRT", "industrial consortium", "accuracy", "workflow", "intelligent system", "automated systems", "autonomy level", "knowledge base", "strain elements", "physical agent", "components", "architecture", "critical value", "verification method", "AI", "automotive industry", "groceries", "traffic vulnerable groups", "emerging companies", "dynamic range", "signal", "car symbolize", "mobility services", "disengagement", "intersections", "suburban roads", "city roads", "identification", "modeling", "functional safety", "processor", "label", "occupancy monitoring", "cab", "IR", "road signal", "traffic lights", "solid-state lidar", "optical character recognition", "silicon", "service life", "automated", "operating temperature", "reflectance", "reflective", "stability", "radar", "navigation", "transportation", "interconnection", "user interface", "data gateway", "sustainable development", "standardization", "image

recognition", "frameworks", "edge computing", "energy consumption", "IM", "manufacturing", "manufacturing industry", "NGIT", "intelligence", "assembly line", "riveting", "interconnection", "digital manufacturing", "digital infrastructures", "in-vehicle system", "brands", "collaborative system", "millimeter wave radar", "domain controller", "monocular", "binocular", "redundancy", "steering gear", "brake", "brake", "change lanes", "computing power", "field of view angle", "high-precision", "inertial navigation", "steering", "mass production", "lane", "vehicle-regulation", "in-vehicle", "navigation", "instrument", "feedback", "noise reduction", "lane keeping", "multi-sensor", "deep learning", "computer vision", "positioning", "wheel speedometer", "path", "heavy trucks", "full load", "empty load", "orientation", "ramp", "road condition", "viaduct", "cross-validation", "real-time", "sharp turn", "climbing", "detection distance", "optical flow method", "dynamics", "simulation", "GPS", "LiDAR", "QR code", "traffic accident", "traffic signal", "traffic jam", "traffic congestion", "traffic data", "traffic police", "AI", "artificial intelligence", "face recognition", "sensor", "sensor fusion", "travel efficiency", "prototype", "OEM", "LED", "big data", "navigator", "IMU", "steering wheel", "unmanned vehicle", "unmanned", "radio", "intelligent transportation", "agent", "intelligent robot", "smart car", "intelligent facility", "intelligent vehicle linkage", "intelligent driving", "machine learning", "millimeter-wave radar", "automotive manufacturing", "laser sensor", "LiDAR", "point cloud", "IoT", "environmental perception", "ecosystem", "neural network", "algorithm", "emergency braking", "infrared", "traffic light", "autonomous system", "automatic parking", "automation", "automatic lane change", "self-driving", "autonomous driving", "autopilot", "autonomous vehicle", "self-organization", "pedestrian detection", "video frame", "ultrasonic radar", "path planning", "Internet of vehicles", "ESP", "automotive operating system", "vehicle active system", "vehicle control system", "vehicle inspection", "lane keeping", "lane departure", "lane coordination", "lane detection", "assisted keeping", "assisted driving", "edge device", "road signal", "road condition", "road boundary detection", "cluster", "validation technology", "high resolution", "high-precision map", "highway", "L1", "L2", "L3", "L4", "L5", "intelligent manufacturing", "camera", "complexity", "Industry 4.0", "software", "hardware", "digitalization", "intelligentization", "pixel", "robot",

”chip”, ”semiconductor”, ”manufacturer”, ”patent”, ”electric vehicle”, ”embedded”, ”engine”, ”online car-hailing”, ”circuit board”, ”V2X”, ”DSP”, ”laser rangefinder”, ”cruise control”, ”phased array”, ”bumper”, ”grille”, ”rearview mirror”, ”production line”, ”visualization”, ”LTE”, ”mapping”, ”electromagnetic interference”, ”laser beam”]

AI education : [”robotics”, ”robots”, ”computer”, ”scores”, ”Carnegie”, ”go”, ”conversational systems”, ”education systems”, ”electroencephalography”, ”electrodes”, ”voltage”, ”Virtual reality”, ”granularity”, ”adaptivity”, ”consistency”, ”deferred”, ”humanistic system”, ”hidden layer”, ”vector”, ”frame frequency”, ”adaptability”, ”slide”, ”mutually reinforcing”, ”interrelationships”, ”meta-intelligence”, ”self-motivation”, ”refine”, ”CCTV”, ”dial-up internet”, ”synchronization”, ”interpersonal interaction”, ”metacognition”, ”old-fashioned”, ”decision-making”, ”principal component analysis”, ”linear regression”, ”trainees”, ”mode”, ”multi-modal”, ”data flows”, ”metacognitive”, ”meta-knowledge”, ”meta-subjective”, ”self-regulated learning”, ”learning science”, ”self-regulation”, ”cognitive ability”, ”virtuous circle”, ”panel discussion”, ”interdisciplinary”, ”non-computer science”, ”dual-degree”, ”hands-on courses”, ”personalized”, ”meta-concepts”, ”IP issues”, ”automation”, ”knowledge-driven”, ”triple AI”, ”Moore’s law”, ”going in circles”, ”scalable”, ”scalability”, ”unicorn”, ”autonomous driving”, ”on-demand”, ”blockchain”, ”knowledge portfolio”, ”multidisciplinary”, ”institutional”, ”agnostic”, ”eye-tracking”, ”labor-intensive”, ”leader”, ”psychometric”, ”AI-driven”, ”sharp opinions”, ”white paper”, ”two sigma question”, ”therapist”, ”short-circuiting”, ”hippocampus”, ”emoticons”, ”spreadsheets”, ”contextual”, ”ed tech”, ”knowledge point”, ”redesign”, ”composition”, ”multiplier effects”, ”transformative”, ”autism”, ”substrate”, ”dyad”, ”fine grained”, ”gaze”, ”prosody”, ”credibility”, ”galvanic skin response”, ”word quest”, ”intervention”, ”steering committee”, ”board”, ”Tic Tac Toe Board”, ”affective computing”, ”emotional computing”, ”maker movement”, ”ontology level”, ”offload”, ”quick learner”, ”study slacker”, ”nanoscale”, ”crazy”, ”determine knowledge”, ”Pythagorean theorem”, ”hedge fund”, ”discipline literacy”, ”pan-education”, ”supply-side”, ”demand side”, ”holographic”, ”principals”, ”externalization”, ”outflow”, ”win-win”, ”digi-

tal image", "parent-child happiness", "proverb", "unstructured", "association analysis", "myopia", "cognitive load", "teaching oriented", "one variable quadratic equation", "customer acquisition cost", "curriculum counseling center", "M & A", "lifelong", "strikes a chord", "poor and humble families", "eagle dad", "architect", "comfort zone", "stress zone", "stretch zone", "personalized teaching", "personalized education", "active learning", "cloud service platform", "Internet Plus", "artificial intelligence", "AI", "man-machine cooperation", "algebra", "geometry", "chemistry", "prototype", "vector representation", "online training", "online education", "big data", "learning efficiency", "learning objective", "learning trajectory", "learning situation analysis", "student interaction", "assign homework", "reinforcement learning", "EQ", "MOOC", "TOEFL", "critical thinking", "instructor", "schedule arrangement", "education industrial chain", "education model", "educational resources", "digital education", "mathematics", "data mining", "data science", "IQ", "smart classroom", "intelligent evaluation", "intelligent question bank", "machine learning", "machine marking", "campus guard", "campus management", "attention analysis", "deep learning", "physics", "knowledge map", "knowledge retrieval", "knowledge point", "neuroscience", "neural network", "personal teacher", "FAQ", "algorithm", "accurate matching", "essential-qualities-oriented education", "practice feedback", "programming", "network teaching", "online classes", "exam results", "chatbot", "EEG", "automatic grading", "automation", "self-efficacy", "self-motivation", "natural language processing", "adaptive learning", "adaptive teaching", "virtual teaching", "virtual reality", "VR", "virtual teacher", "visual tracking", "problem solving", "computer vision", "speech recognition", "classroom teaching", "curriculum", "tutoring", "causes of mistakes", "domain model", "higher education", "sigma", "university", "concept", "science", "professor", "subject", "accuracy", "database", "questions and answers", "PE", "educator", "simultaneous interpretation", "humanity", "psychology", "sociology", "public school", "private school", "leadership", "economics", "training class", "final exam", "human-computer interaction", "science experiment", "doctor", "preschool", "knowledge base", "learning material", "dual degree", "undergraduate", "postgraduate", "logical thinking", "continuing education", "professional training", "lesson plan", "Chinese", "kindergarten",

”do exercise”, ”teach students in accordance with their aptitude”, ”textbook”, ”K-12”]

Real-time networks : [”real time communication”, ”Smart speakers”, ”embedded”, ”AR”, ”VR”, ”sale force”, ”human resources”, ”workflow”, ”partnership”, ”RTC”, ”WebRTC”, ”world wide web”, ”html”, ”web payment”, ”augmented reality”, ”peer to peer connection”, ”nickname”, ”transmission”, ”transit”, ”data channel”, ”VP eight”, ”ht 64”, ”codex”, ”encryption”, ”encrypted”, ”use cases”, ”specification”, ”captioning”, ”status”, ”token”, ”APP”, ”blocking”, ”monetization”, ”gamer”, ”communication stack”, ”session”, ”server”, ”misconfiguration”, ”turn”, ”turn”, ”dealer”, ”UCP”, ”DCP”, ”monitor”, ”signaling”, ”production-based”, ”cellular network”, ”applications”, ”logic of the signaling”, ”security patches”, ”WIFI”, ”IP”, ”logging”, ”use cases”, ”SMS”, ”API”, ”spinning bikes”, ”delay”, ”GitHub”, ”AV1”, ”5G”, ”SIM”, ”URLC”, ”live stream”, ”live broadcast”, ”frame”, ”influencer marketing”, ”live booth”, ”immersion”, ”medium-intensity”, ”complaint rate”, ”renewal rate”, ”ultra-low latency”, ”tone quality”, ”cooperation of experts and teachers”, ”full-stack”, ”GDPR”, ”fraudulent”, ”damage assessment”, ”industrial compliance”, ”retention rate”, ”stuttering”, ”router”, ”architects”, ”perceptual coding”, ”resolution”, ”image enhancement”, ”Topology”, ”Generative Adversarial Nets”, ”noise reduction”, ”downlink”, ”ROI”, ”style conversion”, ”segmentation”, ”emotional computing”, ”translation”, ”left-behind children”, ”defense”, ”CPU”, ”GPU”, ”NPU”, ”SDRTN”, ”super resolution algorithm”, ”residual”, ”multiscale”, ”discriminator”, ”gradient-based”, ”regularization”, ”p-relu”, ”convolution”, ”visual aid system”, ”input method”, ”multi-modal”, ”emoji”, ”media convergence”, ”C-terminal”, ”anchor”, ”overseas returnee”, ”architecture”, ”near-field”, ”far field”, ”interference”, ”reverberation”, ”recording pen”, ”simultaneous interpretation”, ”retrieval”, ”semantic”, ”TTS”, ”roundtable forum”, ”terminal”, ”product capability”, ”flow”, ”VC”, ”graphic and character community”, ”monetization rate”, ”portal”, ”internet celebrity”, ”web celebrity”, ”bilateral verification”, ”zone log”, ”fault tolerance”, ”usage path”, ”cycle time”, ”Lao Tie”, ”inclusiveness”, ”network effect”, ”ecommerce”, ”e-commerce”, ”data traffic”, ”5G”, ”H264”, ”IP address”, ”TCP”, ”UDP”, ”WebAssembly”, ”We-

bRTC", "WebSocket", "packet loss", "middleware", "QR code", "cloud game", "internet", "artificial intelligence", "AI", "human-computer interaction", "transport protocol", "transport layer", "signal", "information flow", "channel", "shared memory", "compatibility", "content creation", "function library", "instant messaging", "handle", "mukbang", "throughput", "AR", "multimodal", "media resource", "real-time interaction", "real-time application", "real-time data", "real-time communication", "P2P connection", "bandwidth", "concurrent", "latency", "congestion", "interface", "operating system", "log", "smart speaker", "module", "streaming media", "traffic", "browser", "deep learning", "IoT", "picture quality", "live streaming", "short video", "social game", "social network", "mobile communication", "port", "algorithm", "cache", "encode", "gateway", "online celebrity", "NAT", "network application", "network connection", "WebRTC", "Internet telephony", "VR", "virtual reality", "virtual gift", "video conference", "video coding", "video communication", "decode", "ASR", "call", "super resolution", "configuration", "firewall", "thread", "process", "SDK", "texture", "cross-platform", "interface", "e-commerce", "compile", "window", "extension", "plug-in", "bit rate", "WiFi", "link", "WebView", "monitor", "Android", "iOS", "transcode", "hard decoding", "soft decoding", "APP", "search engine", "broadcast", "machine code", "runtime", "screen recording", "uninstall", "operation and maintenance", "DLL", "WWW", "container", "key", "agent", "compression ratio", "interpolation", "source code", "registry", "Linux", "users", "status", "internet celebrity", "social", "mislead", "startups", "circle", "content-production", "media", "work flow", "collaboration", "recommendation system", "interest exploration", "audio and video", "trans-platform", "framework", "mixture stack", "preview", "beauty", "handler", "engine", "main thread", "across platforms", "cross-terminal", "cross-platform", "online", "shooting", "screenshot", "front end", "client", "server", "evolution", "preloaded", "standard", "exclusive", "time delay", "latency", "lapse", "server", "small program", "third-party", "downlink", "mobile terminal", "mobile", "routing", "render", "transmission", "coverage", "weak network", "congestion", "feedback", "network fluctuations", "uplink", "live show", "customer service", "softphones", "application", "app", "extension", "systemic sourcing", "backstage", "subprocesses", "master process", "main process", "automatic

upgrade", "child window", "initialize", "message listening", "unvarnished transmission", "header file", "global", "instruction", "executable file", "integrity", "symbol file", "call", "low-level", "traffic", "three-way call", "API", "cursor", "constraint", "capture", "request", "message box", "point-to-point", "signaling", "cluster", "deployment", "end-to-end", "monitor", "open source"]

Smartphone : ["resolutions", "interfaces", "neural network", "processing unit", "codec", "60fps", "installation-free", "alliance", "end side", "intelligentization", "middleware", "kernel", "scheduling mechanism", "database", "communication mechanism", "fragments", "stack", "I/O", "io", "virtual machine", "tatic compilation", "lagging", "ecology", "underlying", "scheduling", "power consumption", "performance", "case", "hypnotic", "water face", "haptic", "speaker", "microphone", "ceramic", "sapphire", "Radio waves", "dual-core", "fall detection", "cellular", "altimeter", "battery life", "electrocardiogram", "electrodes", "music streaming", "wide color", "flyovers", "split view", "touch-sensitive layer", "touch-sensitive layer", "stereo field", "infrared camera", "flood illuminator", "proximity sensor", "ambient light sensor", "dot projector", "facial authentication", "fusion system", "graphics performance", "single processor", "portrait segmentation", "shortcut", "render", "dual-camera system", "wide camera", "optical image stabilization", "aperture", "lens", "exposure", "white balance", "sets the focus", "highlights", "panorama", "masks", "shutter lag", "buffer", "interframes", "bokeh", "full-frame", "depth", "battery life", "bands", "roaming", "cellular", "carrier", "HD", "biogas", "biogas fuel cells", "LCD", "engineering", "pixel masking", "anti-aliasing", "track pad", "powerhouse", "single camera system", "aperture", "tone", "background blur", "megapixel", "arsenic-free", "beryllium", "structural bands", "structural aluminum band", "widened", "versatile", "precision-machined", "anodized", "machine learning", "immersive", "finishes", "encrypted", "fall detection", "cellular", "band", "sport loop", "reflective yarn", "surgical-grade", "chlorinated water", "biometric", "terabyte", "gaming console", "Autofocus", "auto exposure", "aerospace grade", "trailers", "sneak peeks", "desktop class browsing", "SD", "thumb drive", "screenshot", "always on", "polysilicon", "complication", "titanium", "watch face", "print", "wide camera", "ultra-wide cam-

era", "semantic rendering", "multiscale", "stereoscopic depth", "studio", "resolution", "zoom wheel", "time lapse", "quick video", "PVD", "matte", "pro", "panel", "haptic touch", "nits", "matrix multiplication", "AR", "people occlusion", "voltage domains", "clock gating domains", "fused combination", "cinematographer", "calibrating for color", "Telephoto", "pro", "WiFi", "bluetooth", "buttery life", "cellular network", "wireless", "LTE", "mobile payment", "touchscreen", "finger-operated", "keyboard", "app store", "third-party software", "digital camera", "4k resolution", "1080p", "facial scanning", "Face ID", "iPhone", "Huawei", "Motorola", "wireless charging", "water resistant", "MicroSD", "USB-C", "OLED", "Samsung Galaxy", "waterproof", "fingerprint", "facial recognition", "chip", "battery charger", "HDR", "text messaging", "iOS", "Android", "web browser", "cloud storage", "smartphone", "Nokia", "BlackBerry", "Leica", "triple camera", "dual camera", "Ultra Wide Angle", "lens", "Kirin", "exposure", "Curved screen", "slow-motion", "bokeh", "pixel", "zoom", "distortion", "fps", "reverse charge", "fast charging", "mA", "plug", "front camera", "selfie", "LCD", "brightness", "time-lapse photography", "auto focus", "anti-shake", "delay", "film", "fingerprint unlock", "QR code", "app", "widget", "horizontal screens", "vertical screens", "SDK", "interface", "NPU", "GPU", "CPU", "open source", "lag", "memory", "UI", "vertical axis", "horizontal axis", "built-in", "volume", "watch face", "haptic", "linear motor", "flip", "speaker", "echo", "ceramic", "sapphire", "in package", "dual-core", "accelerometer", "gyroscope", "ECG", "altimeter", "wearable", "stereo", "infrared", "flood illuminator", "transistor", "storage", "ISP", "optical", "white balance", "shutter", "SIM", "scroll", "PVC", "low-carbon", "panorama", "diagonal", "gigabyte", "go dark", "aluminum", "stainless steel", "cropping", "scaling", "IC"]