Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Visual Analysis of News Stories Using Neural Language Models

Jena Satkunarajan

Course of Study:	Informatik
Examiner:	Prof. Dr. Thomas Ertl
Supervisor:	Johannes Knittel, M.Sc.
Commenced:	January 11, 2021
Completed:	July 08, 2021

Abstract

With the introduction of computers of varying sizes in the everyday life of the majority of the world population, we have seen a rapid increase in the amount of textual contents produced and distributed across the digitized globe. Among the insurmountable amounts of text found in the internet, news articles are of particular interest to many journalists, scientists and any other groups interested in the events captivating the public interest. As hundreds and thousands of online news providers report about the important and less important topics, it becomes an almost impossible challenge to gather and collect the valuable knowledge provided by all these sources. To gain an overview over the general happenings or to learn about specific topics thus becomes the task of identifying the novel information among an ocean of recurring, duplicate and rewritten stories. This thesis presents a combined approach to interactively visualise the novel content and the evolution of topics in news story corpora. A prototype framework is developed that utilises the GPT-2 transformer neural network based language model to assess the novelty of textual contents. Building on the resulting novelty scores, the textual contents of articles are visually highlighted to emphasise the novelty of the content. The novel article content is presented in multiple views, providing increasing levels of aggregation as the underlying article data grows in size. Employing a term weighting scheme incorporating the novelty scores, the ensuing document vectors are utilised to model the topics of the article corpus over time. The resulting, time-dependant topic clusters are presented in a multi-layered visualisation approach, providing multiple perspectives on the evolution of topics over time. The different visualisations and functionalities are combined into an interactive framework with multiple, coordinated views.

Contents

List of Figures

1 Introduction

Living in the digital age, the amount of new information produced from online resources seems endless. Textual data is published from multiple sources and in varying formats, constantly adding up in massive quantities. Blog posts, social media or online news providers generate new data on a daily basis, unpredictably accelerated by global phenomena like pandemics or natural disasters. One, even if not a data analyst, might see the need to view and examine the sheer amounts of textual contents, in order to gather relevant and previously unknown knowledge. The task at hand is not only aggravated by the quantities involved, as novel information will be buried among an overload of duplicate content: Social media users share and copy posts of particularly high interest, creating a web of slightly altered replicates spanning from the original content. In a similar fashion, distinct online news portal reporting the same event might present the same piece of information in different ways. Articles might appear at various timings, building upon previous reports by appending additional information marked as updates, even if the update offers little to no new insights. A full picture of the novel knowledge to be gained at specific times requires both

- an analytical distinction between previously known and unknown information,
- as well as an aggregation of the individual contributions of each article circulating at the time span of interest.

Gathering valuable insights from unstructured textual data has been and is still studied thoroughly in the field of text mining [Fra06]. Adapting and integrating statistical techniques utilised in novelty detection [PCCT14] has produced a number of works specializing in finding novel content in large text corpora. Though the tools to solve the aforementioned task seem readily available, it is only one step towards an insightful conclusion. Having extracted valuable, new information from a stream of texts, understanding its thematic relations and implications given the context of previous data requires further processing. This is especially relevant and challenging as successive new text entries form a growing sequence, among which the evolution of specific themes or events is to be followed. A structured approach is needed

- to reduce the complexity of the growing data at hand, by thematically grouping existing and new texts into clusters representing the relationships in the data
- to present a visual overview of the resulting themes as they evolve with the addition of new texts
- and to provide interactive tools to dynamically change the focus between multiple themes of interest.

Research on topic modelling [KB19] offers techniques to find latent topics among groups or clusters of texts and many works in the field of visual analytics [WT04] couple the results of such topic modelling algorithms with interactive visualisations to present and explore the topical evolution of temporary ordered text corpora.

While visualisation approaches exist to interactively analyse the evolution of novel topics, such as StoryTracker [KNMK13], there is little research on combining such approaches with state-of-the-art techniques like *deep neural networks* for novelty detection. Conversely, there exist new approaches to extract novel text content like the work by Knittel et al. [KKE18]. Yet, there is a lack of tools that surround the findings of such approaches with an interactive framework that allows to visually inspect the development of novel content over long periods times and for large amounts of text.

1.1 Contribution

This work proposes a combined approach to address the previously described task of extracting novel content from a given document corpus and subsequently model the temporal evolution of the prevalent topics found within the ordered set of documents. The approach utilises state-of-the-art, deep neural network based *language models* to first analyse and locate novel textual information in long sequences of news articles. The findings of the models are subsequently incorporated into advanced text processing and topic modelling techniques to effectively group articles of similar thematic content. The results of the process are embedded into a multi-level, interactive visualisation framework utilising *multiple coordinated views* to present the information gathered in various degrees of detail:

- On the lowest level, the approach allows to filter and examine singular news articles, visually and distinctively highlighting novel text passages.
- On the intermediate level, the temporary ordered articles are aggregated with respect to given search queries and filter settings. The presented results represent a summary of the novel, most important content, relevant to the query and subject to the constraints set by filters.
- The highest level corresponds to a timeline of the most important topics prevalent in the given corpus of news articles. The results of the topic modelling process are visualised with an emphasise on the evolution of topics, highlighting both the continuity of long lasting topics over time and the emergence and disappearance of new, potentially short lived topics. To inspect the content of articles associated with selected topics of interest, the visualisation is enriched with interactive tools to view relevant text passages and keywords in isolation, emphasising the novel content.

A web based prototype of the interactive visualisation framework is implemented and discussed, exploring the use cases of the combined novelty detection and topic modelling approach and analysing its advantages, disadvantages and future direction.

1.2 Structure

The thesis presents the approach according to the following structure:

Chapter 2 introduces the necessary theoretical and technical background to the techniques and approaches utilised throughout the thesis.

Chapter 3 explores related research on the topics of novelty detection, topic modelling and topic visualisation and compares previous works with the approach presented here.

Chapter 4 illustrates the formation of the prototype, explaining the reasoning behind the concepts of this work and how they tie in in the overall approach.

Chapter 5 presents the concrete implementation of the prototype and the selected technologies involved.

Chapter 6 showcases selected application scenarios for the approach and its resulting prototype and subsequently discusses the benefits and shortcomings.

Chapter 7 concludes the work with a look into alternative design decisions and future improvements.

2 Foundations

The following chapter discusses fundamental concepts and core technologies utilised throughout this work. First, we provide an introduction to Natural Language Processing, explaining basic and more advanced techniques to process natural textual data, to describe and model its statistical properties and to discover prevalent topics based on these statistical properties. The chapter continues with a compact overview of the most important concepts related to the field of Deep Learning and relevant to this work, laying the foundation to explain and discuss the usage of GPT-2 and its transformer based language model. A short summary of the relevant concepts borrowed from the fields of Visual Analytics and Text visualisation to build an interactive visualisation framework concludes the chapter.

2.1 Natural Language Processing

Understanding natural languages has been an ever present task and with its complexity, Alan Turing deemed it difficult enough to model his famous test [1] around it, have arisen a multitude of challenges. With computers popularised and standardised as everyday equipment for consumers and scientists alike, the field of computational linguistics has since targeted the issue of making artificial intelligence ”understand” natural language in various ways. At the heart of Natural Language Processing (NLP) stands the complex, composite task of finding a suitable representation for human language a computer can work with and defining an automatic processing system to extract the desired output. As Heimerl [Hei17] and Brill et al. [BM97] compactly summarise, the processing and analysis methods applied in NLP are mainly influenced by two paradigms:

- A rule-based approach to manually encode and utilise domain-specific linguistic knowledge, provided and often hand-coded by experts.
- An empirical, data-driven approach based on the automated, statistical analysis of text corpora, i.e. working directly with ”real” data, finding recurring patterns and subsequently generalising to *learned* rules.

The latter paradigm has been more and more prevalent as textual data became available in almost arbitrary amounts, rendering the appliance of linguistic knowledge by hand to such data sizes ineffective, due to cost and feasibility constraints. In contrast, with the increase in computing resources and advanced probabilistic models, statistical approaches offer a more extensible and robust framework when dealing with big data sets. While Heimerl points out that semi-supervised approaches combining linguistic rules with automated inference methods are gaining traction nowadays, the remainder of this work will focus mainly on data-driven statistical approaches.

[1] https://en.wikipedia.org/wiki/Turing_test

Nowadays, applications related to NLP or utilising NLP driven techniques are plentiful. Examples relevant for this work include

- **Document Classification**, where documents are assigned class labels from a predefined or learned set of labels. In the case of **document Clustering**, a set of documents has to be separated into groups of similar documents with respect to one or multiple properties of interest. **Topic classification** is an example for the a classification task, where documents are labelled by the topic or theme they belong to. Complementary to classification, in **topic clustering** documents are clustered according to the latent topics among the documents.
- **Information Retrieval** (IR), where given a query consisting of terms, phrases or documents, an IR system is tasked with finding related documents, that is, documents containing identical or similar words and phrases, documents with similar content or documents retrieved in a similar context.
- **Language Modelling**, a task to assess the likelihood of texts occurring as they are, enabling to distinguish between texts written by humans versus texts written by artificial intelligence for example.
- **Text summarisation**, where a given text is to be summarised by aggregating the most important words, sentences or passages within the text.
- **Part-of-Speech (POS)** tagging, where the individual components of sentences are labelled according to their corresponding grammatical function, example labels being *nouns*, *verbs* or *adjectives*.

2.1.0.1 Machine Learning

Mitchel [Mit97] defines machine learning (ML) as the process of a computer program learning to solve a given class of tasks T, improving its performance with respect to a performance measure P as it gathers more experience E with respect to T. Goodfellow et al. [GBC16a] further describe the roles of tasks, experiences and performance measures:

- The **task** specifies the overall goal to achieve and thus governs how the underlying program processes a given data example.
- The **experience** is provided by the input data set, that is, the machine learning program learns from the data examples, or samples, with respect to the given task.
- The **performance measure** guides the evaluation of the program's performance, thus providing feedback on how well the program has learned to solve the task.

The concept of experience can be further distinguished by the problem category: Supervised learning problems additionally provide labels with the data examples, indicating the correct output expected from the machine learning program when processing the specific data example and thus guiding the learning process. Unsupervised learning problems on the other hand require the program to estimate the correct output without any guidance, thus relying solely on the learned properties from the data examples.

Overall, a machine learning problem can be formally defined after Murphy [Mur21] as the task of learning a function $f : X \rightarrow Y$ that maps from the given set of input features $x \in X$ to a target representation $y \in Y$. The difference between supervised and unsupervised learning lies in the prediction of the target y. In the supervised setting, the machine learning algorithm is given a set of samples $\{(x_n, y_n)\}_{n=1}^{N}$ learning the mappings from inputs x_i to the correct targets y_i, thus estimating the conditional probability distribution $p(y|x)$, i.e. the likelihood of the outcome given the observation. In an unsupervised learning task, the program is only given samples $\{x_i \mid i = 1, ..., N\}$ and subsequently has to learn the probability distribution $p(x)$, i.e. the likelihood of the observation at hand for any possible outcome. Both supervised and unsupervised learning tasks aim to find the parameters θ to model a function $f(x; \theta)$ that that maximises the probabilities assigned to the correct targets y in the case of supervised learning. In the case of unsupervised learnin, the task is to model a probability distribution $P(x; \theta)$ that best describes the given observations the best. The **model** produced by the machine learning program is the result of the parameters learned while processing the given inputs and a "snapshot" of the programs current understanding of how to solve the given task.

When denoting parameters in a machine learning program, one has to distinguish between the **model parameters** to learn and the hyperparameters defining the setting of the task, as Goodfellow et al. explain. **Hyperparameters** are control variables that affect the learning process but do not determine the task performance, that is, they influence the quality of the learning process, but not the quality of the model's output. As such, hyperparameters are not learned, but fixed by the practitioner before training ensues. **Training** refers to the learning process, where the machine learning program receives the training data samples, subsequently processes the samples and fits the underlying model parameters to achieve the best possible performance for the given task. This might be done in an iterative process, where more and more data is fed into the model to improve the performance step by step. Goodfellow et al. note that this can be seen as an optimization problem: When training a machine learning program, the **training error**, i.e. the error rate of the model output after a particular training step, is monitored and improvement is defined as reducing the training error. Though as the authors further add, a machine learning task deviates from an optimization task as not only the training error is of relevance, but so is the **test error**. The test error is measured after the training process is completed, utilising a new data set the model has not "seen" yet, i.e. the samples have not been part of training. Of major importance for the data-generation process is the assumption of **independent and identically distributed samples** in each data set provided for the generation. After Goodfellow et al., the assumption of a shared probability distribution and independence of samples enables the comparison of training and test error:

- A decreasing training error, but an increasing test error implies the overfitting of the trained model to the training samples. **Overfitting** describes the phenomenon of the model learning the provided data samples "too well", increasing the complexity of the model with respect to the training samples, such that the model does not generalise to new data samples.
- If neither error improves, the model is **underfitting**, that is, the model is neither able to extract the important properties of the training samples with respect to the given task, nor does it achieve the same for new data samples.

The over aching goal is to **generalise**, that is, the model should be able to perform well not only for the training set, but for any new, independent data samples. Goodfellow et al. name the capacity of a model as a possible tuning point to prevent both over- and underfitting. The capacity determines the range of functions a model is capable of fitting, i.e. learn the optimal parameters to solve the

given task. The Section on Deep Learning and Deep Neural Networks discusses the example of how linear function compositions are not able learn the XOR function, essentially underfitting on any given training data set.

Different factors might influence generalisation, requiring to **fine tune** the learning algorithm and possibly modifying it. Several **regularization** techniques have been developed to control the model complexity and improve its generalisation capabilities, a range of which Hastie et al. [HTF09] discuss in their work.

Fine tuning the model and the the algorithm setting has to be strictly separated from testing the model's generalisation capabilities: Utilising the test data repeatedly to optimize the generalisation error produces a final model that is optimized with respect to the test set, overfitting the model on it and underestimating the true test error after Hastie et al. In practice, in addition to the training and test data sets, a validation set is utilised strictly to monitor the prediction error of a "freshly" trained model. Thus, a model is trained using the training set, subsequently *fine tuned* with the validation set and finally tested for its generalisation capabilities on the test set.

2.1.0.2 Text Normalization

Jurafsky and Martin [JM09] refer to a corpus as a machine-readable collection of digitized, structured textual data. As the data is collected from various sources, a relevant example being web based news article providers, the texts are likely to be unprocessed and noisy and as such often referred to as "raw data". This poses several challenges when performing NLP tasks: Special characters and non-standard words such as emoticons, currencies or abbreviations and neologisms add noise to the data, obscuring results of information retrieval or text summarisation tasks for example, as the parsing unit might not recognize said characters and words.

The example above illustrate the need to convert the unprocessed text into a standardized, canonical form, easing the subsequent processing. Text processing, which Indurkhya and Damerau [ID10] define as the conversion of unprocessed natural text into a well-structured sequence of meaningful units, incorporates subtasks commonly performed in a selected order depending on the application, forming a text processing pipeline. Multiple tasks to normalize text exist, among which **sentence segmentation**, **tokenization**, **stemming** and **lemmatisation** make up a significant portion of this work's text processing pipeline. The aforementioned terms shall thus be shortly presented and discussed in the following.

- **Sentence segmentation**, as the name suggests, denotes the process of splitting up written natural text into sentences, most commonly utilising punctuation characters like periods, question marks or commas as boundary landmarks. The decision upon what is considered a boundary again depends on the application and its settings, like the language for example. In the English language, periods, exclamation marks and question marks are very commonly used as sentence boundaries.
- **Tokenization** denotes the further segmentation of sentences or passages of the text into words. Multiple tokenization approaches exist, again varying depending on the setting. In space-delimited languages like English, white spaces are commonly utilised as word boundaries. A token after the initial segmentation might still consist of multiple word components or incorporate non-alphabetic characters like punctuation characters, for example

could've. The several, task specific steps are performed by **tokenizers**, on the basis of predefined rules or machine learned. Tokenizers commonly utilise filter mechanisms, using regular expressions for example, to detect and split punctuation from words, creating two or more separate tokens, to recognize special character sequences like currencies or to split words containing numbers.

Given the sentence *He went to the park to make something, which made her go home*, a possible list of output tokens from a tokenizer could be *he, went, to, the, park, to, make, something, which, made, her, go, home*. When determining statistical properties like the frequency of word types, the unique words appearing in a given word sequence, the above tokenization output raises certain issues. Assuming the number of distinct tokens equals the number of types, will lead to an inaccurate number of 13 word types, despite *went* and *go*, and *make* and *made* originating from the same words *go* and *make* respectively.

- **Word normalization** helps to generalise such cases by transforming or reducing words into a standard form. While the cleaning steps performed by tokenizers can be seen as a basis step towards generalisation, more advanced techniques focus on the morphology of words, the study of the composition of words from smaller "meaning-bearing" units [JM09]. In the example above, *went* is a morphological variant of *go* and to reduce the latter word to its former base form, **stemming** and **lemmatisation** are commonly employed. Jurafsky and Martin describe stemming as a crude, heuristic approach, as it reduces words to their stem, the central part of the word containing its meaning, for example *go* in *goes*. Lemmatisation is a more sophisticated approach, where words are mapped to their canonical form, their lemma, for example both *is* and *was* to *be*. Jurafsky and Martin denote the difference in complexity between stemming and lemmatisation: Where stemming applies a comparatively simple, commonly rule based reduction, lemmatisation requires a morphological analysis of the word and analysis of the surrounding context to disambiguate multiple potential lemmas.

2.1.0.3 Representation

Many NLP applications rely not only on normalized and cleaned textual data, but base their results upon a meaningful representation of the texts. The word *meaningful* can be taken very literal: Searching a documents for related phrases given a query, one assumes the information retrieval system to search for phrases that exactly match and for phrases that are similar in their meaning. Jurafsky and Martin define the term **word senses**, that is, the multiple meanings a single word can have: *Windows* can either mean openings for light inside buildings or the operating system.

Choosing a representation for the words of a given document set, thus, becomes a task of encoding the meaning of the words in way that enables to compare and assess the relationship between similar and dissimilar words. Jurafsky and Martin name the concept of **vector semantics**: Given a document from a corpus with a vocabulary of n distinct words, each word is mapped to a point in n dimensions. In the resulting vector space model, every document is represented as a vector in n dimensions, each dimension corresponding with a word in the vocabulary. The construction of the vectors typically relies on statistical properties, where the i-th entry in the word vector is weighted with respect to the i-th word in the vocabulary. Heimerl [Hei17] names the boolean weighting and the term frequency weighting as two examples for simple weighting schemes and points out their shortcomings. In the first approach, every entry in a vector is assigned a binary value denoting

whether the word appears in the corresponding document, treating every word the same, irrespective of its actual importance with respect to its document's content. In the latter approach the binary values are replaced with the absolute count of each word with respect to the document, skewing the perception of importance: A word appearing a hundred times is not necessarily ten times more important than a word appearing only then times. The **tf-idf** weighting scheme is a more sophisticated approach, incorporating the relevancy of a word with respect to both its document and the overall document corpus in its weighting. As Manning et al. explain, the approach constructs a composite score from two core weighting components:

- The term frequency count $tf_{t,d}$ of term t denotes its count specifically for the document d. Commonly, rather than using the raw frequency, the weight is additionally normalized to account for discrepancies in counts as described in the example above. Examples after Manning et al. include dividing the weight by $\sum_{t' \in d} tf_{t',d}$, the overall number of terms in the document d, dividing by the maximum term frequency $tf_{max}(d)$ of a term in the document or taking the logarithm of the term frequency $1 + log(tf_{t,d})$.
- The inverse document frequency idf_t of term t encodes the relevancy of a word with respect to the document corpus, that is, it accentuates term weights for terms confined to a few documents, but appearing often there, while scaling down weights for terms appearing often and across multiple documents. The inverse document frequency first requires to calculate the proportion $\frac{N}{df_t}$, with df_t being the document frequency of the term t and N being the overall number of documents. Taking the logarithm of the aforementioned proportion gives $idf_t = log(\frac{N}{df_t})$. Going back to the example before, the word *the* is likely to appear in all documents, producing a score of 0 or close to 0. On the other hand, if *death* is "exclusive" to a selected few documents, the rarity of the word across the corpus will yield a low document frequency and subsequently increase the inverse document frequency.

The tf-idf score of a term t in a document d is then calculated as

$$tf\_idf(t, d) = tf_{t,d} \cdot idf_t$$

marking words appearing seldom across the corpus, but often in particular documents as an accurate representation of the contents of those documents, while diminishing the influence of words shared by multiple documents in the corpus and thus likely to be less representative. The approach presented in this work utilises the tf-idf weighting as a "base line", further modifying it to encode the novelty of the text passages the terms occur in, as described later on in 4.

Having determined a representation of the documents of a given corpus enables the comparison of the documents. A common task involving such comparisons is to find similar documents given a query or another document. A few selected similarity measures among a multitude of existing approaches will be shortly presented and discussed in the following:

- The **euclidean distance** [Hua08] is a basic similarity measure commonly utilised in tandem with vector based document representations. Interpreting documents as a point in vector space, the similarity of the documents can be calculated as the inverse distance between the points. A drawback of using the euclidean distance is the natural increase in distance as the difference in length between documents grows, irrespective of the content similarity.

- The **cosine similarity** measure [Hua08] offers a practical solution to the aforementioned issue, as the measure calculates the cosine angle between two vectors. The angle is agnostic to the length of the vectors and is mainly influenced by their orientation, thus growing or shrinking as documents share more or less terms. The cosine similarity is computed by calculating the dot product of the given vectors v and w and dividing it by the product of their lengths

$$cosine\_similarity(v, w) = \frac{v \cdot w}{|v||w|}$$

where the dot product of v and w is calculated as $\sum_i^n v_i \cdot w_i$ and the length of a vector v is calculated as its euclidean length $\sqrt{\sum_i^n v_i^2}$. As Manning et al. point out, the denominator in the equation above applies a length-normalization, producing unit vectors and measuring the angle between them. Unit vectors are of the same length, 1, thus eliminating the aforementioned issues related to different document lengths.

Document vectors stemming from the vector space model discussed so far are very sparse: Each document naturally only consists of a fraction of the document set's vocabulary and thus the vectors consist mostly of 0 entries. Jurafsky and Martin highlight reasons to desire more compact representations, examples being the increased complexity when training a classifier with such vectors or the computational effort to operate on vectors with multiple thousands of entries. **Word embeddings** [PSM14] offer an alternative representation based on short and dense vectors learned by a trained classifier and are discussed in more detail in section 2.2.4. To stick to the vector space model for now, sparsity can be reduced by simplifying the vectors, that is, reducing the size of the vocabulary and thus the dimensionality. Of course, trimming the vocabulary means removing words and arbitrary removal might result in the loss of important, "content bearing" terms. A common approach is to remove **stop words**, high frequency terms like determiners (*the*), conjunctions (*and*) or auxiliary verbs (*be*).

2.1.1 Language Modelling

The field of Automatic Speech Recognition (ASR) is concerned with the conversion of audio signals produced by human speech into a highly accurate textual translation of the spoken words relating to the signal. Any representation of the resulting words is inherently stochastic, as noise in the signal data induces uncertainty. As Jurafsky and Martin denote, the issue of converting ambiguous data into text can be found in other applications as well, examples being automatic query completion in search engines or spelling correction. The aforementioned tasks require a probabilistic model of language, or shortly **language model**, to predict the likelihood of a sequence of words occurring as it is, for example assessing whether the partial sequence *The weather today is rather* is followed by the word *cold* or *gold*.

As natural language follows certain grammatical rules and structures, the likelihood of words change as their context vary. In English, a sentence ending on the word *the* is rather unlikely, barring an error. Consider the event of a word w_n appearing as a random variable X taking on the value w_n, where the probability $P(X = w_n)$ is simplified to $P(w_n)$. The probability of a sequence of words $w_1, ..., w_{n-1}$ followed by the word w_n can then be seen as a chain of random events, describing a

joint probability distribution. Jurafsky and Martin illustrate how the chain rule can be applied to calculate the joint distribution as a sequence of conditional probabilities:

$$P(w_1, ..., w_{n-1}, w_n) = \prod_{i}^{n} P(w_i|w_{1:i-1}) \ [JM09]$$

The authors further explain why it is highly inefficient at best and infeasible at worst to estimate the true distribution with above equation: One way to calculate the conditional probability $P(w_n|w_{n-1}, ..., w_{n-k})$ is to utilise the **relative frequency**, i.e. estimate the probability as $\frac{|w_{n-k},...,w_{n-1},w_n|}{|w_{n-k},...,w_{n-1}|}$ in a corpus containing "enough" samples of either sequence. This is a challenging task, as it first requires a corpus that contains a representative amount of samples of all word sequences of interest to accurately estimate the relative frequency. Assuming such a corpus is given, modelling the joint distribution for n words requires a product over n factors, each factor estimated via the relative frequency count, subsequently resulting in a potentially very small probability.

More practical and widely used are **n-gram language models**, based on Markov models [MTG09]. The key assumption, also called the Markov assumption or Markov property, allows to model the conditional probability distribution of random variables with respect to a limited past. Precisely, in the context of language modelling, the property assumes the word w_n in a Markov model of order $(k-1)$ to only depend on its $k-1$ preceding words of the sequence, approximating the conditional probability $P(w_n|w_1, ..., w_{n-1})$ as

$$P(w_n|w_1, ..., w_{n-1}) \approx P(w_n|w_{n-k+1}, ..., w_{n-1}) \ [JM09]$$

As Jurafsky and Martin point out, the n-gram probabilities can be estimated as **maximum likelihood estimates** (MLE) utilising the relative frequency as established before, resulting in normalized counts ranging between 0 and 1. In the case of a bigram model, a first order Markov model, the probability $P(w_n|w_{n-1})$ is calculated as

$$P(w_n|w_{n-1}) = \frac{|w_{n-1}w_n|}{\sum_x |w_{n-1}x|} = \frac{|w_{n-1}w_n|}{|w_{n-1}|} \ [JM09]$$

where the first equation determines the proportion of bigrams starting with the same word w_{n-1} and ending on w_n or another word x respectively. The denominator can be simplified to the unigram count of the word w_{n-1}, as every bigram in the denominator contains the unigram w_{n-1}.

While n-gram language models offer a straight forward solution to model word probabilities, certain disadvantages can hinder their performance. A major factor in the effectiveness of n-gram models is the size of the training data, i.e. the underlying corpus. Word sequences occurring rarely with low corresponding counts will result in low probabilities or zero probabilities, heavily affecting the overall sequence probability. In the worst case, if the model encounters a word it has never seen during training, the entire sequence probability will be set to 0. **Smoothing** is a common method to handle unseen words and sequences, where probability mass is reallocated to low frequency events by discounting high frequency events. Additive smoothing [CG99] achieves this by adding a constant λ to each n-gram count, pretending to have seen additional training samples. The denominator of the maximum likelihood estimates is normalized to accommodate the "new" data samples, resulting in new bigram probabilities

$$P(w_n|w_{n-1}) = \frac{|w_{n-1}w_n| \cdot \lambda}{|w_{n-1}| + |V| \cdot \lambda}$$

with $|V|$ being the training vocabulary size.

While smoothing alleviates the issues of insufficient or sparse training data, generalisation remains a difficult task for n-gram models. A change in the application domain is likely coupled with a change in the underlying vocabulary and the statistical properties of the application corpus. Subsection 2.2.4 introduces neural network based language models as an alternative to n-gram language models with better generalisation properties and discusses the advantages and disadvantages of such models.

Assessing whether one language model performs better than another requires a metric to evaluate the predictive capability of the models. **Perplexity** is such a metric commonly utilised to evaluate language models, where the quality of a model is assessed with respect to its predictive performance on new and unseen data.

After Jurafsky and Martin, the perplexity of a model θ can be calculated as the normalized, inverse probability assigned to a test set T of n words:

$$PP_\theta(T) = \sqrt[n]{\frac{1}{P_\theta(w_1, ..., w_n)}}$$

The intuition of utilising perplexity to evaluate the model quality can be seen in the assigned probabilities: A better model is likely to assign higher probabilities to unseen data samples, as it has generalised better and is thus more ”confident”. A higher average probability will thus decrease the perplexity, marking a higher quality language model.

Suppose the underlying model is a unigram language model assigning word probabilities $P_\theta(w_i)$. Manning et al. [MRS08] additionally illustrate how perplexity can be expressed in terms of cross entropy:

$$PP_\theta(T) = \sqrt[n]{\frac{1}{P_\theta(w_1, ..., w_n)}} \approx \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P_\theta(w_i)}} = 2^{H_\theta}$$

with $H_\theta = -\frac{1}{n} \sum_{i=1}^{n} log(P_\theta(w_i))$ being the cross entropy for the model with respect to the true probability distribution.

As the authors explain, the above exponentiated cross entropy can be viewed as the branching factor of a language model, i.e. defined by the authors as the ”number of possible words that can follow any given word” [MRS08]. Intuitively, the higher the confidence of a model in selecting the most probable next word, the lower will be the branching factor and thus the perplexity.

2.1.2 Topic Modelling

When analysing large corpora of text with hundreds or thousands of documents, it is often practical to not examine every single individual document, but rather analyse collections of documents and extract common and shared properties among the collection members. **Topic modelling** is an unsupervised statistical learning approach commonly utilised to cluster documents according to latent topics described by the contents of the document corpus. Assuming a given corpus consists of documents that can be categorized by their **theme**, i.e. words appearing within the same semantic context: Intuitively, not every word in a document will be related to a document’s topic or carry any

importance with respect to it, while documents of the same topic are likely to share those words that are highly related to the topic. Topic modelling aims at extracting the topic defining keywords that function as a summary and representation of the topic and thus the key content of the documents they appear in. Furthermore, topic modelling can be applied to reduce the complexity of the both corpus and the documents, grouping and summarising similar documents that share the same theme, while separating documents of distinct topics.

2.1.2.1 Latent Dirichlet Allocation

As suggested by Liu et al. [LTD+16], a topic can be statistically described as a probability distribution over a set of words, assigning higher probabilities to related words. Conversely, every document can be described as a distribution over topics, the assigned topic probabilities reflecting the relevancy of a document's word content with respect to each topic. Recalling *unsupervised learning* tasks introduced in subsection 2.1.0.1, topic modelling thus becomes the task of fitting a probability distribution that best describes the given composition of words representing each document. After Blei [Ble12], the aforementioned probability distribution can be derived as the conditional probability distribution for the latent topics, given the observed documents.

Latent Dirichlet Allocation [BNJ03] assumes the given documents to stem from a generative process, resulting in the conditional distribution describing the latent topics. LDA subsequently leverages the assumption to *infer* the document-topic and topic-word distributions that best approximate the conditional distribution. Given M documents, each a sequence of N_d words from a fixed Vocabulary V, with K possible topics, the generative process of a document is described as follows:

- A categorical Dirichlet distribution $P(\theta_d; \alpha)$ describes the topic mixture for the document and the available topics.
- The topic-word distribution for each of the K topics is again described by a Dirichlet distribution $P(\omega_i; \beta)$,
- For every of the N_d words of the document, sample a topic z_d from a multinomnial distribution $P(z_d|\theta)$ derived from the topic mixture.
- Then, sample a word w_{z_d} according to the multinomial distribution $P(w_{z_d}|\omega_i)$ derived from the topic-word mixture.

The document-topic Dirichlet distribution is parameterized by α, determining the mixture of topics per document: A high value indicating more evenly distributed probabilities, while a low value concentrates the probability mass around certain topics. Comparably, β parameterizes the topic-word distribution, its value characterizing the relevancy of words with respect to the topic. The equation of the conditional probability distribution describing the latent topics of a document corpus resulting from above generation can then be stated as:

$$p(\theta, \omega, z, w|\alpha, \beta) = \prod_{d}^{M} P(\theta_d; \alpha) \prod_{i}^{K} P(\omega_i; \beta) \prod_{j}^{N} P(z_{d,j}|\theta_d) P(w_{z_{d,j}}|\omega_i) \quad [BNJ03]$$

Various inference algorithms exist to approximate above equation, Variational Inference [HBWP13] and Gibbs Sampling [GS04] being two examples. Viewing this as a machine learning task, we now want to maximise the probabilities assigned to the documents serving as our observations.

Having found the document-topic distribution and the topic-word distribution that achieve the aforementioned goal, we can manually associate documents and words with topics according to each distribution.

2.1.2.2 Non-negative Matrix Factorization

The previously introduced LDA interprets topic modelling as solving a probabilistic model. Alternatively, there exist approaches to topic modelling borrowing techniques from the field of multivariate analysis. Non-negative Matrix factorization (NMF) [PT94] is a dimension reduction technique to factorize matrices as low-rank approximations. The application of NMF for topic modelling relies on the concept of **document-term matrices**. Given a shared vocabulary V of size n for a corpus of m documents, the documents can be presented in a matrix of size $n \times m$:

- Each document is a row vector akin to the representation discussed in subsection 2.1.0.3, thus the vector entries reflecting whether the document contains each of the n words aligned along the columns. Recalling the construction of the document vectors utilising functions of word and document frequencies, we can determine each vector to contain only non-negative entries.
- Conversely, every column vector represents the corresponding word's membership to each of the m documents.

The resulting matrix $D \in \mathbb{R}^{m \times n}$ is non-negative by nature, such that NMF can be utilised to decompose the matrix D into two new, lower dimensional matrices: A matrix $W \in \mathbb{R}^{m \times k}$ and a matrix $T \in \mathbb{R}^{k \times n}$, where $k \ll min(m, n)$ is provided by the practitioner. To relate the decomposition to topic modelling, we can view the desired dimension k as the number of topics assumed to be prevalent in the document corpus. The resulting matrices can then be interpreted as follows:

- W represents a document-topic matrix, each row corresponding to a document. Each of the k columns then provides a numeric assessment of the topic's relevancy with respect to each document.
- The matrix T can be seen as a topic-word matrix determining the content of each topic: Each of the k rows represents a topic, while the n columns correspond with the words of the vocabulary V. Each entry then gives insight about the relevancy of a word with respect to each topic.

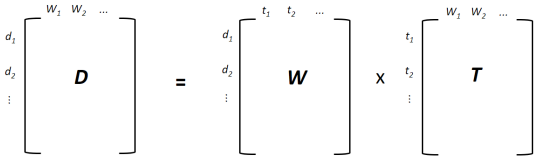


Figure 2.1: Decomposition of the document-term matrix D into the document-topic matrix W and the topic-term matrix T.

Figure 2.1 illustrates the decomposition of the document-term matrix D. In this work, we select NMF over the previously introduced LDA. This is motivated by the nature of the algorithm, allowing to define and, very importantly, modify the document-term matrix provided for decomposition. The resulting document-topic and topic-term matrices are simple to interpret and utilise, allowing for further processing, like computing the similarity of topics. Lastly, NMF returns deterministic results for multiple runs on the same input data, an important property with respect to the prototype developed in this approach: The user is allowed to create arbitrary groupings of articles with respect to their time ordering, thus being capable of testing the topic modelling results for different article groupings. Deterministic outputs are key for comparing the different topic modelling results.

2.2 Deep Learning and Deep Neural Networks

The field of deep learning has evolved into a major driver of many state-of-the-art, performance leading problem solving techniques, finding application in a broad spectrum of fields such as medical imaging [BKS+19], image recognition [FHY19] or automatic speech recognition [PCZ+19].

Central to all deep learning techniques are deep neural networks, layer wise architectures of numerous smaller processing units, configured to *learn* the parameters to approximate a function that solves a given task. The term *deep* refers to the stacking of multiple hidden layers to compose deep networks, transforming a given input into a different representation as the data passes through each layer and deriving the input's important features. The number of layers of such a network denotes the depth of the network, where every layer consists of multiple processing units commonly named *neurons* and the number of neurons determines the width of a layer. Starting from the input layer, every neuron receives an input from the neurons of the previous layer, computes a mathematical function to transform said input and propagates the output to the neurons of the next layer. The final output layer receives the transformed inputs and emits the output $\hat{y} = \hat{f}(x, \theta)$: The output produced by the approximation $\hat{f}$ of the $true$, data generating function f, given the input x and the learned parameters θ. While Deep Learning can be applied both in a supervised and an unsupervised task environment, this work will focus primarily on supervised tasks such as classification. In the aforementioned case, the neural model is provided a labeled data set, where the labels y function as the ground truth. The model then learns the parameters to approximate a function that produces outputs $\hat{y}$ as close as possible to the ground truth for the given input. The learning process is guided by a *loss function*: The difference between $\hat{y}$ and y is utilised to adapt the parameters learned so far, expecting to subsequently produce an output with smaller loss.

Beginning with fully connected Feed forward networks, many neural network variations like convolutional neural networks, recurrent neural networks or encoder-decoder-networks haven been since derived, often tailored towards specific applications. The subsequent sections will first provide a detailed look into Feed forward networks and the core principles of non-linear activation functions, the forward and backward propagation and loss functions, before moving on to short discussions of more advanced network architectures.

2.2.1 Feed Forward Neural Networks

A neuron in a deep neural network can be described as a function computing the weighted sum of its input. Given an input vector $x = [x_1, ..., x_n]^T$, a weight vector $w = [w_1, ..., w_n]^T$ and a bias b, the neuron output is calculated as

$$z = \sum_{i}^{n} w_i x_i + b = w \cdot x + b [JM09]$$

where the second equation simplifies the notation by expressing the weighted sum in terms of vectors and the dot product. Above equation produces a linear transformation of the input vector and the task of the neural model would be to learn the parameters w and b to approximate the desired function. The XOR problem is a commonly discussed issue ([JM09], [MRS08], [Mur21]) showcasing the limitations of linear functions for function approximation: The XOR function for two binary variables x_1 and x_2, $x_1, x_2 \in \{0, 1\}$ can be expressed as

$$f(x_1, x_2) = \begin{cases} 1, & \text{if } x_1 \neq x_2 \\ 0, & \text{else} \end{cases}$$

A linear function, i.e. a function linear in its parameters, would not be able to compute the XOR function as it is not linearly separable. As Jurafsky and Martin denote, a linearly separable function only requires a linear decision boundary, as provided by a linear function. A neural network of one or multiple hidden layers with linear activation functions can be collapsed into a single hidden layer with linear activation, as Goodfellow et al. [GBC16a] discuss. Such a layer can be thought of as separating the space into two hyperplanes, mapping inputs falling into either region to 0 and 1 respectively. Intuitively, such a separation does not solve the XOR problem.

This motivates the usage of non-linear activation functions, that is, the intermediate output z is further transformed using a non-linear function a to produce $y = a(z)$. As noted by Jurafsky and Martin, the activation function can be applied to both scalars and vectors, where in the latter case the function is applied element wise. The advantage of a non-linear activation function can be illustrated via the universal approximation theorem [2]: We can approximate any continuous function with a non-constant, bounded and monotonically increasing continuous activation function.

The sigmoid function, the hyperbolic tangent and the rectified linear unit (ReLU) are common activation functions and shown in figure 2.2. A good choice of the activation function depends on the requirements with respect to its properties, for example might one select the sigmoid function if a differentiable (at all points) activation function is required, while ReLU is widely popular due to its convergence properties [HSS15].

A Feed Forward network is constructed by chaining multiple layers of neurons with non-linear activation functions, where the input passes through each layer in one direction towards the output layer. The term *fully connected* is used to describe such Feed Forward networks, as every a neuron in layer l receives the outputs from all neurons in the previous layer $l - 1$. Each layer is associated with a weight matrix W^l shared by its neurons, such that the output of a layer l can be described as

[2] https://en.wikipedia.org/wiki/Universal_approximation_theorem

$$(1)\ \sigma(x) = \frac{e^x}{1+e^x}$$

$$(2)\ tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$(3)\ ReLU(x) = max(0, x)$$

Figure 2.2: The sigmoid function (1), the hyperbolic tangent (2) and the rectified linear unit (ReLU) activation function (3) are three examples for activation functions in deep neural networks. ReLU is among the most preferred activation functions due to its simplicity and fast convergence [HSS15].

a vector $v^l = a(z^l)$, with $v_i^l = a_i^l(z_i^l)$ being the output of neuron i given an activation function a and $z_i^l = W^l a^{l-1} + b^l$ being the weighted outputs of the previous layer. The final output of the network can be written as $\hat{y} = f(x, \theta) = o(W_o \cdot v_o)$ where o is the selected output function and y the target output for the given input. θ denotes the set of parameters to learn, being the collection of weights and biases $\theta = [W^1, b^1, W^2, b^2, ...]$ of each layer. The output function o receives a special notation here to denote the variety of output functions available, as the selected function depends on the task and the required target. In the case of multinomial classification for example, the network might be required to output a probability distribution over the set of classes. Commonly utilised for such cases is the **softmax function**, which Jurafsky and Martin define as

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad 1 \leq i \leq N \quad [JM09]$$

where z denotes a N-dimensional vector and as such the softmax function outputs again a N-dimensional vector. The softmax function normalizes the given vector and outputs a distribution ranging between 0 and 1, the probabilities summing up to 1.

The so far discussed *forward propagation* or *forward pass* emits an approximation of the outputs the "true" function solving the task at hand would provide. A loss function indicates how well the function was approximated, where the resulting loss $L(\hat{y}, y)$, i.e. the difference between expected output and actual output, builds the first step to adapt the learned weights to improve the approximation. For (multinomial) classification tasks, the cross entropy function is commonly utilised to determine the loss. Recalling the calculation of the cross entropy for an estimated distribution q and the true distribution p as $H(p, q) = -\sum_x p(x) log(q(x)$, we identify the output of the neural network $\hat{y}$ taking on the role of the estimated distribution, while y describes the true distribution. As such, the cross entropy loss of a neural network for a multinomial classification task with n classes can be written as

$$L(\hat{y}, y) = -\sum_{i}^{n} y_i log(\hat{y}_i) \quad [JM09]$$

where y_i and $\hat{y}_i$ denote the probability assigned for the i-th class respectively.

Jurafsky and Martin illustrate in their work [JM09] that the above loss function can be simplified if the true labels are deterministic, i.e. the correct class for the given input is assigned a probability of 1 by the true distribution, while the remaining classes have probability 0. In this case the cross

entropy loss can be expressed as the negative log likelihood of the correct class label estimated by the network:

$$L(\hat{y}, y) = -log(\hat{y}_i) \text{ with } y_i = 1 \quad [JM09]$$

The objective is then to find a parameter set

$$\theta^* = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{M} L(\hat{f}(x^i, \theta), y) \quad 1 \leq i \leq M \quad [JM09]$$

that minimises the average loss over all data samples $x = x_1, ..., x_M$, i.e. that maximises the estimated likelihood of the correct class given the input, as Jurafsky and Martin formulate.

Finding the minimum of the loss function with respect to the learned parameters is not an easy task, as Jurafsky and Martin and Goodfellow et al. argue: With the integration of non-linear activation functions, the loss function becomes nonconvex, that is, global convergence is not guaranteed and depending on the initialization of the model parameters, might the optimization algorithm only ever find local minima. The optimization algorithm at hand is most commonly *gradient descent*. Intuitively, we have a function $\hat{f}$ of multiple parameters to minimise, where the derivative of the function with respect to the parameters indicates the direction in parameter space with the greatest change of $\hat{f}$. Utilising the partial derivative $\frac{\delta}{\delta w}\hat{f}$ to measure the change of $\hat{f}$ with respect to individual parameters w, we can construct the gradient $\nabla_\theta \hat{f}$, a vector of all partial derivatives of $\hat{f}$ for every parameter w of the parameter set θ [GBC16a]. If $\hat{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n$ maps from vector valued inputs to vector valued outputs, the derivatives can be expressed as the Jacobian matrix $J \in \mathbb{R}^{m \times n}$, $J_{i,j} = \frac{\delta}{\delta w} f(\hat{w})_i$ [GBC16a]. To minimise $\hat{f}$, in our case the loss function, we can thus employ gradient descent by moving into the direction the negative gradient points to, i.e. update the previously learned parameters θ_t as:

$$\theta_{t+1} = \theta - \eta \nabla_\theta L(\hat{f}(x, \theta_t), y) \quad [JM09]$$

where η denotes the learning rate, a hyperparameter to prevent the algorithm from converging too slow due to small directional increments, or overshooting the target with large increments, after Jurafsky and Martin. Gradient descent updates the parameters of each layer utilising the gradient of the loss with respect to the parameters. Computing the gradient of the loss function requires to "backtrack" the network, i.e.compute the product of the derivatives with respect to each layer and its neurons. The **backward pass** or **backpropagation** provides an efficient algorithm to compute the gradients and is discussed in detail in the works of Goodfellow et al. [GBC16a] and Murphy [Mur21].

Recalling the activation functions shown in figure 2.2, we can identify a potential issue to encounter during gradient based learning. The computation of the derivatives for the gradient requires to differentiate the activation functions of the neurons. Depending on the choice of activation function, the resulting derivative might be small: The sigmoid function for example saturates as the input approaches either $-\infty$ or $+\infty$, leading to very small derivatives close to 0. As the derivatives are multiplied according to the chain rule, the over all gradient will *vanish*. Conversely, if the network produces many errors for the current set of inputs, the resulting loss will be large, leading to a *exploding* gradient. Vanishing gradients are commonly prevented by selecting a non-saturating activation function, motivating the usage of ReLU: The derivative is either 1 if the input $x > 0$ and 0 if $x < 0$, the only issue being that the derivative is not defined for $x = 0$. Other preventive techniques employ regularization [GBC16b] and analyse the influence of weight initialization [YC00].

2.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) derive from Feed Forward networks by introducing the notion of *memory* or *context*. The feed forward networks discussed so far do not contain cycles, i.e. information flows, starting at the input, in one direction through the network until the output is produced. RNNs add a circular flow component in the form of *hidden memory units*: The input undergoes the transformation steps discussed for feed forward networks to extract relevant features, which are then stored by memory units in the hidden layer. Note that the memory unit can be simply a hidden unit with storage properties, but for exemplary purposes, the following will make a distinction between hidden units employing the non-linear activation and memory units storing intermediate results. The memory units then provide the intermediate results to produce the output: The output is formed with the information extracted from the current input and the information from the previously processed input stored in the memory units. These cycle inducing, "recurrent links" between hidden units, as Goodfellow et al name them, constitute the recurrent hidden layers of RNNs. RNNs thus specialize in processing sequential data, i.e. the input x is segmented into a sequence of values $x_t,\ t \in \{1, ..., n\}$, the indexing representing an implicit or explicit temporal ordering. The contextual information from previous steps provided by the hidden units can span the entire input sequence, such is the case when the hidden units are updated after every step. The forward pass for a given input vector $x = [x_1, ..., x_n]^T$, the memory h_t, activation functions a, o and weight matrices W_h, W_m, W_o (biases omitted) can then be expressed as:

$$h_t = a(W_h x_t + W_m h_{t-1})$$
$$y_t = o(W_o h_t)$$

for $t \in 1, ..., n$, $h_0 = 0$ and h_{t-1} being the previously stored memory. The simplified network illustrates the important concept of weight sharing, i.e. the weights are shared across multiple time steps. This allows the model to abstract from specific input positions when identifying important features and propagate those features as the input sequence is being processed. Subsection 2.2.4 illustrates how the the advantages of contextual information stored in memory units can be leveraged for neural language modelling.

2.2.2.1 Long Short Term Memory

Hochreiter and Schmidhuber [HS97] identified the issue of vanishing and exploding gradients when employing Backpropagation Through Time to train RNNs. As a progression from RNNs, the authors proposed Long Short Term Memory (LSTM) recurrent networks as an answer to the challenges when working with long input sequences: Far reaching dependencies are difficult to learn from as the gradient vanishes, effectively stopping the propagation of contextual information extracted at the beginning of the sequence as more and more time steps are processed. As Jurafsky and Martin denote, the addition of units to selectively "forget" or "remember", that is, to remove or keep the stored contextual information, alleviate the issues of vanishing and exploding gradients. For that, LSTMs introduce *gated units* into the previously discussed recurrent architecture. These LSTM cells maintain a state to actively control the contextual information that is propagated or removed by updating the state appropriately. The cell gates act as binary gates employing the

sigmoid activation function, with its output values ranging between 0 and 1 and thus indicating how much information to keep. The gates take in the input at the current step x_t and the memory h_{t-1} and subsequently update the internally maintained context state as the input and previous state pass through a *forget gate*, an *input gate*, an *update gate* and an *output gate*. LSTMs have shown to perform well on sequential data, for example in speech recognition [LW15] and are often subject to approaches combining different neural network architectures, for example in time series classification [KMDC18].

2.2.3 Transformer Neural Networks

Research on LSTM over the years has motivated a multitude of approaches to maximise its effectiveness when processing sequences, yet remains the core issue of information loss as inputs pass through the recurrent architecture: LSTMs process the data sequentially, input by input, learning ”along” long gradient paths with continuously updated hidden memory states. While LSTMs are able to alleviate the issue of vanishing gradients compared to RNNs, the issue is still prevalent, especially for very long input sequences. Additionally, as Vaswani et al. [VSP+17] point out, does the sequential nature of LSTMs prevent the efficient utilisation of parallel computation. **Transformer neural networks** thus move away from the sequential approach of maintaining an up-to-date hidden memory of previous inputs and instead integrate the concept of **self-attention**.

Transformer networks originate from the encoder-decoder sequence to sequence architecture [SVL14], a popular network architecture utilised when training models to map from a specified input domain to a specified output domain. This is commonly applied in language translation for example, where a sequence of words from language A is first processed by an *encoder*, a sequential network like a RNN or LSTM. Relevant extracted features are subsequently processed by a *decoder*, again a sequential network, to convert the sequence into the target language B. While sequence to sequence models specialize in converting variable length sequences, transformer networks map from and to sequences of the same length.

Transformer networks are heavily utilised in NLP applications like text generation or text translation. In such cases, the input consists of a sequence of words, turned into vector representations, i.e. word embeddings, by the encoder and the decoder respectively. An important distinction from RNNs lies in the processing of the input sequences, as transformer networks processes the entire sequence at once. The sequential nature of RNNs imposed an order onto the processed input, while the learned input embeddings in transformer networks do not carry positional information on their own. Transformer networks thus add positional encodings, i.e. vectors calculated based on the input's relative position in the sequence, to the input embeddings. The input is subsequently processed by self-attention layers, marking the heart of the transformer network architecture. A self-attention layer makes use of having the entire input at its disposal, to determine the relevancy of inputs with respect to the rest of the sequence. The need to store a memory of important features encountered so far is eliminated by comparing the $i-th$ input against all of its predecessors. The comparison is guided by the notion of *attention*, that is, the network computationally determines the most relevant inputs to attend to when focusing on a specific input, including itself. To encode the relevancy, Vasvani et al. introduce **query**, **key** and **value** vectors:

- A query vector q_i represents the current input of interest and center of focus, constructed by linearly transforming the input i with a learned weight matrix W_q

- A key vector k_i represents a preceding input, the input of interest is compared against, constructed by linearly transforming the input i with a learned weight matrix W_k
- A value vector v_i represents the input's contribution to the output for the current input of interest, constructed by linearly transforming the input i with a learned weight matrix W_v

Given an input sequence of length $n \in \mathbb{N}, n > 1$, to compute the self-attention of a specific input i, the self-attention layer takes the query vector q_i and calculates the pair-wise dot products of q_i and the key vectors v_j, of the remaining inputs j in the sequence. These dot products, representing i's relevance score with respect to all other inputs of the sequence, are subsequently normalized using the softmax function to produce the relevance weights $w_{i,j}, \forall j \in n$. The output for i is then computed as the weighted sum of the relevance scores and i's value vector v_i and further processed by the remaining components of the transformer architecture. As the outputs can be calculated independent of each other, the procedure described so far can be easily employed in parallel by forming input embedding matrices and query, key and value weight matrices. Note that both the encoder and the decoder consist of such self-attention layers, producing self-attention vectors for their corresponding inputs. The *encoder stack*, consisting of multiple encoders, processes the input by computing the embeddings and the positional encodings first, subsequently transforming the input in the self-attention layers followed by a feed forward network and normalization steps. The *decoder stack* utilises the resulting output embeddings as its input, adding positional encodings to the input similar to the encoder. The input is subsequently processed step by step, going through self-attention layers similar to the encoder stack. Though, a key difference distinguishes the attention mechanism of encoders and decoders, influencing the architecture of succeeding transformer networks like BERT and GPT-2 as we will see later on: The encoder is allowed to peek towards "future" inputs, i.e. inputs to the right of the currently processed position, utilising the context of these inputs. The decoder *masks* all inputs starting at the current position up until the end of the sequence, preventing the model to simply look up the next output it is tasked to generate. To allow the network to "pay attention" to multiple aspects of inputs and relationships between them, Vasvani et al. furthermore introduce the concept of **multihead attention**. The different stacks of transformer networks thus consist of not only one, but multiple self-attention layers working in parallel, each with a distinct set of parameters to learn.

Transformer neural networks are thus capable of modelling long distance interaction and dependencies in long input sequences. Inputs sequences are processed as a whole, relying on the attention mechanism to capture relevant contextual information instead of using a hidden memory mechanism which needs to be updated after every processing step. The highly parallelizable process of both encoders and the decoders allows to effectively speed up the training of the network using additional computational resources.

2.2.4 Language Modelling with Neural Networks

Neural networks based language models perform very well in various NLP applications revolving around the prediction and generation of text, outperforming the previously introduced statistical methods like n-grams. Neural language models learn to approximate the probability distribution $P(w_n|w_,...,w_{n-1})$ for a given word sequence $(w_1,...,w_{n-1})$, thus predicting the most probable word following the given sequence. Recalling the training process of neural networks, a neural language model is trained with various samples of text sequences, such that the model learns **word**

embeddings, accurate, abstract representations of each word it encounters. Word embeddings are short and dense, real-valued vectors and each word is associated with such a vector. Contrary to the count based vectors introduced in section 2.1.0.3, the values for word embeddings are learned by trained models, word2vec [MCCD13] or the BERT transformer language model [DCLT19] being two examples. The word embeddings are learned based on contextual statistical information. In the case of word2vec, the co-occurrence statistics of neighbouring words are utilised to train a binary classifier predicting whether a word is likely to appear near a given context of words. The learned weights of the classifier are then used as the word embeddings. In comparison, BERT takes in entire sentences as input and creates dynamic, context dependant word embeddings: Different *word senses* are captured by employing *masked language modelling* (MLM), where specific words in a given sequence are masked. The model is then tasked to predict the masked input by processing and understanding the bidirectional context. BERT combines the task of MLM with *next sentence prediction* (NSP), where the model is given two sentences and is tasked to predict whether one sentence contextually follows the other. The transformer language model thus learns to represent words with respect to the context they appear in, even across sentences. The learned embeddings provided by word2vec are static, **pretrained** embeddings, that can be utilised without further processing. Comparably, but on a bigger scale, BERT provides **pretrained language models**, i.e. models based on the embeddings learned from sizeable amounts of unlabelled data. Such pretraining is commonly performed in an unsupervised or semi-supervised fashion. These pretrained models perform well for various tasks [PNI+18], [EBA19], even if the specific tasks are performed on data unrelated to the training data set used for pretraining, thus offering a solution for tasks where labelled data is sparse. Pretrained models can be further fine tuned by training such models with task specific, labelled data and thus adapting the learned parameters for the new task. Both pretraining and fine tuning combine the concept of **transfer learning**, which many of the new transformer based models like BERT specialize in. Examples for such fine tuned models are the many variations of BERT [BLC19], [MMO+20].

To summarise, neural language models offer important advantages in areas where statistical methods fall short:

- As shown in the sections before, recurrent network and transformer networks are capable of processing sequences of increasing size and incorporating contextual information extracted from previous inputs. This is especially true for transformer networks, being able to process very long sequences without losing valuable information due to vanishing gradients. Language models based on such neural networks can utilise the additional contextual information to dissipate long distance relationships and dependencies between words, for example in translation tasks.
- The usage of word embeddings reduces the number of parameters to learn, as the resulting vectors are dense and shorter than count based vectors. Furthermore, word embeddings allow neural language models to generalise well: The learned, real-valued vector representation of words allows to easily compare seen and unseen words. If the word *picture* has not been part of the training corpus, but is represented similarly to the already seen word *illustration*, the model will be able to generalise and predict the former word in a context the latter word appears in.

- The dynamic, context dependant embeddings learned by pretrained language models like BERT can be utilised to disambiguate different contextual meanings of words. This allows the underlying model to further generalise and provides it with more information, potentially increasing its confidence and thus the quality of its predictions.
- Statistical models like n-gram language models are usually domain and data bound, i.e. it is difficult to utilise n-gram models trained on a specific data set in application domains with structurally different data. Neural network based models allow to solve such tasks in two steps, first pretraining a model that provides a baseline performance for general tasks, while task specific performance can be greatly improved by fine tuning.

This is not to say that statistical models are unanimously inferior, as there are certain use cases where the benefits of statistical models might outweigh the aforementioned advantages of neural language models. Statistical models like the previously discussed n-gram models are easy and especially fast to learn, requiring less expertise to use. They further require less computational power compared to the extensive weight learning and optimization process of neural language models.

2.2.4.1 Generative Pretrained Transformer 2

The **Generative Pretrained Transformer** (GPT) and its successors GPT-2 and GPT-3 are transformer network based language models comparable to the previously introduced BERT. The different variations of GPT differ mostly in the size of the respective models and the size of the data set they have been trained on, with the largest GPT-2 model containing 1.5 billion parameters and GPT-3 containing up to 175 billion parameters. As GPT-2 offers models in multiple sizes to use for research, while GPT-3 is both licensed commercially and simply too large for research purposes like this thesis, the reminder of this work will mainly focus on GPT-2. GPT-2 is specialised in **autoregressive generation**, which Jurafsky and Martin [MMO+20] describe as the generation of text word by word, where each word is conditioned on the preceding sequence. More generally, GPT-2 performs well in *causal language modelling*, that is, predicting the the next word given a sequence of words. GPT-2 exhibits certain architectural differences compared to BERT, reflecting the different intended usages of either transformer network:

- BERT, as discussed before, specializes in extracting bidirectional context on sentence level by employing *masked language modelling*. BERT is not *autoregressive*, as it has access to the entire input sequence when attending to a specific input within the sequence. While the model thus learns an accurate representation of the sequence by recovering the *masked* inputs, it ”loses” the ability to generate text: The model does not learn to predict an upcoming token conditioned on the preceding sequence. BERT consists of a multi-layer encoder stack and no decoder stack. Its functionality as a *bidirectional encoder* requires the encoder stack only, as the self-attention layers of encoders allow to attend to the entire input sequence during all processing steps. As shown by the authors [DCLT19], BERT thus performs well in NLP tasks like *question answering*, where the model is predominantly required to understand the contextual dependencies within a given question sentence to derive a possible answer.
- GPT-2, as alluded to before, is *autoregressive* in its nature, learning a generative model to produce texts token by token. Complementary to BERT’s approach, GPT-2 employs a multi-layered decoder stack architecture, omitting the encoder stack as the transformer network is not concerned with bidirectional encoding. Each decoder utilises masked self-attention

similar to the mechanism discussed for the general encoder-decoder transformer network. GPT-2 was trained on web data crawled from the social media platform *reddit*, containing up to 40GB of text across 8 million cleaned documents and a vocabulary size of 50 000. GPT-2 language models are *pretrained* with a language model objective, maximising the conditional log-likelihood for any given word of a corpus and its preceding context. Depending on the size, the models allow a context size of up to 1024 tokens. The so learned word embeddings are utilised to encode the given input word sequences in predictive and generative tasks, modelling the probability of a given sequence with respect to the corresponding embeddings. The *pretrained* models can be subsequently *fine tuned* on task specific, labelled data sets, further optimizing the learned embeddings and model parameters for the task at hand.

Having decided on of the GPT-2 based, different sized models for language modelling, we can utilise the model both to model the probability of a sequence and to generate text. For the former task, we can input a sequence whose likelihood we are interested in, for example to determine whether the sequence contains any unlikely words of word structures. GPT-2 will calculate the likelihood of the sequence as the average log likelihood of words of the sequence, conditioning the $n-th$ word on the $n-1$ preceding words when determining its likelihood. In the case of text generation, the model takes in a sequence $w_1, ..., w_{n-1}$ acting as the context and starting point to continue on. The model subsequently processes the sequence and conditions any follow-up token on the provided context. To predict the upcoming token w_n, the model *samples* for the token w_n maximising the conditional probability $P(w_n|w_1, ..., w_{n-1})$, i.e. the model finds the most likely word given the context. GPT-2 allows to repeat this process with a predefined text length, such that the model starts with an empty sequence or a given context and iteratively generates follow-up token until the desired sequence length has been obtained. In the generation process, there are several different methods to determine a specific token at a specific position. **Greedy Search** [Hug20] selects the token that maximises the sequence likelihood given the preceding context. This is prone to producing repeating sequences and little variation in the generated text, as high probability words dominate less probable, yet possible words given the context. **Beam Search** [Hug20] is often considered as an alternative, expanding multiple promising sequence paths rather than deterministically following the path of the locally highest probability. **Beam Search** does so by keeping track of multiple *beams*, i.e. multiple tokens branching from the current position and follows these branches while monitoring the most likely path among the tracked paths. This produces less repetition and more natural sounding text sequences, though it does not eliminate the repeating sequences. A more sophisticated approach and the method utilised for this work is **top-p sampling**. *Sampling* denotes the process of probabillistically picking the next token of the sequence. This ensures that low probability words, that are plausible continuations of given a specific context, are still picked from time to time. The randomness induced by the probabilistic approach can result in unnatural word sequences, which we can counteract by restricting the set of words to consider. In *top-p sampling* [Hug20], we define a value $p \in [0, 1]$, such that the next word is sampled from only the subset of words whose cumulative probability exceeds p. This can be combined with *top-k sampling* [Hug20], specifying the maximum number k of words to consider in the subset, such that highly improbable words are eliminated from the sampling process. The *top-p sampling* method thus reflects the natural text generation considerably well: Human produced texts are usually a colorful mix of both highly likely, but also uncommon and unlikely word sequences, depending on the context and the intention of the writer or speaker.

2.3 Visualization Techniques

This final section concludes the foundations chapter with an assemble of different visualisation and interaction techniques utilised throughout the approach proposed in this work.

2.3.1 Multiple Coordinated Views

When presenting multi-faceted data, it is often desirable to have multiple perspectives to view the data from. Certain aspects and insights might reveal themselves as we interactively compare two or more different visual presentations of the same underlying data. *Multiple Coordinated Views* (MVC) [Rob07] is a visualisation technique targeting the visual, exploratory analysis of potentially high dimensional data. At the heart of the technique stands the coordination of multiple views on the same data, each view providing the user with interactive tools to actively search and reveal potentially hidden insights. The MVC approach is central to many interactive visualisation approaches, as it allows for the effective analysis of high-dimensional, complex data. It further allows to integrate multiple approaches, yielding different visualisation results, without interference and in a space-efficient manner with respect to the presentation canvas. Thus, the MVC approach is key to the approach of this thesis and the development of the interactive visualisation framework prototype.

2.3.2 Gaussian Smoothing

The **Gaussian blurring** or **Gaussian smoothing** is a common filter technique employed in image processing, aiming to reduce noise and smooth the borders of images [GH11]. The Gaussian smoothing is a form of kernel convolution: A convolution matrix, essentially a grid of values, is passed over a matrix or array of numbers representing the input data. In the one-dimensional case, most relevant in the context of this work, the kernel is a one-dimensional window, which is passed through the array of values. The input is subsequently transformed, adjusting the values with respect to their local neighbours by producing weighted averages. The transformation and the weighting depend on the type of kernel and in the case of Gaussian smoothing, the Gaussian function [3] is applied for the transformation. The Gaussian function in one dimension can be expressed as

$$g(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp^{-\frac{x^2}{2\sigma^2}}$$

where σ denotes the standard deviation. σ denotes the window size of the kernel, i.e. the size of the neighbourhood to consider for the weighted averages. In image processing, the Gaussian blur is applied to all pixels of an image, weighting pixels at the center of the kernel more heavily, while the pixels on the borders are roughly preserved. The Gaussian blur thus is commonly applied when edges and boundaries are to be preserved. The Gaussian blur further acts as a low-pass filter, reducing high value components while aligning lower value components in the surrounding neighbourhood. In our approach, we borrow the image processing technique and apply it in the

[3] https://en.wikipedia.org/wiki/Gaussian_function

context of novelty in textual contents. Chapter 4 discusses the usage of Gaussian smoothing to average out a sequence of novelty scores, subsequently influencing the outcome of the background visualisation of novel text regions.

2.3.3 Node-Link Diagram

Node-link diagrams are a frequently utilised visualisation technique to present graph or graph-like structures. Given an acyclic, directed graph $G(V, E)$ with vertices $v \in V$ and edges $e \in E$, a corresponding node-link diagram maps the vertices to visual objects called *nodes* and visualise the edges as *arcs*. Nodes represent *data points* and can be presented in multiple forms, circular and rectangular forms being common choices. Edges are utilised to connect nodes and signify their *relationship*. A node-link diagram can thus be viewed as a visualisation of a network of data points and their relationships, where the explicit presentation of edges allows to track paths formed by interconnected nodes. The specific choice of form and dimensions of nodes and edges is commonly utilised to encode additional information, for example the volume of a data point in the size of the node or the relevance of a connection in the width of an edge. Ghoniem et al. [GFC04] denote certain advantages and disadvantages of the node-link visualisation:

- Node-link diagrams are easy to read, as nodes and edges provide an intuitive representation of data points and their relationships
- The explicit visualisation of relationships and connections allows the simple traction of sequences of connected data points, further providing an intuitive way to display hierarchies.
- The readability of node-link diagrams depends heavily on the size of underlying data set and the relationships among the data points, as drawing a huge amount of densely connected nodes induces heavy visual clutter.
- Node-link diagrams thus scale relatively poor with increasing data set sizes, if left unmodified.

Chapter 4 discusses how we utilise the node-link diagram in the context of topic evolution, but modify the resulting visualisation with interactive mechanisms to reduce clutter and scale well for large data sets.

3 Related Work

The following chapter describes relevant scientific literature closely related to the approaches presented in this work. Three areas are of central interest:

- Detecting and visually emphasising passages within documents qualifying as novel and their insights
- Clustering documents according to representative topics extracted from the documents
- Visualising the evolution of the extracted topics, with respect to their temporal aspects and emphasising on continuity and the lack thereof respectively

A range of works addressing the aforementioned areas of interest will be analysed, describing both similarities and discrepancies in their approaches compared to this work.

3.1 Detecting novel information in text

Works on novelty detection in text corpora are plentiful, but often differ vastly in their methodology.

The Distance, or inversely the similarity between documents, is a commonly used metric to assess the novelty of documents with respect to their history. Earlier works have been studying different similarity measures based on the shared word content between documents. Zhang et al. [ZCM02] for example evaluate five novelty measures based on the similarity of documents, among which distributional similarity measures are of high interest for this work. While Zhang et al. explore differences in word distributions retrieved via Maximum Likelihood Estimation, this work utilises probability distributions from neural network based language models. The authors further state that the cosine similarity measure performs most stable among the tested measures. Though cosine similarity finds no direct usage to detect novel text passages in this work, it is utilised heavily to compare topic clusters later on.

When assessing the similarity of documents, the documents can either be compared as one entity or segmented into their components: According to Tsai et al. [TZ11], novelty can be measured on different levels, including document-level novelty and sentence-level novelty. The authors describe a model to compute a document's novelty with respect to the proportion of novel sentences in the document. Documents are segmented into sentences and sentences compared to their history using the cosine similarity metric. The novelty of a sentence is set with respect to the highest similarity to a sentence from its history. Subsequently, the document novelty is calculated as the proportion of novel sentences compared to all sentences and a document assessed as novel if the resulting rate exceeds a predefined threshold. In a comparable manner, the work presented here calculates probabilities for each sentence of a document using its neural network based language model and assesses the novelty of a document with respect to the aggregated probabilities of its sentences.

Closest related to the novelty detection approach presented here is the work by Knittel et al. [KKE18]. The authors propose a visual analytics approach utilising character based Long-Short-Term Memory (LSTM) recurrent networks to highlight novel text passages. Using the LSTM recurrent networks, the authors build character-level language models to assess the likelihood of text fragments and sequences, defining novelty with respect to the probabilities assigned by the models. Knittel et al. argue that low probabilities assigned by complex language models stem from less predictable character or word sequences, indicating novel and more interesting information contained in such regions. The authors focus on character-level dependencies, training their model by feeding characters step by step into the LSTM recurrent network. They subsequently predict probability distributions $P(c_i|c_{context})$, where c_i is the character in the current step and $c_{context}$ the preceding context. In contrast, this work utilises OpenAI's GPT-2 [RWC+18], a pre-trained, transformer based language model. The model generates word-level probability distributions by predicting a token's probability given its predecessors for a given sequence $w_{i-n}, ..., w_i$, conditioning each word token on its preceding left context. The usage of a pretrained language model is motivated akin to Knittel et al.: The language model's inability to uniformly predict among the words of a given sequence indicates at novel content the model has not "seen" yet, resulting in low probabilities for these text regions. With their character based LSTM recurrent network, Knittel et al. capture and incorporate subword information like morphological variations into the computed probability distributions. This allows a more fine grained and robust assertion of novel words, as the model is more likely to recognize and assign similar probabilities to variations like *perfect* and *perfectly*, even if the model has not encountered one of the variations beforehand. In contrast, word based language models, as used in this work, are trained on *word embeddings* and word vectors as introduced in chapter 2, relying on the underlying tokenization and word selection to reliable discern different variations. As such, the frequency of encounters with specific word variations influences the model's ability to predict them, resulting in lower probabilities for previously unseen variations of similar words. As sequences within text regions grow in length, character based language models have to process more tokens over more time steps to capture long range dependencies. If the preceding context is chosen too short in length, the model is more likely to inaccurately assess later subsequences as novel. Word based language models cover larger sequences with smaller frictions of processed tokens, thus requiring a smaller preceding context. To summarise, both works investigate the application of trained, neural network based language models to detect novel text regions. While Knittel et al. extract insights from character-level dependencies, the approach presented here analyses on the word-level how the underlying language model's confidence in predictions correlates with novelty in text.

3.2 Visual representation of textual content and insights

Different visualisation techniques have been explored to visually present the textual content in a manner that allows to quickly gather relevant insights. Word clouds find usage in many applications aiming to summarise text corpora. The technique commonly extracts the most frequent words of a corpus and weights the word font with respect to the frequency. Each keyword is algorithmically placed on a given canvas, often optimizing the space usage while encoding the importance of each word in its position [BGN08]. Word Cloud Explorer [HLLE14] and SparkClouds [LRKC10] are two examples building more complex frameworks with and around word clouds, pairing world clouds with natural language processing and filtering methods or integrating and conveying trends

between word clouds respectively. Word clouds are not directly utilised in this work, as several studies, for example Hearst et al. [HPP+20], Felix et al. [FFB18] and Heimerl et al. [HLLE14], have evaluated shortcomings of word cloud visualisations: The lack of a natural reading order and layout dependant difficulties in grouping topic related words being two main examples contributing to the decision. Still, two core concepts of word clouds can be found in works discussed later on, as well as the approach presented here: Extracting important keywords from text corpora utilising a selected metric for importance, for example similarity or novelty and displaying the keywords in a layout targeted at preserving additional context like temporal ordering.

Strobelt et al. [SOR+09] developed a compact representation of the key content of documents based on a mixture of document images and keywords. The authors extract descriptive images and key terms in two separate multi-step processes to produce document cards, as the authors call the compact visual summarisation of the underlying document. Document cards consist of up to four images, sized by their semantic weight and iteratively positioned on the canvas utilising a packing algorithm. After placing the images, the rest of the canvas is populated with key terms, the number of terms chosen according to the available space to avoid overcrowding. Important terms are oriented towards the middle of free canvas spaces, where position and size of a term depend on its relative weight with respect to the document the term originates from. The approach presented in this work employs a comparable visualisation to summarise the novel content of a document. After the underlying language model predicts the novelty of each sentence of a document, the most important keywords of every sentence are extracted. The importance of a keyword in this case is evaluated by calculating TF-IDF scores for every word in the corpus. Documents are subsequently presented in a compact, ”card-like” fashion, akin to the document cards by Strobelt et al. As this work only processes the textual content of the given documents, the entire card is utilised to present all of the sentences of a document. Unlike the approach by Strobelt et al., no textual information is displayed at first, but sentences visually encoded as horizontally stacked bars. The bars are ordered by their appearance within the original document and colored according to their novelty. On interaction with a sentence bar are shown the most important keywords from that sentence. Thus, while Strobelt et al. focus on summarising the semantic content of a document in a compact visual representation, the card-like visual encoding of this work emphasises first the distribution of novel content within a document and provides an aggregation of the important content on demand.

Hoover et al. [HSG20] presented exBert, an interactive visualisation tool to investigate the contextual text representations learned by transformer based language models. Transformer language models [VSP+17] incorporate self attention layers, modelling the relevance of the remaining tokens of an input sequence for each specific token of the sequence and thus capturing contextual relationships. Hoover et al. provide three views in exBert to analyse the development of internal word representations as they pass through each attention layer: An attention view to explore the attention layers and the attentions between input tokens, a corpus view to display searched tokens within an annotated corpus and a summary view presenting histogram summaries of metadata encoded in the searched corpus. In contrast to exBert, the approach presented in this work does not aim to provide insights into how its underlying language model predicts novelty, but rather visualises the results of its application in novel text detection. That said, this work adapts some visualisation techniques found particularly in exBert’s attention and corpus view. Among other functionalities, the attention view provides a duplicated, vertically aligned view of the tokens of a given input sequence, as well as the attention relationships between each token pair, drawn as edges. On selecting a masked language model and interacting with a token, exBert shows a small window with predicted alternative words at that spot, given the left and right context of the input

sequence. In a comparable fashion, when interacting with sentences in the dedicated article view of the framework developed in this work, a side panel will display alternative sentences the underlying language model predicted, given the preceding left context in the article. The corpus view of exBert displays the top 50 most similar tokens found in the annotated corpus for a given query token, shown within their context in the corpus and sorted by similarity. exBert colors the token backgrounds in the corpus view, where the opacity correlates with the attention score for the specific token, highlighting the contextual relevancy of each token. Likewise, the approach presented here colors the background of the sentences shown in its dedicated article view, mapping the model-assigned probability to the opacity and thus emphasising the novelty of each sentence.

Knittel et al. [KKE18] offer different visualisation modes to enable customizing the appearance of novel text passages and their surroundings. The authors compute the odds of a character not being sampled by their model and map these odds to the saturation of the background color or the transparency of the text. Akin to Knittel et al., this work first normalizes its computed model novelty scores between 0 and 1 and subsequently utilises the scores as inputs to a continuous color scheme for the background colors. In the work of Knittel et al., the probability dependant colors can be mapped to the background or the foreground. Furthermore, the colors can be smoothed by either applying a Gaussian blur or averaging the probabilities sentence-wise, allowing the colors to extend to and emphasise not only a specific novel sequence, but also its surrounding context. Similarly, in this work, the normalized model scores can be either mapped directly to the background color of text passages or Gaussian blurred sentence-wise before the mapping. This enables varying degrees of locality when emphasising the surrounding context, which Knittel et al. achieve differently by adding the ability to change the radius of the blur or completely hide sentences that do not meet a specified threshold. Contrary to Knittel et al., this work always maps the resulting colors to the background of the text, while coloring of the text itself is reserved for interactive highlighting of specific words using different, non-conflicting colors. While Knittel et al. provide a dedicated view to inspect novel text passages only, the approach presented in this work further presents multiple perspectives to investigate the insights gained from novel text passages, showing the distribution of novel sentences, model predicted alternative text passages, the temporal distribution of documents and the evolution of topics in the document corpus. Overall, Knittel et al. focus on visually encapsulating the novelty of local text passages in a document and provide interactive tools to further emphasise and distinguish a passage's novelty compared to its textual neighbourhood. While the approach presented in this work adapts the text highlighting scheme utilised in the work by Knittel et al., it extends the visual and interactive capabilities to a global context, incorporating exploratory topic visualisation and insights gained from the temporal evolution of the document corpus.

3.3 Topic Modelling: Identifying novel and recurring topics

Latent Dirichlet Allocation (LDA) [BNJ03] is a generative probabilistic model of a corpus and an often utilised method for topic modelling. Documents are assumed representations of random mixtures over latent topics and every topic is associated with a distribution over words. LDA is commonly applied in topic modelling to retrieve topics from a collection and classify the relevance of each document for the extracted topics with respect to the document's word distribution. An application example relevant to this work is Dynamic Topic Models [BL06], a topic modelling

approach for time series data. Dynamic Topic Models extracts topics in a sequentially organized corpus of documents, akin to the time ordered news article corpus the approach presented in this work operates on.

Non-negative Matrix Factorization (NMF) [PT94] is a matrix factorization algorithm for finding the positive factorization of a given positive matrix. NMF finds usage in topic modelling as a dimension reduction and clustering method, where a document corpus represented as a term-document matrix is factorized into a document-topic matrix and a word-topic matrix. Xu et al. [XLG03] show in their work how NMF can be utilised effectively to cluster documents according to latent topics. The approach is adapted similarly in the work presented here to determine topics of time ordered document clusters.

The following works mostly utilise one of the aforementioned topic modelling algorithms, in conjunction with a recurrent procedure: A topic analysis technique is employed to reduce the dimensions and complexity of the corpus, i.e. mapping documents to representative sets of keywords. Topics are subsequently extracted from the alternative document representations and further processed, e.g. to incorporate time.

TIARA [WLS+10] is one example, where Wei et al. employ such a procedure to first summarise a collection of documents into topics and afterwards interactively visualise the evolution of resulting topics over time. The authors framework processes a given document corpus in multiple steps, applying first LDA or document clustering to derive document-topic and topic-word distributions from the corpus. Wei et al. pre-define a threshold and assign topics to documents, where the document-topic probability exceeds said threshold. The approach presented here determines the number of topics to extract for a document cluster based on the average pair-wise cosine similarity between all documents and subsequently assigns each document the topic with the highest document-topic probability for the given document. Alternativey, the prototype developed in this work allows the user to manually define the number of topics to search for in a specific article cluster. Following the topic analysis, the authors rank the resulting topics by importance, where they specify the importance with respect to the topic content coverage and the topic variance. As a result, in TIARA topics are ranked higher if they cover more content within the corpus, while appearing in only a few documents. The approach presented in this work focuses less on content coverage and more so on the content of the documents assigned to a topic. Both TIARA, as well as the approach presented here, aggregate documents with respect to the time intervals they are associated with. While Wei et al. create dynamic intervals adjusted around (peak) topic activity, this work accurately segments the corpus according to the publication date of the news documents, while offering the possibility to group clusters of multiple, subsequent segments.

Pokharel et al. [PHJG19] analyse and visualise the evolution of topics across multiple social media platforms. The authors present a framework that utilises LDA to extract topics as well as term significance values for every associated term. Topics within the same time window are ranked according to the standard deviation of the topics, i.e. a higher standard deviation proposes a ”more” emerging topic. Topics across different media sources are then compared with respect to similarity, where similarity scores are constructed based on the semantic similarity of a topic’s associated terms. This work compares and ranks topics in a comparable manner, albeit the applied techniques differing in certain aspects. The approach presented here utilises tf-idf based keyword vectors from the documents to extract topics using NMF. The cosine similarity between the topic’s associated keyword vectors is subsequently used to model and visualise the evolution of topics within and across adjacent time windows. The preference of cosine similarity over semantic similarity as the

comparison metric, is motivated by the assumption that news articles within the same time window share multiple words and phrases, as they cover similar or the same events. As this induces very similar tf-idf vectors, the cosine angle between the underlying documents will be small, hinting at repeating content. Conversely, outliers with respect to the cosine angle, having a high distance to "neighbouring" documents, indicate novel content and thus aligning with the central focus of the approach of this work.

TopicFlow [MSH+13] by Malik et al. is a comparable approach, where the authors analyse twitter stream data over time to extract complex topical trends in twitter discussions. The approach applies binned topic modelling, a novel statistical topic modeling application presented by the authors, generating topic models independently for adjacent time slices of data. Bins represent time slices of equal width, where each bin holds an unrestricted number of tweets. Similar to Pokharel et al., the authors apply LDA to each bin, retrieving word distributions for every topic and topic distributions for every document. Both TopicFlow and the approach presented here allow changing the granularity of the modelling by scaling the size of the bins/groups and adjusting the number of retrieved topics per bin/cluster. Additionally, this work offers the option to automatically determine an appropriate number of topics by choosing the number proportional to the average similarity of the documents of the respective cluster. Akin to the approach presented here, Malik et al. utilise cosine similarity to align topics and subsequently visualise emerging, converging and diverging topics interactively.

Choo et al. [CLRP13] propose UTOPIAN, a visual analytics system for topic modelling centered around semi-supervised NMF. Akin to the approach presented here, the UTOPIAN framework performs NMF on a given data set to retrieve relevant topics, which the authors subsequently present in an interactive node-link diagram based visualisation. The authors prioritize NMF over LDA due to its consistent and deterministic nature, as they allow user provided inputs for NMF to alter the modelling output. Where NMF originally performs on the input matrix $N \in \mathbb{R}_+^{m \times n}$ and a non-negative integer k to produce factors $M \in \mathbb{R}_+^{m \times k}$ and $F \in \mathbb{R}_+^{k \times n}$, the semi-supervised approach by Choo et al. adds reference matrices $V \in \mathbb{R}_+^{m \times k}$ and $G \in \mathbb{R}_+^{k \times n}$ for M and F respectively. The authors further include diagonal matrices assigning weights on the columns of V and G to penalize differences between M and V and F and G respectively. As a result, by changing the reference and diagonal matrices, the output matrices M and F are regularized, augmenting the resulting topic model with prior user knowledge. While not semi-supervised, the approach presented in this work augment the inputs for NMF as well, imposing restrictions on the resulting topics. The document-term matrix representing the documents and acting as the input matrix N, is constructed based on modified TF-IDF vectors: The score for every word of a document incorporates the novelty scores stemming from the probabilities assigned by the underlying language model. Thus, while UTOPIAN adds user provided prior knowledge to NMF to refine topic keywords or merge or split topics, the approach presented here adds context about and restrictions with respect to novelty. For example, modifying the inverse document frequency with the average novelty scores of the affected document, induces greater influence to keywords appearing in more novel documents, as assessed by the underlying model.

With Story Tracker [KNMK13], Krstajic et al. present a visual analytics system for temporal analysis of news stories. The authors develop an incremental approach that extracts stories from online news streams, identifying and visualising stories that split and merge over time. To achieve this, they cluster articles in consecutive 24-h time intervals, comparing neighbouring clusters sequentially to analyse the evolution of new, recurring and overlapping topics. In contrast to the works presented

before, the clustering is employed using phrase based cluster algorithms, with the core assumption that similar articles are more likely to share phrases. While the assumptions are similar, contrary to Krstajic at al. the system presented here generates descriptive topics from representative keywords extracted directly from the documents. StoryTracker further compares stories from time adjacent clusters by calculating the Jaccard distance between clusters, using the story title, document titles and descriptions. In this work, every keyword is assigned a modified tf-idf score, thus topics are subsequently compared via the cosine similarity of their respective keyword vectors. To show the evolution of topics, Krstajic et al. set empirically chosen thresholds to connect and present stories as merging or splitting depending on their cluster similarity scores. In a similar fashion, the approach presented here models the evolution of topics by comparing the aggregated similarity scores of articles assigned to time adjacent topics.

3.4 Visualising the evolution of topics

Krstajic et al. [KNMK13] define three requirements to effectively visualise the topical variations in a corpus of documents over time: Encode the flow of time in an understandable manner, encode the importance of themes and events as they evolve over time and incrementally incorporate new data with as little (major) changes to existing visual representation as possible.

ThemeRiver [HHN00] makes use of a river metaphor to effectively encode the flow of time as well as the importance of individual themes. The authors map time to the horizontal axis, while each vertical section represents an ordered time slice. Themes, defined as single words, are then depicted as colored currents flowing horizontally, where the thematic strength is encoded in the width of the current.

ThemeDelta [GJG+15] builds upon a comparable visualisation to display temporal trends among time-indexed textual data sets. The framework by Gad et al. performs dynamic temporal segmentation and topic modelling to determine and focus on significant shifts in topics. Akin to the aforementioned river metaphor, ThemeDelta utilises sinuous trend lines with variable width to depict trends across time, encoding the prominence of a trend at a particular time in its line-width and the category in its color. Similar to ThemeRiver, trend lines flow from left to right on a horizontal time axis, where the horizontal space along the axis is divided equally among segments and similar trends grouped to topics along the vertical axis. Unlike ThemeRiver, ThemeDelta uses splines to create smooth curves and communicate a better perception of continuity.

The method of illustrating the flow of topics along the horizontal or vertical axis has been adopted in a similar fashion by the following works, as well as the approach presented in this work: The TIARA [WLS+10] framework visualises the topic evolution in a stacked graph based layout, first generating keyword word clouds as summaries of the underlying topics and aligning the word clouds along the horizontal time axis. Topics are then stacked vertically as layers, while the number of associated documents is encoded in the height of each topic layer. Malik et al. utilise a node-link diagram based flow diagram in TopicFlow [MSH+13], visualising the topic evolution as a horizontal graph segmented into time slices. The nodes of the graph represent topics and edges between neighbouring time slices depict the topic similarity. The framework maps the number of associated documents to the size of the topic nodes, while the similarity between topics is reflected in the strength of the edges. StoryTracker by Krstajic et al. visualises its topic evolution in a similar manner, placing in its main view lists of daily story clusters of a specific time frame along the horizontal axis and

connecting clusters via shapes based on Bézier curves. Story clusters are ordered within each list by their cluster strength, choosing rectangular representations for the stories. As before with TIARA and TopicFlow, the height is mapped proportionally to the number of articles associated with that story. Visual mappings as described before can also be found in the approach presented in this work, albeit embedded into a different layout: The topical evolution over a specific time frame is visualised as a vertical graph, showing only the relevant topics per time step as rectangular nodes and their relationships to time adjacent topics as edges. Similar to the aforementioned works, the height of each node describes the strength of the cluster, while the width of edges between time adjacent clusters denote the cluster similarity. In contrast to the foregoing works, time frames can be set arbitrarily by selecting a day-wise or month-wise segmentation, where subsequently, multiple time adjacent topic clusters can be variably grouped together to form one time segmented cluster. While the previous works focus on a flow like, left to right alignment, the vertical orientation of the graph in the approach presented here is chosen specifically to ensure the scalability of the visualisation with respect to the dynamic and varying sizes of the clusters. The layout further allows to utilise the remaining space on both sides of the graph to incorporate more context and perspectives.

A common approach to provide multiple perspectives and a wide range of interactive tools is to include multiple dedicated and possibly interconnected views. TIARA does so by including a view besides its stacked graph visualisation to inspect documents containing specific keywords. TopicFlow provides four coordinated views, supporting its topic overview with additional context: The flow diagram is accompanied by the topic panel on one side, presenting the discovered topics in the underlying data group. On the other side, the corresponding documents can be inspected in detail in the tweet panel, alongside statistical information like the topics with the highest probability for each associated document. The final and fourth view allows interactive filtering of topics. In a comparable, but slightly different approach, both StoryTracker and the approach presented in this work combine multiple perspectives, increasing the detail as perspectives change. Story tracker's overview provides a broader, less detailed temporal context showing news stories and their relationships over time, while the main view shows the topic evolution for a specific time frame as described before. This work utilises its vertical graph based topic evolution visualisation as a starting point to explore the general topic distribution for arbitrary time frames and cluster groupings. Akin to the topic panel in TopicFlow, a list representation of all topics shown in the overview is provided. Additionally, topic clusters in the overview graph can be individually selected or grouped and subsequently examined in a cluster view. The cluster view allows to inspect and compare selected topic clusters in more detail, displaying the associated article distribution and the most important keywords functioning as summaries of each topic. The keyword evolution can then be further analysed in a visualisation comparable to the main view of StoryTracker: Article keywords are vertically aligned in lists and the lists horizontally placed according to the article order, forming a time-ordered cloud of word lists. In contrast, neither of the aforementioned related works provide a dedicated view to further inspect the keyword evolution for an associated topic. StoryTracker's three level view is completed by an article view, showing selected articles and their detailed information, akin to TIARA and its interactive keyword view. Similarly, in the approach presented here, keywords in the keyword view can be selected to open an aggregated article view showing sentences containing the selected keyword. In summary, while TIARA and TopicFlow enrich their topic visualisation with a single keyword view and multiple views providing textual and statistical context for the displayed topics respectively, StoryTracker and this work present additional information across multiple levels, starting at a broad overview and consecutively increasing the detail as the data is narrowed down. Where StoryTracker begins at an overview of topics and

finishes the zoom down on the data at the level of a single article, the approach presented here allows the user to begin at the overall temporal evolution of topics and interactively explore groups of topics, specific topics, clusters related to topics, articles within these clusters up until the keyword evolution of specific articles within these clusters.

StoryTracker, TopicFlow, as well as the work presented here approach to visually emphasise the emergence, continuation and divergence of topics. In StoryTracker, time adjacent story clusters with a similarity score exceeding a predefined threshold are visually encoded the same, to emphasise on topics ”splitting” from the same parent. Extending the emphasise on evolving stories to its main view, the framework only colors and connects stories if they evolve over several days, while stories appearing on a single day remain gray. Closer to the approach presented here, TopicFlow displays the convergence or divergence of topics via multiple paths that enter and leave topic nodes: The authors draw connections between any topic node pairs, if the cosine similarity exceeds a specified threshold, thus not only comparing the most similar topics with each other. Additionally, topic nodes are colored according to their evolution state, i.e. whether the associated topic is emerging, continuing, ending or a standalone topic. Where TopicFlow orders its topic nodes by size, thus displaying the most prevalent topic at a specific time at the top, the approach presented here utilises the layout of nodes in its vertical graph based topic overview as a way to visualise the continuity of topics or the lack thereof. After determining all prevalent topics in the clusters to be displayed, the topics are horizontally ordered the same for all clusters: If a topic appears within a cluster, a node will be drawn, otherwise the space will be left empty. As the cluster topic nodes are vertically stacked according to the flow of time, the emergence, continuity and disappearance of topics can be directly tracked. Similar to TopicFlow, this work furthermore utilises the connections between topic clusters to emphasise the evolution. For that, two edge weighting mechanisms are provided: The similarity weighting draws edges between every pair of adjacent cluster topics and maps the cosine similarity to the thickness of the edge. The keyword weighting displays converging and diverging topics, i.e. edges are drawn between clusters if and only if they share at least one associated keyword, while the thickness of the edge depends on the exact number of keywords shared. Additionally, in the keyword view, the user has the option to filter keywords with respect to their continuity in two ways: Either display discontinuing keywords appearing only once across the current group of temporary ordered documents, or display only continuing keywords appearing in at least two documents or more.

Several interactive tools for explorative analysis are incorporated in the approach presented here, as well as the discussed related works. In TIARA the authors allow the user to merge and split topics, comparable to the grouping of topic clusters in the approach presented here. In both cases, the underlying system adjusts its topic distribution output with respect to the changes, grouping for example a set of distinct topics into one representative topic. TopicFlow provides a search functionality to locate topics containing specific keywords and displays the result both in the topic panel and the flow graph. Additionally, the filter panel allows to filter topics depending on their size or their evolution state and filter edges depending on similarity. StoryTracker allows to adapt the outcome of the visualisation by changing the clustering algorithm, the weighting of keywords or the sorting of topics within clusters, adapting the focus selectively on the intra cluster similarity or the importance of individual clusters for example. In a comparable manner, the work presented here provides an interactive menu to adjust the time frame as well as parameters like the number of topics to extract per cluster, the number of articles to group per cluster or the edge weighting scheme. Similarly motivated, this work pairs its keyword view with a collapsible menu providing a filter, sort and a search mechanism to aggregate, to isolate or remove keywords of specific topics.

StoryTracker additionally approaches to reduce clutter by incorporating the filtering of connections between clusters depending on the strength of the connection and optimizing the layout. In addition to a comparable edge filtering mechanism, the work presented here provides a cluster filtering mechanism: To fully eliminate particularly cluttered regions within the graph, individual time steps including all the afflicted topic clusters and their connections to time adjacent clusters can be collapsed. Overall, the approach presented here adapts and extends the interactive tools provided by the discussed works, allowing to individualize and filter the results of the visualisation on multiple levels.

4 Conceptual Design & Realization of the Prototype

In this chapter, we dissect the conceptual decisions underlying the prototypical realization of the proposed visual analytics approach. We first define the requirements to fulfill with respect to novel content detection, topic evolution modelling and building the interactive visualisation framework. Afterwards, we describe in detail how the design and architecture of the prototype are tailored towards the defined requirements, explaining how each decision contributes to each task at hand. Each section dedicated to a core component of the approach presents in detail the task it aims to solve, the conceptual solution it proposes and the abstract implementation of the component in the prototype. We finally combine each of the building blocks of the presented approach and present the resulting interactive visualisation framework consisting of multiple coordinated views.

4.1 Requirements

We define the goal of the approach as realizing an interactive visualisation system for novel text and topic evaluation analysis in time-ordered corpora. At the center of the approach stands the detection of novel content in sequences of text, determining the data of interest among the extensive, but not necessarily informative inputs. The extracted data is to be subsequently presented in multiple, interactive views providing different insights, with two major, functionality defining points of focus: Assisting the user in the comprehension of texts and extraction of relevant, new information, as well as enabling the user to grasp the evolution of the corpus content over time, presenting key topics and how they evolve. We can thus define a set of requirements to fulfill with respect to each core functionality, as we aim to realize the overall approach:

Finding novel text passages Central to the approach presented in this work is the system's ability to process text documents and identify the passages containing novel content with respect to the document's past. Given a sequence of documents $d_1, ..., d_{n-1}, d_n$, the system should be able to determine the passages containing novel content in document d_n, after seeing the contents of the previous documents. This identification of novel content can be based on new words, entities and phrases and sentences the system has not observed before, actively distinguishing from synonyms, words of similar meaning and reformulations.

Present & highlight novel text passages Having extracted novel text passages, the content needs to be displayed in a way such that novel and already seen contents are easily distinguishable. Furthermore, novel content needs to be visually highlighted to stand out and catch the attention first, guiding the viewer while easing and fastening the reading process. As multiple documents are

considered, the system should provide a summarising view of individual documents, presenting only the novel and most important words or sentences at first and providing the reminder of the documents on demand.

Identify representative topics To inspect the key themes shared by multiple documents of larger document sequences, the system should incorporate a topic modelling component, finding the latent topics in a given document (sub-)set. To emphasise the influence of documents containing novel content, the system should enable to modify the results of the topic modelling procedure by primarily attending to novel text passages when determining topics related content in documents. For very large data sets, where it is infeasible to view all documents and their topics at the same time, the system should enable user-defined document groups, clustering documents within a specified time frame and subsequently determine the topics of these dynamic document groups.

Present the evolution of topics over time To provide an overview over the entire data set or sequences of documents over long time periods, the system should provide a view reducing the underlying data to its main topical content and features. The overview should illustrate the prevalent topics at a specific time, showing the most important, topic related keywords for example. The visualisation should enable the viewer to identify newly emerging, continuing and disappearing topics as the document clusters are explored along the timeline. If a specific topic or group of documents is to be examined in more detail, the system should provide the functionality to select such topics or groups and display additional information and details on demand.

Multiple Coordinated Views Providing all the visualisations and functionalities described so far at once, might not only overload the user with information, it might also be infeasible to present the sheer amount of details involved, especially if the underlying data set is large. As a solution, the system should integrate the various sources of information into different views. Each view should aggregate functionalities and visualisations providing insights of similar type and similar usage. The arrangement of the views should reflect a natural workflow guiding the user, for example in a multi-level visualisation starting with a low level view on singular items and objects and ending with a high level overview over a specified time frame.

Interactive filtering and search Complementing the multiple coordinates views, each view should provide filter and search functions to explicitly narrow down the scope of data presented to the user, with respect to specific items, topics or properties of interest. The filter mechanisms should allow to adjust the provided functions and augment the presented results: Restrict and expand the time frame and articles to extract the information from, change the way novel text is visually highlighted, specify the topics to visually track or isolate specific contents with respect to a given search query and more.

4.2 General Design

Designing a prototype for the proposed approach, the requirements as stated above suggest to model the prototype around two major components, dedicated to the novel content detection and to topic modelling respectively. As the aim is to provide multiple dynamic views and perspectives on the data, revealing insights that may be hidden in a static presentation, the user predominantly decides the outcome of the data processing and subsequent presentation. Figure 4.1 illustrates how the general design thus adopts the *human-in-the-loop* [EHR+14] approach: The user is given the roles of the main initiator and main driver of the data processing and visualisation procedures. As the data passes through each processing pipeline, the user can adapt the parameters of the applied functions to determine what information is extracted from the data, while directing any information exchange between the pipelines. The user is given the tools to interact with each of the views presenting the novel textual contents and topics found in the corpus, both deciding on the data or information to compose the currently displayed visualisation from and directing the information flow between views. Such tools include filter and search mechanisms, a fluent switch between different views, as well as the capability to select and deselect visual components to temporarily show or hide the corresponding visualisation.

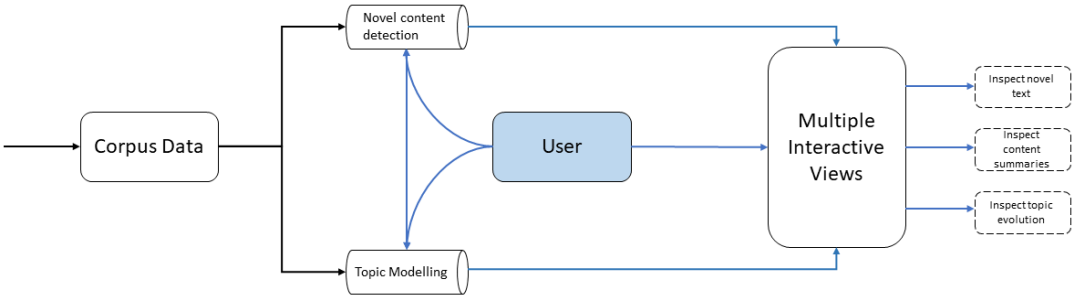


Figure 4.1: The general, abstract design for a prototypical realization of the approach. The user-driven approach places the user at the center of the system, influencing the output of and the exchange between the novel detection pipeline and the topic modelling pipeline. The user then dictates the outcome of the presentations of the outputs, interactively changing the data to present and the properties to highlight.

4.3 Prototype Architecture & Realization

From the previously discussed general design, we can derive the possible architecture of a prototypical implementation. Two main components will be responsible for the processing and transformation of the data related to the major functionalities of the prototype: A component detecting novel content among the supplied texts and a component finding topics and modelling the topic evolution. Both main components can be decomposed into subcomponents consisting of procedures to process the various data sources and to supplement the inputs with additional data and user input. Figure 4.2 illustrates the resulting architecture of the prototype. A **web crawler** fetches the unprocessed articles to construct the initial **news corpus**. The **text processing pipeline**

subsequently transforms and cleans the articles, producing a corpus of documents ready to be presented as they are or to be further processed. The major novelty detection component on the right side of the figure utilises the **GPT-2 language model** to predict novel content, which forms the basis for the visualisations and data presented in the *article view* and the *summary view*. The major topic modelling component on the left gets as inputs the **tf-idf vectors** produced from the cleaned corpus, optionally modified by the predictions of the language model. Taking into consideration any user defined restrictions or groupings with respect to the articles involved, the **topic clusters** and **topic keywords** outputs are interactively presented in cascading degrees of detail in the *topic graph*, the *cluster view* and the *keyword view*. The following sections explore each of the components shown in figure 4.2. We discuss in detail the problem the component attends to, the design decisions to solve the task at hand and the abstract realization of the component.

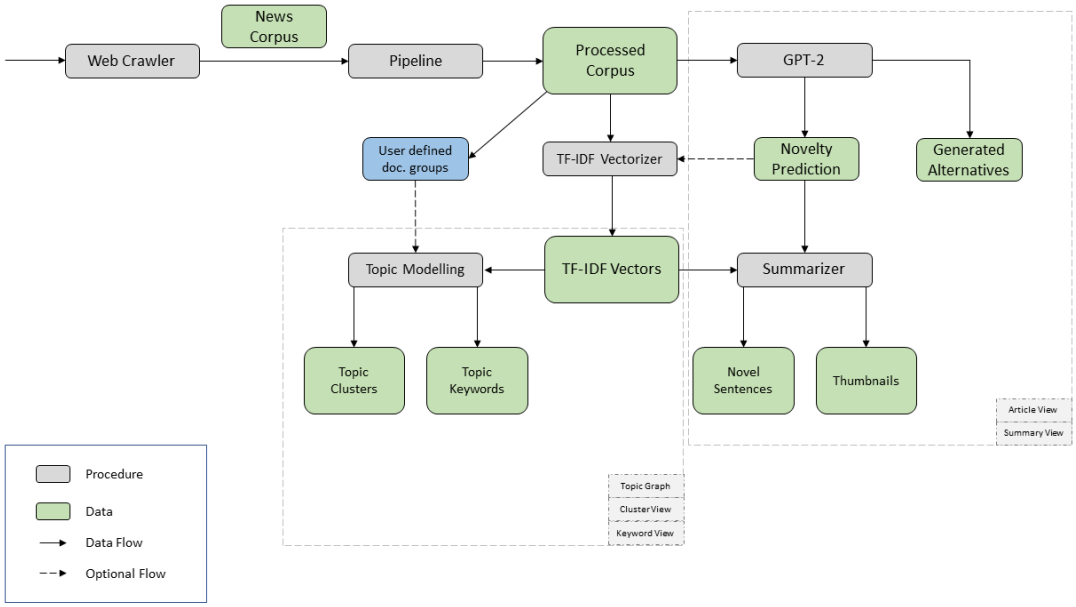


Figure 4.2: The architecture of the prototype is mainly defined by two main components and their procedures invoked on different sources of data. At the heart of the right component, GPT-2 is utilised to predict novel text passages and the results are subsequently presented in the *article view* and *summary view*. The left component employs topic modelling to produce a topic overview shown in the *topic graph*, while the *cluster view* and the *keyword view* allow to explore topics in more detail.

4.3.1 Data

While the proposed approach holds the potential to be utilised as a novel text detection and topic evolution solution for arbitrary sequences of temporary ordered text documents, the prototype developed in this work is based and tested on a corpus of news stories. Recalling chapter 1, we quickly touched upon the structure and properties of online news articles: News stories are reported by multiple providers at once and important topics or events might last for a long time, being covered across multiple articles, gradually shifting from a central topic to a side story as public

interest recedes. Depending on the situation, a story or event might attract major public interest over a longer time period. Yet, there might be a lack of new content to report if the supply of new information is lacking, prompting news providers to revisit formerly covered content from different perspectives. News articles collected from multiple news providers over a long time span are thus likely to contain clusters of articles with similar content, often sharing the same key words and reformulated phrases. This important property of news story corpora plays a significant role in the selection of neural network based language models to detect novel content.

The news data set in question consists of 300 000 to 400 000 digitized, English news article stories crawled from multiple online news outlets. Each article in the corpus thus corresponds to a single news story. The unprocessed corpus is unordered, though each article is accompanied by necessary meta data to structure the corpus:

- A **publishing date** in various formats.
- A **retrieval date** from the web crawler in various formats.
- The name of the **news provider** and a **URL** linking to the web page.
- The article **title** and the article **content**, both as strings.

The article content has minimum structure according to the formatting of the corresponding displaying web page. After employing the processing steps described in chapter 5, the processed corpus contains articles of the following form:

- The title is a tokenized and lemmatized list, or vector, of words.
- The content is a list of sentences, each sentence a vector of tokenized and lemmatized words.
- The uniform date corresponds to the article's publishing date or the retrieval date, if the former is not provided.
- Two separate fields contain the normalized provider name and URL.

4.3.2 Novelty Score calculation with GPT-2

We have established in the sections before that articles reporting about similar events and topics will consist of similar and recurring keywords and phrases. It seems thus logical, to define *novel content* as passages of text containing content not encountered in the preceding context of said passage. The new content can be new words that are not part of the vocabulary of the preceding context, possibly excluding semantically similar words. Equally, new sentences in a passage might derive from pre-existing sentences with the addition of important content words, for example the sentence *He will **not** take the job* adding important information with the addition of the word *not*. Viewing the last example from the perspective of a language model, potentially novel content might be hidden in passages the model deems unlikely, either due to the usage of uncommon words or irregular sentence structures. Chapter 2 further introduced how a GPT-2 based language model can be utilised to calculate the likelihood of a given word sequence, i.e. the model averages the likelihoods of every word in the sequence, conditioned on the respective predecessors. GPT-2 performs well in language modelling tasks due to its extensive pretraining, thus having learned accurate word embeddings to represent any word input it receives. Furthermore and very importantly, as a transformer based language model, GPT-2 is able to model the likelihood of long word sequences, thus being able to

condition tokens on extensive contexts. We now combine both ideas for the novel content detection component of the prototype: Given a document d_n and its preceding documents $d_1, ..., d_{n-1}$, the component utilises GPT-2 to extract the novel content in document d_n with respect to the context $d_1, ..., d_{n-1}, d_n$. As we are interest in novel passages and sentences within given documents, we assign **novelty scores** to each sentence in a document, allowing to compare individual sentences within a document and across different documents with respect to their novelty.

Let the document $d_n = [s_1^n, ..., s_m^n]$ consist of $m > 0$ sentences, each sentence s_i^n a sequence of words $(w_1^{s^n}, ..., w_k^{s^n})$, $k > 1$. We first define the novelty score $N(s_i^n)$ of sentence s_i^n as its inverse probability assigned by the GPT-2 language model, conditioned on the context H:

$$N(s_i^n) := \frac{1}{P_H(s_i^n)}$$

As the **preceding context** H, we consider all documents preceding the current document plus the preceding sentences of the document, thus all sentences in the corpus leading up to the sentence of interest. Intuitively, if the current sentence of interest consists of

- Keywords and phrases that appear often in the given context or
- Keywords and phrases that are similar in their meaning compared to many words in the context, thus sharing similar word embeddings,

GPT-2 will assign a higher probability to the sentence, as the content has been encountered already with respect to the context, decreasing the novelty score. Conversely, the probability of the sentence will decrease as the sentence carries words and phrases *new* to the model with respect to the context, thus increasing the novelty score. To summarise, the novelty score corresponds directly with the model's assessment of how likely a sequence is, given the preceding content in the text corpus.

To calculate $P_H(s_i^n)$, we consider both s_i^n and H as sequences of word tokens: As H denotes the textual context preceding the current sentence of interest in the text corpus, we consider $H' = (w_1, ..., w_T)$ to be a concatenated list of the words of all preceding documents $d_1, ..., d_{n-1}$, each document a concatenated list of its word tokens. We then utilise GPT-2 to condition each word $w_t^{s_n}$ of the sentence s_i^n on the preceding context $H' = (w_1, ..., w_T)$ plus the preceding the tokens in the sentence, resulting in $H = H' + (w_1^{s_n}, ..., w_{t-1}^{s_n})$.

We can then model the probability of a given sentence s_i^n conditioned on H as the joint probability $P(w_1, ..., w_{t-1}^{s_n})$:

$$P_H(s_i^n) = P(w_1, ..., w_{t-1}^{s_n}) = \prod_{t=1}^{k} P(w_t^{s_n} | w_1, ..., w_T, w_1^{s_n}, ..., w_{t-1}^{s_n})$$

Chapter 2 discusses how and why the calculation of such joint probabilities is a hard task. Thus, rather than computing the probability directly to then obtain the novelty score, we can model the novelty directly with the help of the **loss** of the model. We utilise GPT-2 to compute the novelty score as the average negative log-likelihood of the tokens of the sentence, conditioned on H. We use the average log-likelihood as it corresponds with the language modelling loss output of GPT-2 when tasked with predicting the given sentence. That is, GPT-2 takes in as input the entire sequence $H + s_i^n$, subsequently conditions each word token of s_i^n on the given context and predicts the

likelihood of the token following the preceding sequence. The model produces the log-likelihood $P(w_t^{s_n}|w_1, ..., w_T, w_{<t}^{s_n})$ of each token according to the cross entropy loss, introduced in chapter 2. Summing up the individual token likelihoods and normalizing by the sequence length then gives the average log-likelihood of the sentence.. The novelty score $N(s_i^n)$ is thus computed as

$$N(s_i^n) = -\frac{1}{k} \cdot \sum_{t=1}^{k} log(P(w_t^{s_n} \mid w_1, ..., w_T, w_1^{s_n}, ..., w_{t-1}^{s_n}))$$

utilising the average negative log-likelihood is computationally efficient, as GPT-2 is able to compute the log-likelihoods of the entire sequence in a single forward pass. In the context of the interactivity of the overall prototype, the decrease in computation time translates directly into a better response time of the system, as the user interacts with the views utilising the novelty scores. The average negative log-likelihoods further realizes the same intuition as describing the novelty score with the inverse sentence probability: The loss of the model corresponds directly with the model's ability to predict the given sequence and the errors it produces during the process. If a sequence is novel with respect to the provided context, the resulting model loss of the sequence is likely to be higher. Further recalling the evaluation of language models in chapter 2, we can retrieve the perplexity of the given word sequence by exponentiating the inverse negative log-likelihood of the sequence. The perplexity evaluates the model's confidence in its predictions, assigning higher probabilities as the confidence grows. Thus, as the model predicts a sequence consisting of words and phrases prevalent in the given context, the model produces less mistakes, resulting in a smaller loss. This again results in a decrease in both the perplexity and the novelty score: The predicted sequence contains little new content after the model.

As the last step, we apply min-max normalization to the novelty scores to obtain scores in the range $[0, 1]$, $\forall s_i^n : \ N(s_i^n) \in [0, 1]$.

We generalize from the sentence-wise novelty scores to the novelty $N(d_i)$ of the overall document d_i by averaging over the novelty scores of all sentences $[s_1^n, ..., s_m^n]$:

$$N(d_n) = \frac{1}{m} \sum_{i=1}^{m} N(s_i^n)$$

Averaging over the sentence scores offsets small outliers within documents and aligns them with respect to the average document-wise scores. It serves as an efficient to compute approximation of the true likelihood of the document given the preceding context. As a result, this will highlight documents with text passages the model deems highly novel and documents with a high volume of novel content, as these documents are assigned a comparatively big novelty score. Alternatively, we can define the novelty score $N(d_i)$ with respect to the highest sentence-wise novelty score occurring in the document:

$$N(d_n) = \max_{1 \leq i \leq m} N(s_i^n)$$

This emphasises on particular outliers within documents, reducing the documents to their most relevant, i.e. most novel, text passage. For our purposes, we focus on the former definition, modelling the document score as the average over the sentence scores. This allows us to interpret the document score as an estimate of the average novel information provided by an article and utilise the document scores to modify the tf-idf weighting scheme, as described in section 4.3.3.3.

Assigning sentence-wise novelty scores to each document of interest, allows to retrieve novel text passages by searching for the sentences with the highest novelty score, corresponding to the least likely sentences according to the underlying language model. utilising GPT-2 as a language model over traditional statistical language models bears two important advantages: GPT-2, as a transformer network, is capable of conditioning sentences on long range contexts. As a document is preceded by many different articles, focusing on different topics, GPT-2 is capable of utilising the context provided by related articles covering similar content to the document of interest, but appearing early in the corpus compared to the document. The second advantage lies with the pretrained word embeddings used by the transformer network based language model: As GPT-2 assesses the novelty of a particular sentence, it is able to recognize words and phrases that are new with respect to occurrences in the given context, but are represented by word embeddings similar to representations the model has already seen, thus correctly decreasing the novelty score for the particular sentence. Outside the contextual information provided with the inputs, as GPT-2 is pretrained on a language model objective, the model is capable of noticing unusual sentences, for example with an irregular sentence structure or containing uncommon words. News stories, with the aim to captivate the reader, can often consist of passages formulated and structured in a way to build up and guide the reader towards the key content of the story. As sentences finish abruptly or are prolonged with decorative phrases and as rather uncommon words are utilised, the GPT-2 will mark these sequences as unlikely and thus potentially novel.

Figure 4.3 shows the novelty scoring for a mock example, emulating a small news story. Note that the scores do not match the scores provides by the GPT-2 language model. They are instead manually constructed and reflect our expectation, to illustrate the core idea discussed in the sections before. Assume the model is provided as context the section on GPT-2 in chapter 2 and subsequently scores the novelty of each of the four sentences. The first sentence provides no new information with respect to the introduction of GPT-2 and is thus assigned a low novelty score. The chapter introduces GPT-2's capability in predicting text, but not *poems*. The word *poem* shares similar contexts with *text* and should thus not be completely unexpected, yet it comes as new information and thus increases the novelty score of the second sentence. The third and fourth sentences each intentionally throw a curve ball at the model: They contain out of place words given the context, like *fishing* or *Cow*, and a sudden change in structure in the case of the unexpected *definitely not*. These odd tokens significantly reduce the likelihood of the sequences, conversely increasing the novelty scores of these sentences: The sentence contains, in the given context, new information.

We utilise the pretrained, but not yet fine tuned, medium sized English GPT-2 model with 345 million parameters for the novelty score calculation. Details on the technical aspects of the model can be found in chapter 5. While the model provides many of the advantages discussed so far, it also comes with an important limitation, preventing the desired usage as described in the passage above: The model allows to condition a particular token on a contexts of up to 1024 tokens. The calculation of the novelty score as defined above requires the model to condition a particular sentence on its entire history with respect to the corpus. Intuitively, the limitation of 1024 tokens is exceeded very fast: Popular stories covered by big news publisher like *The New York Times* average around 600 to over 1000 words [1]. Dissolving each word into a separate token would thus render the novelty score prediction for sequences of more than 2 articles infeasible.

[1] https://www.newswhip.com/2017/01/long-shared-stories-social-media/

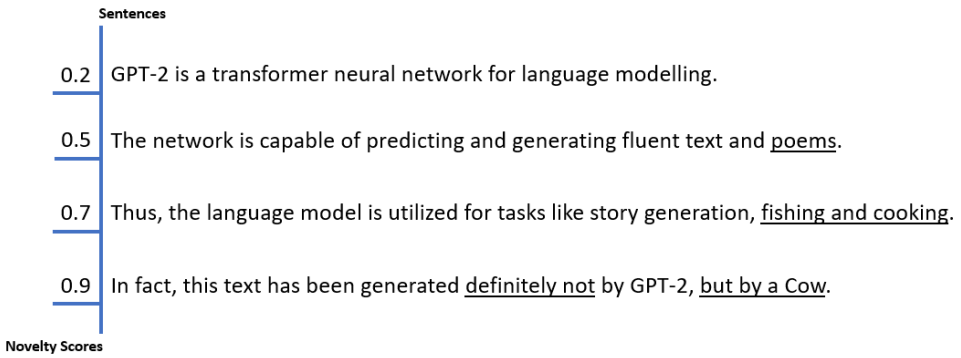


Figure 4.3: A four sentence mock example and constructed novelty scores illustrating the potential novelty scoring by GPT-2. We assume the model is given as context the introductory text on GPT-2 in chapter 2. The model subsequently scores the first and second sentences rather low, as they contain little new information. The third and fourth sentences contain unexpected and unusual words, reducing the overall likelihood and thus increasing the novelty of the sequence.

As a workaround, we employ a sliding windows approach[2]: If the context size $|H|$ exceeds a predefined maximum number of input tokens c_{max}, for example the maximum number the model can process, we condition each token on the last $c_{max} - 1$ tokens of the given context, sliding the context window by 1 position as the sentence is processed. We thus approximate the conditioning on the entire context sequence with a fixed sized, moving window, providing the model with as much context as the technical limitations allow. Using a windowed context is not strictly disadvantageous: Depending on the surrounding context, textual information encountered once again can have novel value. The shorter context induced by the windowed approach can thus be interpreted as a ”short term memory”, updating its context faintly similar to the LSTMs introduced in chapter 2.

We further employ *fine tuning* to side step the technical limitations and improve the likelihood assessments made by the GPT-2 language model. For that, we train the pretrained GPT-2 model on 10% of the given news article corpus. That is, we utilise said portion of the corpus as a labelled training set and employ supervised learning to fine tune the parameters of the pretrained model. The fine tuning of the model on a subset of the data set is motivated by two major reasons:

- The model learns to adapts its parameters to the specific use case, that is, the data set consisting of news stories. The fine tuning ”prepares” the model with respect to the average length of news articles and the common and unusual sentence structure and vocabulary utilised in news stories. The model thus learns to find and learn from common patterns in the data.
- As the model has ”seen” a proportion of the data set, it has learned about potential sentences and text passages being produced by the corpus. The sequence samples provide the model with a first intuition of the likelihood of certain sequences, increasing the confidence of the model when predicting such sequences.

[2]https://huggingface.co/transformers/perplexity.html

Intuitively, both cases will lead to an increase in the probability assigned to text passages the model has some form of knowledge about, thus likely to contain less novel information and aptly decreasing the resulting novelty score for such text regions. In the reminder of this work, we assume the novelty scores utilised in various functions to stem from the fine tuned GPT-2 model, employing the aforementioned sliding windows approach to approximate long range contexts.

4.3.3 Utilising the Novelty Scores

The key motivation behind the numerical assessment of the novelty of text passages was to integrate the resulting scores into a visual framework, allowing the user to efficiently locate and extract the most important information from a text document.

As the corpus at hand consists of temporary ordered news articles forming a sequence, we identify two different levels to extract from and visually emphasise the novel content. Thus, the following first focuses on individual documents at a specific time, finding and visually highlighting text passages of interest with respect to the sentence-wise novelty scores. The text highlighting is employed within a dedicated *article view*, presenting the textual content of individual articles. In preparation of transforming news articles into word vectors, a modified tf-idf weighting mechanism incorporating novelty scores is introduced, with the aim to emphasise on and potentially restrict to novel keywords within the corresponding articles. Moving up from the level of individual documents to sequences of documents, we propose *thumbnails* as a space-efficient visual metaphor to summarise and quickly compare novel content in individual documents. utilising the novelty scores to extract the most important passages within a document, we further aggregate documents with respect to search queries and filter settings and provide the *summary view* as a search page themed overview of the relevant documents retrieved.

4.3.3.1 Conceptual realization and implementation in the framework

As the following sections discuss different visualisations, functions and views of the prototypical framework, we want to quickly address the difference between *abstract* or *conceptual* realization and the actual *implementation* in the framework.

We denote a visualisation or a view as an *abstract* or *conceptual* realization, when proposing the conceptual design of such a *potential* components of the resulting framework. In such a context, we discuss the elemental components the corresponding view or visualisation could consist of. We then argue about how these components contribute to solve the task at hand and explore potential shortcomings. Most of this chapter is concerned with this line of thinking, analysing the conceptual realization of different problem solving ideas.

Conversely, we designate the views and visualisations implemented into the final interactive visualisation framework as the actual *realization* or *implementation* of the aforementioned conceptual designs and components. Section 4.3.6 is mainly dedicated for this purpose: Presenting the visualisations and views discussed throughout the preceding chapter in the context of the finalized framework and providing functional details omitted in the presentation of the concepts.

4.3.3.2 Highlighting novel text passages in the *Article View*

Inspecting individual news articles, potentially among a sequence of articles to be explored consecutively, can be tedious without visual guidance. To accelerate the analysis of articles and to prevent overlooking potentially import content, we identify and visually emphasise novel text passages with respect to their novelty scores. We employ text highlighting by mapping the novelty score of each sentence in the article to the visualisation of its background: Both the hue of the color and the opacity of the background increase and decrease proportionally to the novelty score of the underlying sentence. We normalize the novelty scores between 0 and 1, before computing the percentage opacity as $op_{s_i^n} = min(N(s_i^n) + 0.1,\ 1) \cdot 100$, adding 0.1 to the intermediate score such that sentences with low scores are still assigned a minimum opacity of 10%. Employing a continuous sequential color scale, we then define the background color with respect to the novelty score. The color scheme is a function that takes in a numeric value from a continuous input domain, in this case the normalized novelty scores, and subsequently maps the input to a continuous output range, from which a color representation is interpolated. The usage of a color scale guarantees that each sentence with a distinct novelty score is assigned a different background color. We utilise a sequential scale, such that an increase in the input novelty score corresponds with an increase in the hue of the mapped background color.

Figure 4.4 depicts the conceptual *article view*, which we later integrate into the prototype as a dedicated view displaying the contents of selected news article. The figure shows again the four sentence example introduced in section 4.3.2. The figure presents the novel content in the manner described so far, with the corresponding novelty scores on the left side. Note, that both the novelty scores and the background coloring and opacity do not reflect the actual outputs of GPT-2 or the color scale, but are constructed for demonstration purposes. The hue and opacity of the background of the first two sentences correspond tot their low novelty scores, thus letting the sentences appear muted. The high novelty scores of the third and fourth sentences are reflected in the increased background visibility, drawing the attention towards these sentences.

The novelty score dependant opacity and background color of sentences ensures that each sentence is distinguishable from its textual neighbourhood, if the novelty of the respective contents differ. As opacity and hue scale with the novelty score, sentences containing higher volumes of novel content are more prominently visible, attracting the attention compared to sentences with little new content. Conversely, sentences falling into the latter category will, to some extend, blend into the canvas, further extending the visual discrepancy compared to standout sentences with higher novelty scores. While the differences in color hue and opacity are not as pronounced in figure 4.4, the figure illustrates a potential shortcoming of the background visualisation described so far: As sentences differ in their assigned novelty score, the background visualisation differs as well, resulting in abrupt transitions. On one hand, this further visually distinguishes novel text regions from less novel regions. On the other hand, it disrupts the visual flow, potentially interfering with the reader's attention as sudden and rigorous transitions in the background visualisation distract from the text.

We thus refine the background visualisation by Gaussian smoothing, introduced in chapter 2, to further smooth the coloring. The smoothing is employed by applying a 1-dimensional Gaussian filter to the input novelty scores, with a chosen window size ω. Given a sequence of scores $[N_i, ...]$, the scores of a passage of consecutive sentences for example, the filter is shifted through the sequence. The filter subsequently averages every ω consecutive scores according to the filter. The resulting

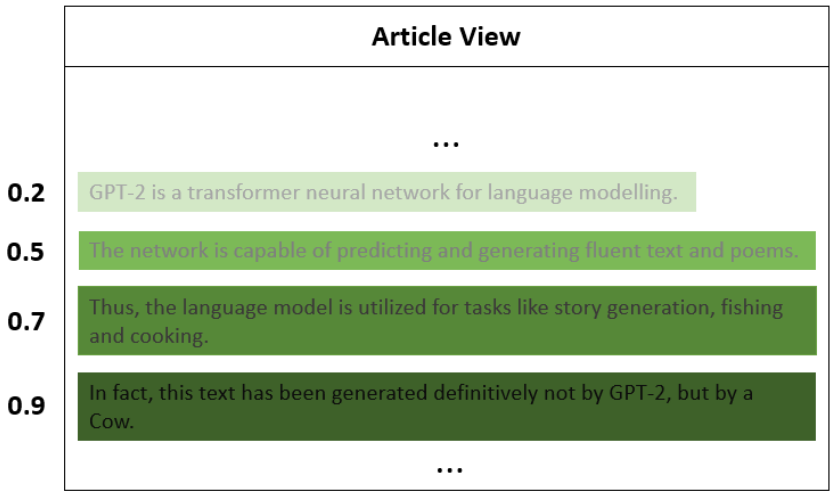


Figure 4.4: The four sentence mock example and constructed novelty scores depicting the novelty score dependant background coloring. Similar to the scores, the resulting colors are manually defined for illustration purposes. The background visualisation of each sentence reflects its novelty, being more visible and drawing more attention as the novelty scores increase.

scores are then mapped to background colors using the color scheme as described before. The smoothing process transforms the scores of the input sequence, calculating weighted averages, with an emphasise on the values at the center of the filter window. Assume as input a list of sentence novelty scores s_i^n of a given document d_n. Applying the Gaussian blur on the sequence smooths out the sentence scores, aligning the scores around the higher value components, while scaling down the outliers. Mapping the resulting smoothed scores to background visualisation, we can observe a reduced discrepancy between color hue and opacity resulting from low novelty scores and high novelty scores respectively. The Gaussian blur thus provides a way to smooth out the transitions between text sequences of varying content novelty. Furthermore, the blurring extends the visual emphasis of highly novel regions to their textual surroundings: The alignment induced by the blurring increases the scores of sequences with a low novelty score, but bordering on passages containing novel content. The increased color hue and opacity stemming from these scores enhance the visibility of the corresponding sequences, hinting towards novel content as the reader approaches the text passage.

Figure 4.5 shows side by side the difference the Gaussian smoothing makes for the visualisation, compared to the direct mapping of the scores. The Gaussian smoothed scores have been manually constructed by applying a 1-dimensional Gaussian filter on the original novelty scores seen on the left of the figure, with a window size of 4. Compared to the visually more distinct background visualisations in the direct mapping on the left, the Gaussian smoothed scores produce a more subtle transition between the backgrounds of each sentence. This slightly reduces the stark visual difference between text regions of low and high novelty respectively as seen in the direct mapping. At the same time, it provides a visually more fluent alternative, emphasising a bit more on the surrounding of the highly novel sentences. Providing the user with both options in the implementation of the approach, we allow to customize the appearance of the text visualisation depending on the requirements

discussed so far. For the concrete window size in the developed prototype, we chose a small multiple of the standard deviation of the given sequence as the default window size. As the window size directly affects how much the values of the sequence are aligned, the standard deviation can be interpreted as an indicator of how much smoothing is required. Alternatively, we add the option to manually define a custom window size, allowing to experiment with different windows.

Figure 4.5: A side by side display of the *article view* and its visually emphasised content. The left view shows the direct mapping of the novelty scores to the background visualisation. The right view shows the Gaussian smoothed scores, with a window size of 4. This results in more aligned scores and thus in a subtly smoother transition between the different sentence backgrounds.

Both the background visualisation and the Gaussian blur have been discussed so far only within the scope of sentence-wise highlighting, i.e. we have viewed sentences as a single unit to visualise. The discussed techniques can be easily applied on a word level, such that the canvas around each word is visualised with respect to the word's novelty score, subsequently smoothing the background visualisation of the overall sentence by employing a Gaussian blur with a window size equaling the number of tokens in the sentence. Figure 4.6 compares the visualisation result for sentence-wise and word-wise direct mapping of the novelty scores to the background of the text.

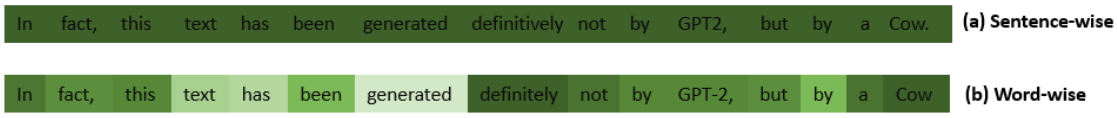


Figure 4.6: A visual comparison of sentence-wise background visualisation and word-wise visualisation of the text background. Both instances display the direct mapping of manually constructed novelty scores to the background coloring and the opacity. Compared to the sentence-wise visualisation above, the word-wise visualisation below clearly highlights the unexpected words contributing to the sentence's novelty, while the overall visualisation is not as fluent.

Comparing the visualisations shown in Figure 4.6, we can identify merits and shortcomings when highlighting the background on the sentence level and the word level respectively: On the word level, the individual highlighting of tokens according to their novelty score supports the easy identification of novel keywords in sentences. The word level visual emphasis on novelty quickly draws the attention towards important words, allowing a reader to skip over surrounding words. This can

be especially helpful if text passages within documents have to be analysed in detail, sentence by sentence, such that a reader can effectively jump from highlighted word to highlighted word. While the enhanced word level emphasis can help finding keywords quickly, if a document consists of multiple such attention drawing words, the multitude of visually highlighted points of interest might overwhelm a reader. This is further aggravated if the novelty of words oscillates within and across sentences, leading to a disrupted background visualisation as the mapped color hue and opacity changes frequently. In the prototype developed in this work, we thus employ sentence level background highlighting. Applying the discussed visualisation techniques on the sentence level removes some of the advantages gained from word level visualisation: Most notably, we lose the highlighting of individual novel keywords as we utilise a single novelty score to represent the entire sentence. If a document mainly consists of sentences the underlying language model mostly assesses as predictable, with the exception of singular, highly novel words, the averaged likelihood will likely lead to similar novelty scores and thus similarly mapped background colors. As a result, large parts of document background visualisation might look the same, failing to point the reader towards the potentially existing, highly novel keywords. Despite these shortcomings, the sentence-wise background visualisation offers great advantages in the uses cases the proposed approach is aimed at. With a large corpus of multiple documents at hand, the more comprehensive sentence-wise background visualisation provides a better visual summary of the overall document's novel content and the trends among the individual passages. This is largely attributable to the more coherent and fluent background visualisation stemming from less variance in the input novelty score data, as the sentence scores average out the individual token scores. The fluency of the background change allows to quickly discern regions of novel content from uninteresting passages, even as high value components in the inputs are smoothed out. In the end, the sentence-wise background visualisation functions well in the context of analysing large text corpora: As the user has to examine multiple text documents, the visual summary provided by the sentence-wise visualisation might be more beneficial in terms of efficiency, than the more fine grained word-wise visualisation.

Overall, the novelty score dependant background visualisation proposed in this approach guides the user in their analysis of the document, providing the user with points to focus on, as it draws the attention towards novel text passages. The novelty score dependant opacity and color hue of the text background makes passages of varying novelty easily distinguishable. Adding the Gaussian blur, the overall visualisation can be smoothed out. This reduces the visual disruption of text regions with little novel content transitioning into regions of highly novel content and vice versa, though at the cost of losing a bit of distinctiveness of different backgrounds.

We proposed the *article view* as a dedicated visual container to inspect and explore individual texts and their novel content. The user can utilise the *article view* to isolate and focus on specific articles of interest and thoroughly analyse the novelty of individual text regions. The dedicated view allows to add complementary information like the publishing date and the news provider name and URL. Going a step further, we can incorporate interactive tools to provide additional insights: Section 4.3.4 introduces the generation of alternative text sequences with the help of GPT-2, providing the user with a way to sanity check the novelty assessment of the model. Finally, section 4.3.6 showcases the integration and realization of the *article view* in the prototypical realization of the interactive visualisation framework.

4.3.3.3 Modified TF-IDF Scores for novel keywords

The section before argued and disclosed the reasoning behind the focus on sentence-level novelty and sentence-level text highlighting when displaying the contents of individual documents. Still, several situations arise where we are interest in particular keywords from articles, preferably those keywords associated with the important, novel content of the article. We can define *important* both in the sense of being meaningful and as related to the overall topic of the document: When we are interested in the novel content provided by a news article, we are often particularly interested in those text passages containing information related to certain topics of interest the article might cover. Topic related keywords act as beacons in such scenarios, as they lead towards the text passages of interest. Solely relying on word-wise novelty scores, i.e. high novelty scores, to extract such keywords can yield inconsistent and erroneous results. That is, there are several instances where a high word score does correspond to a novel word after the model, but not necessarily with an important word:

- A high word score can stem from uncommon words the model simply did not expect in the context of a particular sentence or sequence. This might occur within a passage containing relevant information, but it might also occur in isolated passages unrelated to the overall topic and the sole novelty score provides no way to discern between each case.
- A high word score might be the result of an insufficiently trained language model, such that the model assesses a word as unlikely given the context, despite the context containing many semantically similar words.
- A high word score might simply occur due to a spelling error, such that the underling language model is not able to recognize the resulting misspelled word.

To retrieve novel, topic-related keywords, we thus utilise the tf-idf score weighting technique discussed in chapter 2 in junction with the pre-computed novelty scores. The tf-idf weighting allows to transform documents into vectors, where each article of n distinct words is presented as a n-dimensional vector. Each entry of the vector corresponds with a word and the tf-idf weighting assigns each word a score denoting its relevancy with respect to both the document it stems from and the overall document corpus. The tf-idf vector representation is thus commonly utilised to group documents related to the same topic, as they are likely to share relevant, topic-related keywords. The tf-idf weight of a term t in a document d is calculated as $tf\_idf(t,d) = tf_{t,d} \cdot idf_t$ with $tf_{t,d} = \frac{c(t,d)}{\sum_{t' \in d} c(t',d)}$ the normalized term-frequency of t in d and $idf_t = log(\frac{N}{df_t})$ the inverse document frequency of t.

We now modify the tf-idf weight of each word utilising the sentence-wise and document-wise novelty scores, combining the relevance encoding of the tf-idf weights and the novelty encoding of the novelty scores. For that, we augment each of the components, i.e. the term frequency and inverse document frequency, as follows:

- The normalized term frequency count as defined before accounts for the frequency of terms in relation to the document length, thus weighting down for example frequent stop words in long documents. Note the frequency count $c(t,d)$ denoting the number of times t appears in d, where each appearance of t in d is weighted equally with 1. We now refine the frequency counts by further prioritizing the words stemming from novel sentences. That is, we weight

each individual count of a term t from a sentence s_i^d in document d with the novelty score $N(s_i^d)$. Formally, the term frequency of $t \in d$ is then computed as

$$tf^*_{t,d} = \frac{\sum_{s_i^d \in d} c(t, s_i^d) \cdot N(s_i^d)}{\sum_{t' \in d} c(t', d)}$$

where $\sum_{s_i^d \in d} c(t, s_i^d)$ individually counts the number of occurrences of t in each sentence in d and weights the count with the novelty score of the sentence, subsequently summing up all weighted counts. With the described adjustment, we discount the term frequency of terms appearing in sentences with low novelty scores. If a term mostly appears within sentences the underlying model deems as familiar given the context, the overall term frequency will be weighted down by the corresponding novelty scores. Conversely, if each occurrence of t is tied to a sentence the underlying model deems novel, i.e. $N(s_i^d) = 1$, the modified term frequency becomes the standard, normalized term frequency without the novelty discount. As such, the modified term frequency reflects both the relevancy of the term t with respect to the document and the novelty of the passages each occurrence originates from.

- The inverse document frequency encodes the relevancy of a term for a document with respect to the overall corpus, placing more weight on terms limited to a few documents. In the context of novelty, this can already indicate towards novel content since affected documents contain words encountered seldom within the corpus. Following a similar approach to the modification of term frequencies, we strengthen the emphasis on novel content by incorporating the novelty scores of each document a term appears in. For that, we simply multiply idf_t with the average over the novelty scores $N(d_m)$ of each document the term t appears in:

$$idf^*_t = log(\frac{N}{df_t}) \cdot \frac{\sum_m^N \mathbb{1}_{t \in d_m} \cdot N(d_m)}{\sum_m^N \mathbb{1}_{t \in d_m}}$$

with $\mathbb{1}_{t \in d_m}$ the characteristic function evaluating to 1 if t appears in the corresponding document and 0 if not. The resulting modified idf score of t functions as a weighted idf score, incorporating the average novelty of the documents containing the term. The modified score thus accounts for the influence of terms originating from documents utilising a vocabulary limited to these documents, yet providing little new insight in the reminder of their textual contents.

We finally combine each modified component to construct the modified tf-idf weighting of a term $t \in d$ as

$$tf\_idf^*(t, d) = tf^*_{t,d} \cdot idf^*_t$$

The resulting weight for a word t preserves the encoding of the relevancy of the word with respect to the document it belongs to as well as the corpus: The modified tf score still increases and decreases as a term is more or less prevalent within a document and the modified inverse document frequency still provides a measure of the distribution of a specific word over the documents of the corpus. With the proposed modification, the tf-idf weighting now additionally involves the influence of words originating from novel text regions in the adjustment of the term frequency and the influence of the novelty of the overall documents in the adjusted inverse document frequency. Intuitively, as

we represent each article as a vector of modified tf-idf weighted word scores, we expect to retrieve relevant, meaningful and at the same time novel words when polling for the keywords with the highest scores.

Overall, the modified tf-idf weighting scheme provides a way to retrieve relevant and at the same time novel keywords from documents. The subsequent sections demonstrate the usage of the keywords retrieved in this manner, including the summary of individual documents as a collection of the keywords and clustering documents according to their topics, based on the keywords as features.

4.3.3.4 Aggregation of novel content in document sequences

So far, we have discussed several visualisation approaches utilising the novelty scores, applied to present a focused view of singular articles. utilising and extending the same proposed techniques, we now look at the aggregation of the most important passages in a document and the subsequent presentation of such aggregations for multiple documents of interest.

We have touched upon the potential task of analysing multiple articles in succession, for example gathering the available information from a sequence of documents related to a topic of interest. In such tasks, if the amount of documents to inspect is large, a user might want to first gain an overview of the documents relevant to their task and subsequently select a subset of the documents to examine. This requires an aggregated representation of each document, summarising the most important properties of the document with respect to the search criterion, such that the user can discover promising looking subsets. In the context of finding novel text passages containing valuable new information about the topic of interest, we need an abstract view, visually compiling the novelty information about each text passage in a concise manner. Ideally, the visualisation provides enough of an overview to select and examine specific regions for further details, omitting on details not necessary for the decision.

4.3.3.5 *Thumbnails* as abstract article summaries

To construct such an abstract summary of articles, we first define the text regions to visualise as the sentences of each article. We thus utilise the sentence-wise novelty scores $N(s_i^n)$ of a document d_n to provide a visual summary of the novelty of each sentence. For that, we propose a glyph-based approach to hint at the novelty of a sentence: The sentences of an article are visualised as vertically stacked horizontal bars, each bar functioning as a glyph that represents a sentence. Comparable to the coloring of the text background discussed in previous sections, we then color the bar associated with a sentence with respect to the novelty score of that sentence. That is, we map the novelty score $N(s_i^n)$ to a color, utilising the same continuous color scale employed in section 4.3.3.2. As a result, each article is abstractly presented as a stack of vertically aligned glyphs, the coloring of each glyph indicating the novelty of the corresponding sentence. Note the omission of the textual content of the sentences for now: The glyphs do not contain any text and only function as a novelty indicator. The omission of text allows to compress the visualisation to a smaller size, as there is no fixed-sized, font-dependant space required to present the words. It thus remains to select an appropriate width and height for each bar with respect to the presentation canvas, such that individual bars can be recognized and potentially interacted with. Thus, we introduce the concept

of *thumbnails*, a card-like visual container for the glyph based visualisation of the sentences of an article. *Thumbnails* present the article glyphs in a central, window like canvas that constitutes to most of the card-like visualisation. The glyph window is optionally complemented with additional information, for example the title of the corresponding article in a small handle on top of the glyph window. Choosing a rectangular container to function as the overall *thumbnail* and placing the bordered, glyph window at the center, we approximate the appearance of a small document, such that the visualisation can be recognized as presenting an article. Figure 4.7(a) shows the conceptual realization of a *thumbnail* and its glyph based abstract visualisation of the four sentence running example of this chapter.

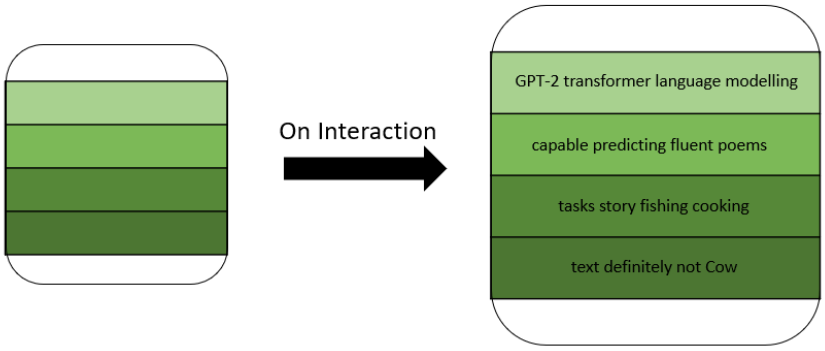


Figure 4.7: Illustration of the conceptual realization of *thumbnails* in (a) and the interactive enlargement in (b). Depicted is the four sentence running example of this chapter and their manually constructed novelty scores, the sentences interpreted as a single article. The standard form (a) displays each sentence as a bar glyph, colored according to the novelty score. On interaction, the *thumbnail* is enlarged (b), displaying a number of important keywords of each sentence, in this case 4 keywords each.

The glyph based representation of articles provides a simple aggregation view over the novelty distribution over the sentences of an article. Encoding the novelty score in the color of the bar representing a sentence, works as a clear and intuitive indication of potential novel content in a sentence. The omission of any textual contents, as we are for now only interested in an abstract overview of the distribution of novel content, results in an uncomplicated and space efficient visualisation. This allows to shrink the glyph based representation to a minimal, yet appropriate size. Placing the resulting visualisation inside a *thumbnail* produces a distinct visual unit, recognizable as an abstract representation of an article. *Thumbnails* allow to add additional information like the article title or the publishing date, while still remaining efficient in their space utilisation, as they mostly consist of the glyph based visualisation. As such, multiple *thumbnails* can be easily aligned along a vertical or horizontal row on the presentation canvas, emphasising on the sequential nature of the underling corpus. *Thumbnails* furthermore provide the opportunity to interactively change the appearance of the thumbnail and add more detail to the sentence glyphs: So far, we have ignored the textual contents of the article for the *thumbnails*, aiming to visualise each article as minimal, abstract aggregations. We can now retrospectively add the textual contents of each sentence inside the corresponding bar, though it comes at the cost of losing the flexibility in terms of sizing. For the text to be readable, the bars and the *thumbnails* would require a certain, minimal width and height. This in turn might remove the possibility to align multiple *thumbnails*, if they cumulative occupied space exceeds the available space on the canvas in either direction. As a solution, we can integrate

an interactive mechanism to show the textual contents only *on demand*: Upon interacting with a *thumbnail*, for example clicking on it or hovering over it, the *thumbnail* and the inside visualisation is temporarily enlarged. The enlargement factor can be chosen with respect to the available space and the desired font size, as subsequent to the enlargement, the textual contents of each sentence are drawn within each glyph representation. As the interaction ends, the *thumbnail* is reverted back to its original size and the textual contents removed from inside the glyphs. What content do we display inside the glyph, when the *thumbnail* is enlarged? Whole sentences, especially long sentences, are difficult to show inside the restricted space provided by the bars. Furthermore, as the *thumbnails* are intended to visually summarise the novel information in an article, displaying the entire text hinders the quick analysis of the articles. It thus seems logical, to reduce each sentence to its core content, showing only the most important keywords for example. Thus, we utilise the previously introduced modified tf-idf score to retrieve the relevant and novel keywords of each sentence of an article. After the modified tf-idf vector for an article is computed as shown in section 4.3.3.3, we sort the vector according to the scores in descending order and select the k first terms as the keywords to display. The number k can be chosen arbitrarily, depending on the available canvas space and thus depending on the desired font size for the keywords. In our case, we chose $k = 3$ as it still allowed to minimize the space required for the *thumbnails* , while $k = 5$ terms provide a decent context with respect to the content of the sentence. Figure 4.7(b) presents the transition from a standard *thumbnail* to an enlarged one, subsequently showing the 4 best scoring keywords of each sentence, functioning as short summary of its content.

The interactive change of appearance and addition of information provides a user-driven solution to selectively examine the textual contents of articles in more detail. The enlargement draws the attention towards its content, highlighting the textual information shown on demand, as well as the novelty dependant coloring of the background. By selecting the k most important keywords according to the modified tf-idf weighting scheme, we effectively summarise the contents of each sentence, providing only the necessary information with respect to relevant and novel words. As the information is provided only on demand, the user can first gain an overview of novelty of the article contents with the help of the *thumbnails*. After identifying specific articles to examine in more detail, the interactive enlargement provides a way to analyse each of these articles successively, one by one.

Overall, *thumbnails* can be utilised as complementary visualisation to abstractly summarise the contents of documents. Vertically or horizontally aligning the *thumbnails* of a sequence of articles alongside a dedicated view showing large parts or even entire articles, allows an efficient workflow where a user can first consults the *thumbnails* to filter out articles of interest and subsequently examine their partial or full content. Section 4.3.6 shows how *thumbnails* have been implemented in the interactive visualisation framework developed in this work.

4.3.3.6 Aggregate novel passages in the *Summary View*

We can pair *thumbnails* with a view showing individual articles, yet for the analysis of multiple documents in a sequence of articles, it can be beneficial to have another level of abstraction. Ideally, between the full content and the summary of individual articles, the user is additionally provided a selection of the most important text passages found within the sequence of articles. In the approach of this work, we integrate such an intermediary abstraction level in the form of the *summary view*: Given a sequence of temporary ordered articles to examine, we again utilising the

sentence-wise novelty scores to first retrieve the most important, i.e. the most novel, sentences within an article. The articles are subsequently displayed as a vertically aligned list to emphasise on both the sequential structure and the temporal ordering. Each article is predominantly represented by the n most novel sentences found in the article, determined via their novelty score. With n being arbitrarily selected, when chosen as a smaller number to filter just a few sentences per article, the *summary view* functions as an aggregation of the most novel content found in the article sequence of interest. We utilise $n = 3$ sentences in our approach, as it provides enough context to account for outlier sentences with little to no relation to the rest of the article. The retrieved sentences of every individual article provide the information to get a first impression of the content provided by the articles of the sequence. Highlighting the background of each displayed sentence with respect to the novelty score, i.e. map the novelty score to the opacity and color of the background as seen in section 4.3.3.2. This helps to quickly compare articles of a long sequence, thus allowing to narrow down and identify articles containing useful, new information without explicitly examining the entire content of each article. Figure 4.13 shows the implemented *summary view* as described so far.

As the novel passages of specific articles catch the eye of the user, such that the need arises to open and examine the full content of said articles, we can again employ the concept of information *on demand*: Changing the perspective to a full text view of a specific article might potentially harm one's mental map of the *summary view*, subsequently losing sight of other text passages of interest. We thus again couple the display of the reminder of the article content with user interaction. For that, instead of displaying at first only the n most novel sentences of an article, we surround the sentences with visual indicators hinting at the reminder of the article. The indicators represent the collapsed article content, where the visual properties of the indicators imply the content volume *hidden* between each of the n most novel sentences. We decided upon a simple, horizontal bar indicator, where the number of missing sentences is mapped to the height of the bar. The decision has an impact on the overall visualisation, as for long articles with a lot of content, a space inefficient visual indicator will disrupt the readability of the displayed novel text passages, due to occupying most of the space in between. As such, we utilise a stack of horizontal bars per hidden passage, mapping every 5 sentences to a single bar with a height of 1 pixel, each bar separated by another 1 pixel of empty space. We have noted before the average length of around 1000 words per articles, commonly encountered in popular news providers. As such, the height of the cumulative indicators scales reasonably well, with a maximum height of $2 \cdot \frac{N}{5}$ pixels for an article with N sentences. The selection of a constant factor for the height increase guarantees that the volume of hidden passages of different articles can be easily compared. Conversely, the minimum height of 1 pixel + 1 pixel for up to 5 hidden sentences allows to map the interactive expansion of the hidden text passages to the indicators: On interaction with the hidden text indicators of an article, for example a click on the respective bars, the article is expanded by displaying the corresponding hidden sentences in place of the indicators. This allows to view the hidden passages individually, as each passage can be expanded or collapsed one at the time by interacting with the corresponding indicators. The article thus temporarily grows along the vertical axis, now showing the formerly hidden sentences between the already seen novel passages. To provide the user also with the novelty of the expanded text passages, we again color the background of each sentence as described in section 4.3.3.2.

The *summary view* can be further coupled with a search mechanism, to efficiently search for articles providing novel information with respect to a search query: Given a query, a sequence of articles matching the query is retrieved and subsequently presented in the *summary view*. The user then first identifies interesting articles related to the query, utilising the summary of the overall sequence

provided by the display of the most novel text passages. As the user acquires knowledge about the novelty of most interesting passages of each article in the sequence, specific articles can then be further analysed, either focusing on individual hidden passages one at the time or viewing the entire article after expanding all hidden passages. Recalling the potential usages discussed in the introduction of the *thumbnails*, we can align the *thumbnails* of each article in the sequence either vertically or horizontally along the *summary view*. The combination of both approaches to visually summarise the contents of articles allows an efficient workflow by switching between different levels of abstraction as required. The user can easily transition from inspecting the novel keywords of an article in the *thumbnails*, to examining the novel passages found in the given article sequence in the *summary view*, to further examine specific passages for potential information not yet displayed.

Section 4.3.6 provides an overview of the integration the *summary view* into the interactive visualisation framework of this work, illustrating the arrangement of the *thumbnails* around the *summary view* and the user-steered interaction between the views.

4.3.4 Generate alternative sequences with GPT-2

With the introduction of the GPT-2 language model and the novelty scores based on GPT-2's likelihood assessment, we have thus far assumed the model to produce sensible predictions. We have argued about how to interpret and explain the novelty scores and the "correctness" of the scores given an appropriate preceding context. Yet, it would be helpful to understand how GPT-2 came to its assessment about certain regions of text. In particular, it would be interesting to compare one's expectation of what follows a certain text sequence with what GPT-2 had expected to follow said sequence, as it sheds light on the novelty score of the actually following text sequence. If a user inspects the highlighting of novel text regions in the *article view* and discovers a highlighted, yet, on closer look, seemingly uninteresting passage, it would be interesting to see the textual content GPT-2 expected at that particular place.

We thus explore this idea in more detail by utilising GPT-2's generative capabilities: For each sentence of an article, we employ GPT-2's text generation to predict the current sentence, given the preceding context. As discussed before in section 4.3.2, we define the preceding context as the article contents of the corpus preceding the current sentence and approximate this context with the sliding windows approach. The generation is employed as described in 2, generating a sentence of roughly the same length word by word. The most likely word at the currently predicted position is selected via *top-p sampling* with $p = 0.95$. We chose the *top-p sampling* over *beam search* and *top-k sampling* as it produced more coherent and less repetitive sequences. To provide a bigger sample size when comparing the expected sentences, we generate up to 5 predictions for the current sentence.

We propose to interactively present the resulting *alternative sentences* in the *article view*. That is, we enable the user to instantly view the alternative sentences of a specific sentence in the *article view* by interactively selecting the sentence, for example by clicking on it. For that, we open a dedicated view within the article view, displaying all predicted alternatives for the currently selected sentence. To emphasise on the textual overlap and similarities between the actual sentence and the generated alternative, we highlight the words shared by the sentences.

The generation and presentation of alternative sentences according to the GPT-2 language model enables to understand the model's expectation with respect to the textual content following the given context. As the expectation corresponds with the model's assessment of how likely the current sentence is, the generated sequences provide insight into GPT-2's understanding of the given context. The generated alternatives can thus be used to sanity check highlighted text passages if the user is unsure whether the assessment is reasonable. It can further be seen as a debugging tool when optimizing the hyperparameters of the test setting and evaluating the difference in results. Section 4.3.6 showcases the visual presentation of the generated alternatives in the finalized interactive visualisation framework.

4.3.5 Topic Modelling

The so far discussed *article view*, *summary view* and *thumbnails* provide the means to explore the novel, textual content in sequences of articles. As these sequences grow in length, it gets successively harder to keep track of the novel content related to specific topics prevalent in the examined articles. Even with the different forms of summaries and aggregations discussed in the previous sections, following a certain topic across multiple articles would require one to view and identify the summaries or passages containing new information about the topic of interest. This potentially becomes a difficult task when following news stories about specific topics or events: Articles usually report about many different topics, the emphasis on each topic depending on the public interest. Topics outside the "mainstream" scope with little coverage can thus be difficult to track. Additionally, the fluctuation of relevant topics with respect to the general interest makes it a challenging task to oversee the evolution of certain topics over time. The same holds for identifying newly emerging topics, which can be further aggravated if topics of high interest and traction overshadow new topics. Lastly, topics like *climate change* might last over long periods of time, emerging and disappearing from the general interest in turn and providing small amounts of new information with each emergence. Depending on the length of the time frame, these small bits of new information can be easily overlooked.

The need arises for a visual presentation of the articles, focusing on the topical evolution over time, while incorporating the information provided by the novelty assessment of the textual contents. We approach such a presentation by extending our framework with dedicated views for each task at hand, building an interactive visualisation that aims to

- provide an overview over the topics prevalent in the set of articles viewed at a specific time,
- display the topics at specific times on a timeline, allowing to analyse the development of topics over time,
- provide the interactive tools to track the continuity and discontinuity of specific topics and
- provide the interactive tools to examine specific topics and the novel content related to them in more detail

The following sections present and discuss the concepts involved in building the necessary components to realize the above specified interactive visualisations.

4.3.5.1 Topic Clustering

Suppose we define a time frame from which we retrieve all articles and subsequently want to model the evolution of topics. We start with the task of determining the prevalent topics in the given sequence of time ordered articles. Chapter 2 presented techniques for topic modelling, from which we utilise the Non-negative Matrix factorization (NMF). NMF allows us to specify a term-document matrix for the words and articles of our selected corpus and subsequently decompose said matrix into an article-topic matrix and a topic-term matrix. This intuitively helps us in the realization of the specified topic evolution visualisation: We can utilise the article-topic matrix to categorize each article as belonging to the most relevant topics according to the matrix output. The term-topic matrix helps us to identify the most representative keywords for each topic found in the given sequence of articles. To construct the term-article matrix, we need to first transform the articles into a vector form. For that, we can again utilise the modified tf-idf weighting scheme introduced in section 4.3.3.3, scoring each word of the articles with respect to relevancy and novelty. Pooling the words from all articles into a set of unique word types, we then obtain the term-document matrix by constructing row vectors for each article. Each entry of such a row vector is a score assigned to a word type, the score originating from the modified tf-idf representation of the corresponding article. Finally, we combine the row vectors into the term-article matrix and apply NMF to extract the article-topic and topic-term matrices. NMF requires the specification of the number k of expected topics to find in the given sequence of articles. We propose to algorithmically retrieve k by determining the overall similarity of the given articles: utilising the aforementioned modified tf-idf vectors, we calculate the pairwise cosine similarity δ_i between each article vector, before averaging over all pairwise similarity scores to obtain the average similarity $\delta = \frac{1}{n} \cdot \sum_i^n \delta_i,\ \delta \in [0, 1]$ for a sequence of n articles. We then set the number of topics as $k = max\ (1,\ (1 - \delta \cdot n))$. Intuitively, if the articles of the sequence belong mostly to the same or similar topics, they will likely produce similar word vectors. In turn the average similarity will be high, thus suggesting to search for a small number k of topics. Conversely, as the articles cover a diverse set of topics, the average similarity will be low and thus lead to a higher number of prevalent topics to search for. As a user-driven alternative, we enable the user to input the desired number of topics to retrieve from the article sequence.

With the k topics NMF returns, we group the given articles by assigning each article to the topic yielding the highest relevancy score according to the article-topic matrix. For each of the k topics, we further specify each topic's content by retrieving the m most relevant keywords utilising the relevancy scores of the topic-term matrix. We select $m = 10$, aiming to retrieve a diverse, but representative set of keywords for each topic. As the final result of the topic clustering procedure, we obtain

- k **topic clusters**, where each cluster is assigned a subset of the articles in the given sequence and
- a summary of the content of each of the topics, in the form of the m most relevant **topic keywords**

We now build the interactive views around the results of the topic modelling procedure, incorporating a multi-layered visualisation approach to effectively present the information from the different scales involved.

4.3.5.2 Visualise the topic evolution with the *Topic Graph*

Recalling our initial goal to present the topic evolution over time, we first need to segment our k topic clusters according to the segmentation of the given articles with respect to the time ordering. The user specifies the time frame to consider: We provide an interactive mechanism with which the user can either select the entire time period span by all articles or specifically restrict the time frame and thus the article sequence to consider. Then, we construct groups of articles for each time step in the given sequence. In our approach, we consider two granularity levels: We segment the articles within a specified time frame either day-wise or month-wise. As the following procedure is the same for either granularity, differing only in the grouping of the articles, we discuss the ensuing concepts assuming a day-wise segmentation. The articles from a specific time step are grouped according to their topic membership, leading to $1 \leq k' \leq k$ article sub-clusters for each time step. This allows us to determine and visualise the topic distribution at each time step, ideally in such way that the distribution of different time steps can be easily compared.

To visualise the topic distribution for each time step, we first need to decide on a layout. This first decision is a challenging one, as it determines how well the visualisation scales with the available canvas space. A simple stacked bar chart for example, aligned horizontally with the horizontal axis displaying the time steps, is likely to either not fit into the available space or produce a lot of clutter: As we consider time frames spanning across multiple months or even years, we might need to present the distribution of topics for multiple hundreds of time steps. We thus propose the concept of *topic graphs*, a node-link diagram of the following structure: The nodes of the topic graph are drawn as rectangular nodes, each individual rectangle corresponding with a time step and thus displaying its topic distribution. The nodes are aligned vertically, in a layer-wise placement of one node per layer, such that the *topic graph* could also be seen as a visualised list of nodes. Figure 4.8 illustrates a portion of an example topic graph, both in concept as well as in the developed prototype, which we will reference in the following passages. Figure 4.8(a) depicts both the concept of a topic node in the above portion and the resulting implementation of a topic node in the developed prototype in the bottom portion of the figure.

The topic distribution is drawn as a simple bar chart, placed inside the nodes. To construct the bar chart, each topic prevalent at a specific time step is represented by a bar. The height of the bar encodes the number of articles in the sub-cluster associated with the specific topic at the time step. The width of each bar is constant for all bars at all time steps, simplifying the visual comparison of different topic bars. The center of the node is used as a baseline to position each bar, such that the height of the bars can be easily compared. Furthermore, we employ a diverging color scheme to map each topic to a color and subsequently color the corresponding topic bars with that color. We chose a diverging color scheme as it allows us to first sort topics according to their similarity and subsequently find a mapping where the similarity of the colors correspond to the topic similarity. The color scheme is employed consistently for all time steps. Additionally, we horizontally order the bars of each individual bar chart the same way across all time steps. That is, if a topic is prevalent at two separate time steps, the corresponding topic bars will be aligned horizontally. Both the consistent coloring and the consistent positioning of the bars in the chart allow to easily compare the topic distribution of different time steps and to track specific topics prevalent at multiple time steps. Note, that the usage of a horizontally aligned bar chart has consequences for the scalability of the topic distribution visualisation: Individual bars might get very thin, if the number of topic groups to display is too high. To salvage this problem, we propose the addition of an interactive mechanism for the user, with which the number of topics per topic cluster can be manually set to

a constant, allowing to test appropriate numbers for the display of the topic bars. Figure 4.8(b) illustrates how the resulting topic nodes with the drawn topic distribution look like, both in concept and in the developed prototype.

Each adjacent node in the topic bar is connected by edges, drawn as straight lines. More specifically, the edges run between each of the topic bars of adjacent nodes. As such, the edges are utilised to encode the topic similarity between the prevalent topics at adjacent nodes: For two separate time steps and their respective subsets of prevalent topics T_1 and T_2, we calculate the pairwise cosine similarity between each topic in T_1 and T_2. For that, we utilise the previously computed topic-term matrix and represent each topic with its corresponding row vector in the matrix. Each row vector contains the relevancy scores for each topic and the article vocabulary, thus we can interpret the cosine similarity between these vectors as the closeness of the topic contents. As an alternative, we can compute the similarity directly as the number of topic keywords shared between each topic, where each topic is represented by the $m = 10$ keywords as selected in section 4.3.5.1. This provides a more interpretable numeric result, since we can specify the number of keywords to consider for the comparison, eliminating the influence of irrelevant words. The resulting similarity scores γ_i are then mapped to the width of each edge by computing the width as $w = 5 \cdot \gamma_i$ pixels, returning a 0 width, non-visible edge if the topics have strictly nothing in common. The upper bound of 5 pixels for the edge width is selected to ensure the scalability of the visualisation. Finally, we color each edge outgoing from a topic bar in the color assigned to the corresponding topic. Figure 4.8(c) completes the illustration of the topic graph by depicting the edges drawn between each of the example topic bars.

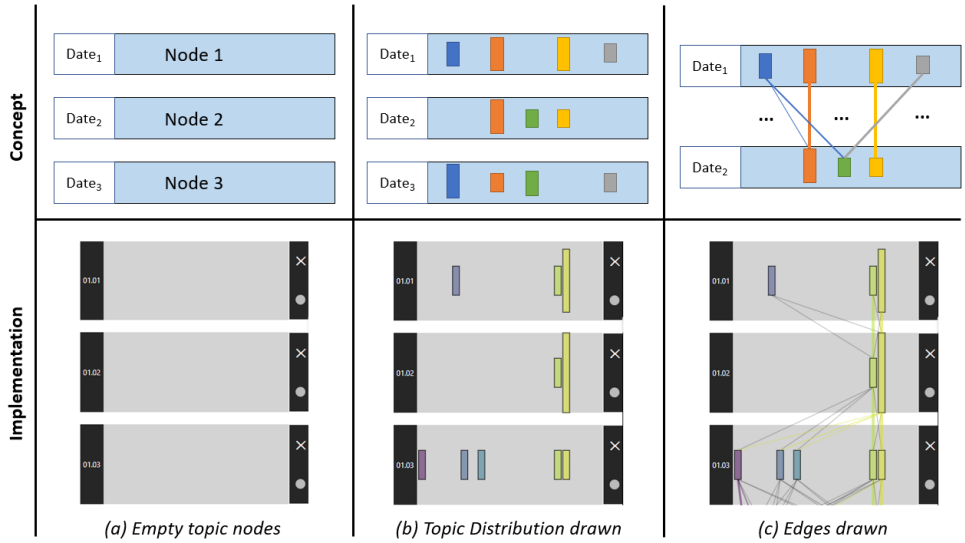


Figure 4.8: A composite illustration of the components of the *topic graph*. The figure shows both the concept behind each component in the above portion and the actual implementation in the developed prototype. Both examples display a portion of the *topic graph* for three time steps, assuming an arbitrary number of articles and topics per time step: (a) depicts an empty topic node, while (b) adds in the topic distribution per time step in each node and (c) finally displays the complete topic nodes with the similarity dependant edges between each topic bar. The topic nodes in the implementation portion display additional, interactive components, which are detailed in the upcoming sections.

The vertically aligned, layer-wise graph visualisation functions as an overview of the topic distribution over the articles at each time step in the time frame of interest. Displaying the topic distribution as a bar chart in each layer works as an intuitive presentation of the relevant topics at a specific time. The article-to-height mapping further functions as an easy to interpret encoding of the relevance of each topic at a specific time and enables to compare the relevance of a topic across multiple time steps. The consistent coloring and the consistent horizontal ordering of the topic bars across all time steps allows to easily track topics: As topics continue to be relevant over multiple time steps, the evolution of such topics can be followed along the vertical layout of the graph. It is simple to recognize if a topic continues or disappears, as both color and positioning are deterministic, allowing to easily verify the prevalence of a specific topic at a specific time step. At the same time, new topics emerging at certain time steps draw the attention, as the corresponding position will be ”empty” in all preceding topic distribution charts. The colored edges further simplify tracking particular topics, while the similarity encoding in the edge width can be utilised to identify related topics across time steps. Mapping the presentation of the topics of a time step to the nodes of the graph, allows to interactively filter entire time steps: We add a simple button mechanism to each node, such that the entire node can be collapsed, temporarily removing the node from the graph. To prevent the disruption of the visual flow, we ”pretend” as if the time step does not exist while it remains collapsed, thus connecting the topic bars of the previous and the subsequent time step with edges as described before. Figure 4.9 depicts the procedure of collapsing and re-activating a graph node in the previously shown *topic graph* implementation in the developed prototype.

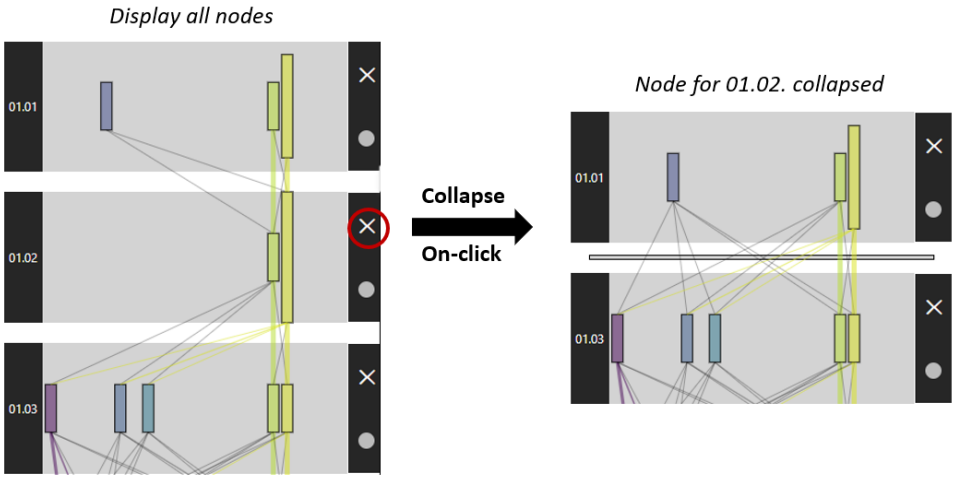


Figure 4.9: A topic node of the *topic graph* in the developed prototype is collapsed by clicking on the corresponding button. After the collapse, the respective node is minimized, while new, temporary connections are drawn between the topic bars of the adjacent nodes.

Note the omission of text labels for the topic bars: To ensure that the bar charts scale well with increasing numbers of prevalent topics at each time step, we do not label the bars directly. Instead, we propose to visualise the topics found in the article sequence as a list, accompanying the topic graph on either side: We select the most relevant keyword related to a topic as the representation or the *name* of the topic and subsequently depict the topic names in a vertically aligned list. To visually connect the topic names and the topic distribution charts presented in the *topic graph*, we color the background of each list entry according to the color assigned to the topic. The ulterior motive, besides the scalability of the bar charts, is to incorporate another interactive filter mechanism with

the list of topics. While the visualisation of the topics already greatly enables the traction of specific topics, said task gets increasingly challenging as the number of time steps and thus the number of nodes in the graph grows. Thus, we add the function to filter specific topics: Upon interacting with a topic in the topic list, for example clicking the corresponding entry, all topic bars and their outgoing edges, as well as topic list entries, not corresponding to the selected topic are greyed out. Only the topic bars and their outgoing edges associated with the selected topic remain colored. This eases the identification of the topic bars associated with the selected topic and allows to examine the similarities of the associated topics with the topics of the adjacent cluster in isolation. Figure 4.10 showcases the topic list and one example of interactively filtering a specific topic for a portion of the *topic graph* implementation in the developed prototype.

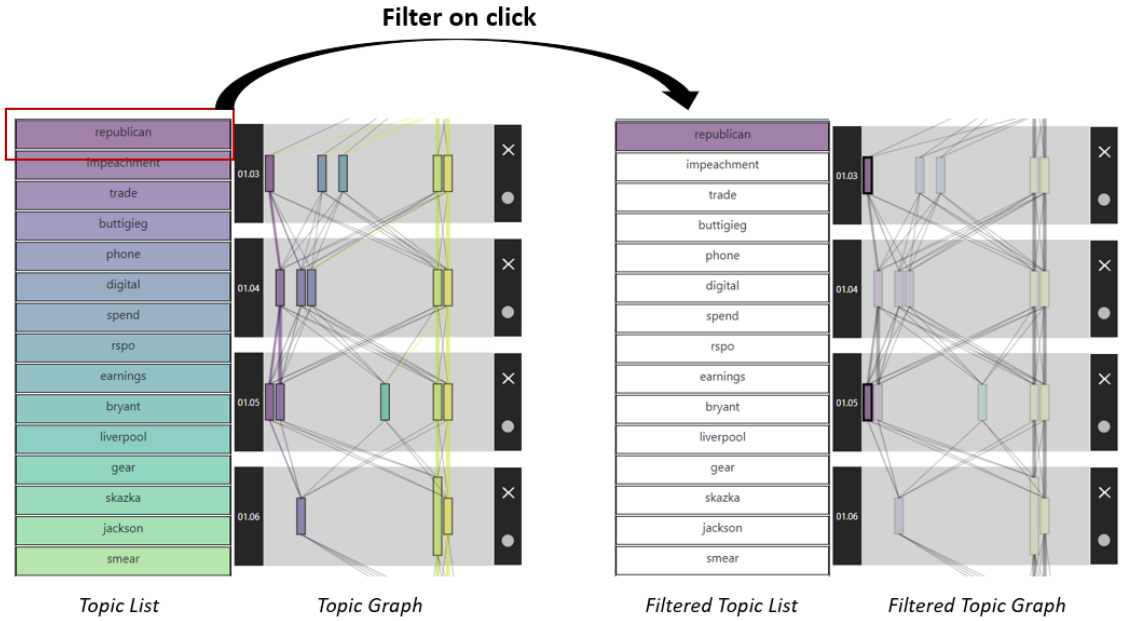


Figure 4.10: An illustration of the topic list accompanying the *topic graph*, as found in the developed prototype. Upon interaction with a specific topic entry in the list, the remaining topics get inactive. As a result, only topic bars associated with the activated topic remain highlighted, while the reminder is greyed out.

In a similar vain, adding to the capability of the prototype to reduce visual clutter, we integrate a simple edge filtering mechanism. Certain issues arise when visualising all pair-wise similarities between topic clusters of adjacent time steps: As the number of prevalent topics increases, more and more edges overlap, making it difficult to visually follow specific edges between topic clusters. While one way to solve the issue would be to employ a more sophisticated drawing technique to ”route” the edges with as little overlaps as possible, it is not the focus of this thesis. Thus, we allow the user to interactively filter edges by

- temporarily completely remove all edges to just focus on the topic bars or
- temporarily remove all edges associated with a similarity score smaller than a user-specified filter value t, which allows to derive the degree of similarity of adjacent topic clusters from the density of edges drawn for different filter values.

In Figure 4.11, we see both of the aforementioned cases for filtering edges to reduce the visual clutter induced by the overlap of edges, where 4.11(b) showcases the removal of all edges. 4.11(c) filters out all edges with similarity scores smaller than an arbitrarily chosen threshold t.

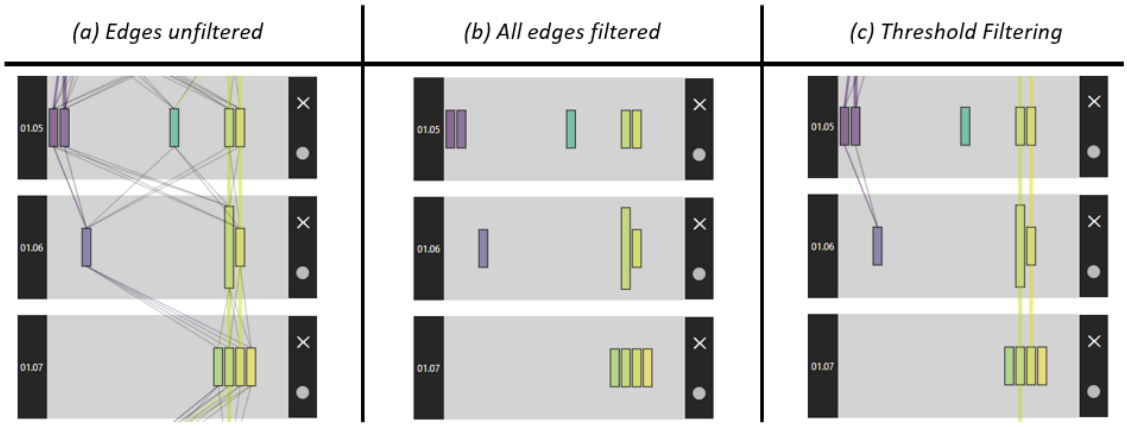


Figure 4.11: Depiction of the edge filter mechanisms of the developed prototype to reduce edge overdraw and clutter. The displayed portion of the *topic graph* first draws all edges in (a), before removing all edges in (b). In (c) the majority of edges are removed as a threshold t for the minimum topic similarity score is defined, highlighting similar topic groups.

The topic list proves a major advantage of the vertical layout of the *topic graph*: We can utilise the remaining horizontal space on the presentation canvas to accompany the *topic graph* with additional visualisations, as we will see in the following sections. The vertical layout can be further coupled with a scroll mechanism to increase the scalability of the visualisation. This provides a crucial advantage, as it allows to fixate the height of the graph nodes, ensuring that the topic bars and edges remain ”visible” as the number of nodes in the *topic graph* grows. Finally, we consider the case where a user is interested in the content of specific topics. As denoted before, we do not label the topic bars and instead provide a list of topic names. Of course, this does not provide any meaningful context about the contents of specific topics. To interactively provide a short summary of topics as a first impression of the topic content, we depict the full list of the m topic keywords we retrieved from the topic-term matrix for each topic. Displaying the keywords in the topic nodes at all times leads to visual clutter, such that we draw the keywords only on interaction with a topic bar, for example by clicking on the bar. This enables the user to inspect and focus on the keywords on demand, while reducing the information overload in the topic graph.

Overall, the *topic graph* provides an interactive, visual solution to gain an overview over the prevalent topics found within a sequence of time ordered articles. The vertical graph approach allows for the analysis of long sequences spanning over large time periods, as the interactive layout provides a scalable environment to visualise the topic distribution at each individual time step. Coupling the vertical graph with interactive filter and focus mechanisms improves the scalability, while it enables the user to selectively track and examine the evolution of topics. Section 4.3.6 provides a look into the realization of the *topic graph* and its interactive mechanisms in the interactive visualisation framework.

We now go a step further and want to consider the exploration of the topics and articles at specific times, i.e. wanting to inspect the corresponding sub-cluster of articles in more detail. For that, we leave the environment of the *topic graph*: Adding more visual objects presenting more information and detail will eventually lead to visual clutter and overload the overall visualisation. Thus, we utilise the available canvas space surrounding the topic graph to incorporate another view, dedicated to the detailed presentation of a selected time step. To provide an intuitive transition from the topic graph to the newly emerging view, we connect each of the views interactively. That is, we place a selection mechanism, for example a button, on each of the graph nodes corresponding to a time step. The selection mechanism adds the capability to focus on the articles of the specific time frame and their topic distribution. Thus, upon selecting the corresponding node, the *cluster view* discussed in the following section expands on the details with respect to the article sub-cluster and its topic distribution.

4.3.5.3 Explore topics of sub-clusters in the *Cluster View*

Having determined a specific time step and its corresponding sub-cluster to take a closer look at, the following questions might arise and motivate the closer inspection: What are the topics, prevalent in the article cluster of this time step, exactly about? What are their characteristic features, with respect to their textual contents? How does the article distribution look like? The topic bars in the *topic graph* provide only a visual indication of how many articles are assigned per topic, so what is the actual number of articles assigned per topic? And how do the articles belonging to different topics compare or differ? To answer these questions, we complement the prototypical realization with the *cluster view*. The aim of the *cluster view* is to summarise the topic related contents found at the time step of interest. For that, we construct a simple view consisting of the following components:

- A small overview presenting the overall number of articles at the time step and the number of topics found in these articles
- A short description of each topic, presented in a way that enables to compare the topics
- A small visualisation showing the article distribution per topic

Figure 4.15 presents the realization of the *cluster view* for an arbitrarily selected time step, incorporating each of the aforementioned components. The topic summary field simply corresponds with the first point of the enumeration. We additionally highlight the core of each topic by emphasising on the most relevant topic related keyword, after the topic-term matrix retrieved from the topic modelling procedure. A keyword field solves the second task of the enumeration, summarising each topic as the list of m keywords yielding the highest relevance score for the respective topic. Each list is presented side by side, enabling to compare the contents of each topic and identify possible overlaps. The background of each list is colored according to the topic colors assigned in the context of the *topic graph*, i.e. we utilise the same color scheme and mapping. The *cluster view* is completed by an article distribution field, displaying the number of articles per topic in a simple bar chart, thus carrying over the same intuition from the topic distribution visualisation in the *topic graph*.

The *cluster view* thus gives a simple and space-efficient overview of the topics found in the article sub-cluster of a specific time step. The space-efficiency of the presentation allows the concatenation of multiple *cluster views*: The selection of multiple topic nodes in the *topic graph* provides the

means to examine arbitrary time frames consisting of multiple, possibly non-adjacent time steps. We arrange the *cluster views* resulting from the selected time steps in a vertically aligned list and position the list of *cluster views* alongside the *topic graph*. Adding the interactive mechanism to remove the *cluster view* of a currently selected and expanded time step, for example via a button on the topic nodes, a user is able to coordinate the exploration of article sequences on multiple time scales and in multiple degrees of detail. Section 4.3.6 provides an overview of how the final *cluster view* is integrated into the interactive visualisation framework.

4.3.5.4 Explore related articles and novel topic keywords in the *Keyword View*

With the *topic graph* and the composition of *cluster views*, we provide the user with a high level overview of the topic evolution and a focused, intermediary level overview of the topics within specific time periods. We complete the interactive decomposition possibilities provided by the prototypical realization with a final, low level view: Suppose a user wants to further expand on a selected time frame presented in a *cluster view*, aiming to inspect and compare the contents of the articles belonging to the prevalent topics at that time step. In particular, might the user be interested in the similarities and dissimilarities of the key article contents related to each topic. For that purpose, we utilise the remaining presentation canvas to extend a selected *cluster view* in the *keyword view*. The intend of the *keyword view* is in parts similar to that of the topic evolution overview: To examine and compare the contents of articles of the selected sub-cluster, it can be helpful to physically group the contents of each article and present them in one, unified view. As before, do we need to consider the restriction set by the available canvas space and the prevention of information overload, disqualifying the option of displaying the entire article. Thus, we utilise again the modified tf-idf score vectors of each article to construct summaries of the article content: Displaying the l highest scoring keywords of each article, we provide an overview of the key content of each article. As the topics have been determined on the basis of the modified tf-idf vectors of the underlying article corpus, are the most important keywords of the articles likely to correspond to the topic the article belongs to. The selection of l depends again on the degree of context we want to provide with each article summary. We automatically set $k < l \leq 30$, with k the number of keywords chosen for each topic. Thus, we extend beyond the context given by the topic keywords and allow to compare articles of different topics in more detail. Alternatively, the user can define the number of keywords to show for each article, allowing for flexibility and the ability to test multiple values. Figure 4.15 illustrates how we use the subset of keywords returned from the article vectors to construct the *keyword view*.

We display the retrieved keywords as vertical lists, aligned horizontally according to the temporary ordering of the underlying articles. This allows to analyse the progression and development of shared and unique keywords across articles of different topics. As seen in the figure, do the displayed keywords vary in their font size: We further count the number of times a keyword appears within an article and map the number linearly to the font size, encoding the relevancy of a keyword for the article it belongs to. To visualise the emergence of new words, we additionally count for each word t the document frequency df_t and map df_t inversely linear to the opacity of the displayed keyword. Thus, keywords appearing across multiple articles within the specified time frame will be less visible, while new and limited keywords are easily distinguishable by their full opacity.

In the *keyword view*, the list-wise presentation of the article keywords enables to track the evolution of the keyword contents over the selected time span. We can easily identify keywords common among multiple articles of the same or different topics. Adding the functionality to interactively dismiss the time ordering and group articles according to their topic membership, we add the ability to compare the common and unique keywords uniting and separating articles of the same topic. The term frequency dependant font size additionally enables to analyse how the relevancy of recurring keywords changes over time and for articles of different topics. Furthermore, we are able to identify emerging and disappearing keywords with the help of the visibility of the keywords, as keywords newly emerging in a specific article immediately draw the attention. To refine the tracking capabilities, we add both an interactive filter and a search mechanism, allowing the user to either

- display only the recurring keywords shared by at least two or more articles or
- display only the unique keywords appearing in exactly one article or
- display only the occurrences of a specific keyword of interest

Finally, the situation might arise where a user develops interest in the article a specific keyword originates from, for example to inspect the text regions the keyword appears in. We want to provide the means to have a quick look into the article, without augmenting the current configuration, i.e. the extended node in the *topic graph* and the selected cluster to examine in the *cluster view*. Thus, we add the functionality to interactively open a space-efficient view displaying the article in place: Upon interacting with a keyword of interest, for example clicking on the keyword, we shrink the *keyword view* in one dimension and open a small view presenting the corresponding article. Figure 4.16 illustrates the idea behind the flow of actions:

The newly opened view functions as a miniature *article view*, displaying only the text passages containing the keyword of interest. Akin to the *summary view*, we can collapse passages with no occurrence of the keyword to save space, bringing them into view as the user interacts with the collapsed indicator. We finalize the miniature *article view* by adding a simple mechanism to switch to another article in the given article cluster where the keyword appears in, enabling to cycle through all affected articles without leaving the dedicated view.

Overall, the proposed *keyword view* completes the topic evolution analysis functionality of the prototype of this work. The *keyword view* complements the preceding *topic graph* and *cluster view* by providing a deeper look into the articles of the topics and topic clusters catching the user's eye in the higher level views. The addition of the *keyword view* allows to deploy the presentation of keyword related insights into a separate place, providing an isolated visualisation of the keyword evolution. The interactive filtering and search mechanisms enable the user to extensively track specific keywords, while the miniature *article view* contributes a way to quickly view articles of interest.

Having now seen the components making up the visualisation of the topic evolution and their abstract realization, section 4.3.6 will now showcase the actual integration of the each of the views discussed so far in the interactive visualisation framework of this work.

4.3.6 The interactive visualisation framework

In this chapter, we have so far introduced and discussed the different building blocks of a prototype realizing the interactive visualisation of novel text regions and the topic evolution in a corpus of time ordered news stories. This section brings together the individual components to form the resulting, finalized prototypical visualisation framework. For that, we present the multiple coordinated views of the framework, illustrating how each of the components and their interactive mechanisms are integrated into the visualisation tool. We round up and complete the developed prototype with the presentation of complementary and interactive functionalities not discussed so far.

4.3.6.1 Multiple Coordinated Views

The interactive visualisation framework is realized as a single-page web application, the technical details presented in more detail in chapter 5. To explore the multiple facets and insights of the given corpus of news stories, most importantly the novel content and the prevalent topics, the developed framework adopts the *Multiple Coordinated Views* approach.

Incorporating multiple distinct views, each view tasked with the presentation of one or multiple coherent facets of the data, allows for the effective usage of the available canvas space and the separation of concerns. Thus, we sectioned the framework into two major views:

- The **text view** is dedicated to the presentation of novel content in individual and sequences of articles. The text view is further subdivided into the interconnected *article view*, *summary view* and the *thumbnails*, introduced in the preceding sections.
- The **topic view** is centered around the visualisation of the topics of the corpus and their evolution. The topic view combines the previously discussed *topic graph*, *cluster view* and *keyword view* into a single, dynamic and interactive view.

The text view and the topic view are decoupled, i.e. the user is able to explore two different subsets of the same underlying text corpus in either view. This simplifies the transition from one view to another, as the user is not required to remember a mental map of the selection and state of one view when switching to the other. We complement each view with a set of interactive functions, aiming to assist the user in the exploration and analysis of the data with an interface focused on the ease of use.

4.3.6.2 The Text View

We arrange the components of the text view to support an effective workflow, narrowing down the exploration starting from a specified sequence of articles to the analysis of novel content in a single article of interest and vice versa. Figure 4.12 depicts the resulting layout of the text view.

The *date selection field* sets up the exploration of the underlying corpus, allowing the user to first specify the time frame of interest in (A)(1). We decided upon a simple input mechanism, as visualising the number of available dates becomes difficult as multiple hundreds to thousands of dates have to be considered. Upon input of a time frame, we display the number of articles published within the specified time period below the selection field. The user can then decide to immediately display all found articles by pressing the button below. Alternatively, the user can further refine

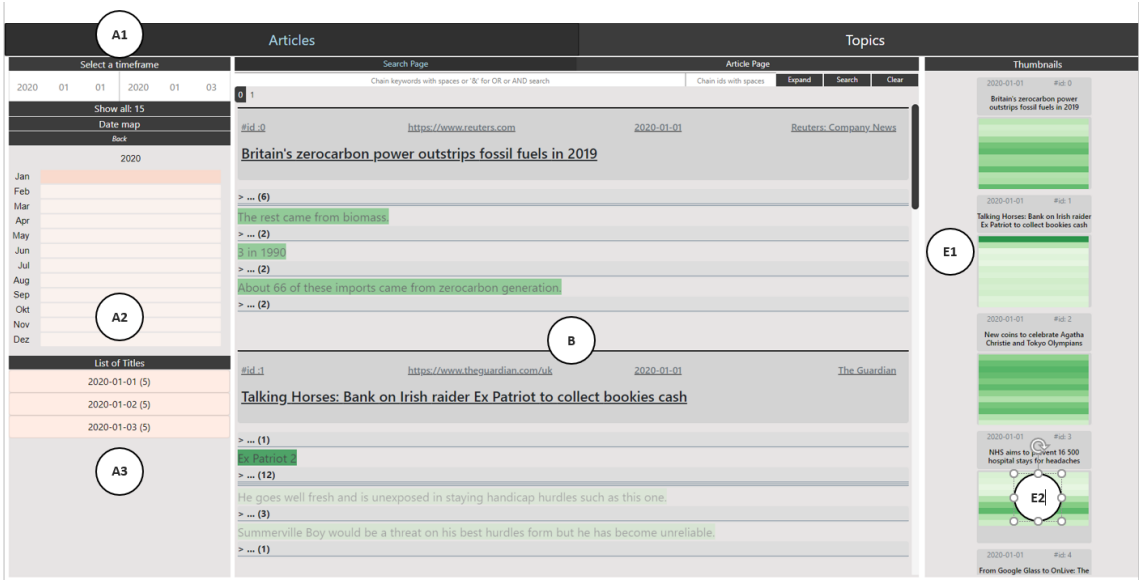


Figure 4.12: The text view of the developed prototype, presenting the visualisations associated with the *summary view*, *article view* and *thumbnails*.

the selection by restricting the time frame to certain months or days utilising the *date heatmap* in (A)(2). The *date heatmap* has two modes: We depict a month as a horizontal bar, where we map the number of articles published during this month to the color of the bar. Upon clicking on a bar associated with a certain month, we indicate the number of articles published for each day of the month via the corresponding rectangle's background color. In both cases, we utilise a continuous color scale to visually encode the gradual differences in article volumes for different dates. When a specific month or day is clicked on and thus selected, the corresponding articles from that month or day are retrieved and displayed: The *list of titles* field in (A)(3) provides a dropdown selection grouping the articles from a specific month or day. Each dropdown field denotes the specific date and the number of articles published on that date, while the background color again functions as an easy to interpret visual indication of the article volume. Upon clicking on a dropdown field, the corresponding list of titles of the articles is displayed. The background of each title is colored according to the document novelty score of the corresponding article, providing an overview over the novel content of the articles. The user can then select a title, which prompts the framework to open the corresponding article in the *article view*. Furthermore, after selecting a month or day in the *heat map*, all corresponding articles are displayed in the *summary view* in (B). The *date selection field* in (A)(1) thus allows to successively refine the set of articles to inspect, starting with a broad time period and narrow down the time frame to a specific day and a specific article from that day. The dynamic *heat map* and the aggregating *list of titles* provide a space-efficient overview of the article distribution for the dates involved, effectively encoding the volume in the background color.

Area (B) has two modes, each mode presenting a previously discussed view and depicted as isolated snippets in figure 4.13 and 4.14 respectively: The *search page view* depicts the *summary view* introduced in section 4.3.3.6, alongside a search bar (C)(1), the identifier selection bar and the expandable search menu (C)(2). As mentioned in the passage before, the article selection originating

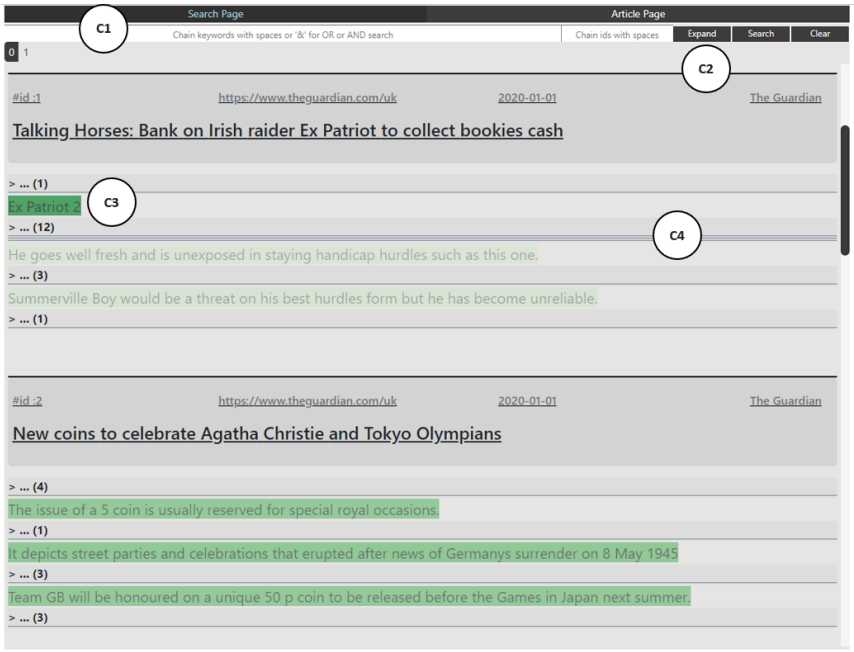
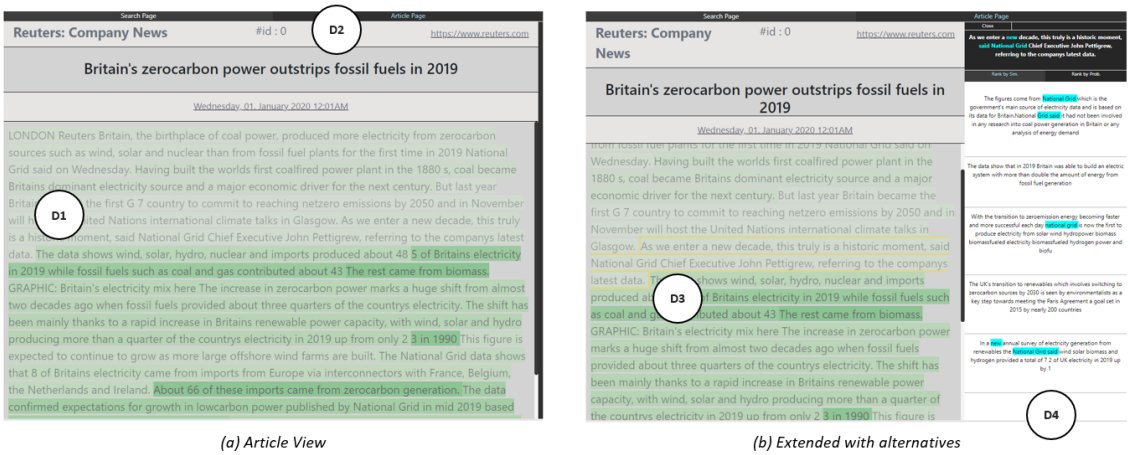


Figure 4.13: The *summary view* of the developed prototype, aggregating the most important text passages of a selected sequence of articles.

from the interaction with the *date selection field* or the *heat map* is displayed in the *summary view*. Each article is depicted as a summary of its most important passages (C)(3), with the indicators (C)(4) in between hinting at the collapsed reminder of the article.



(a) Article View *(b) Extended with alternatives*

Figure 4.14: The *article view* (a) of the developed prototype, presenting the content of a specific article and visually highlighting the novel content. Upon clicking on a specific sentence, a small view presents the alternatives generated by GPT-2 (b).

We embed each article in a bordered frame, accompanying the article with a highlighted presentation of the title, its publication date, the corresponding news provider and the URL to the article webpage. The articles are then presented as a vertically aligned list, inspired by common web search engines like Google or Bing. To handle the computational load when showing multiple hundreds of articles

at once, we paginate the *search page view*, only depicting 10 articles at once and displaying the reminder as the user interactively switches between pages. The search field allows to query articles containing specific strings or keywords. The search string can contain optional parts, separated by white spaces, and keywords, that must occur together, chained with *&*. Alternatively, the user can directly query for specific articles via their unique identifier, clearly denoted in the *summary view*, *thumbnails* and *article view* for each displayed article in some way. Multiple identifiers can be specified at once to retrieve batches of articles, allowing the user to compose a *summary view* of arbitrarily selected articles and their important passages. Instrumental to the interactive capabilities of the *search page view* is the expandable search menu (C)(2) and its search and filter options. Here, the user can refine the set of articles displayed and augment the presentation by

- changing the order in which the articles are presented. The user is able to order the articles either by their publishing date or the average novelty score of the articles, thus providing two different perspectives on the order and easing the exploration of long sequences of articles.
- filtering the articles according to the news provider. The user is shown a list of all providers that have published an article in the specified time frame. Upon selecting an entry from the list, the *search page view* depicts only articles originating from the selected provider during the specified time frame.
- filtering the articles according to their novelty score. The user can specify a minimum and a maximum novelty score, such that only the articles with an average novelty score falling within the specified range are shown.
- activating or removing the Gaussian blur. If the user is interested in comparing the visualisation results of either blurring or not blurring the sentence background visualisation or is simply not satisfied with either result, the blurring can be interactively (de-)selected.
- changing the blur window. If the current smoothing is too strong or too weak, the user can adjust and experiment with the blur window size.

The search and filter options provided by the search field and the expandable menu provide the means to interactively customize and refine the results presented by the *summary view*. The user is thus given multiple tools to narrow down the articles of the specified time frame to a specific subset of various specific properties. The *summary view* itself, as discussed before, allows the user to effectively gather an overview of a long sequence of articles.

The second mode of Area (B) is selected by switching the view to the *article view* in 4.14(a). The *article view* (D)(1), as discussed in section 4.3.3.2, depicts the novel text passages of a selected article. The selection can occur in the *list of titles*, by clicking on a title of an article in the *summary view* or clicking on a *thumbnail*. Each article is displayed alongside its identifier, its date, the news provider and the webpage URL in the header of the article (D)(2). The sentence backgrounds are visualised according to the novelty score of the sentences, where hovering over a sentences highlights it by temporarily increasing the background opacity to 100 %. Additionally, as alluded to in section 4.3.4, we provide for each sentence a set of alternative sentences: As can be seen in 4.14(b), upon clicking on a specific sentence (D)(3), a view opens up in the *article view*, displaying 5 alternative sentences generated by the GPT-2 language model (D)(4). We highlight the overlap of words between the actual sentence and the generated sentences by coloring the background of the corresponding words.

The text view is finally completed by the *thumbnails* in figure 4.12(E)(1), introduced in section 4.3.3.5. The list of *thumbnails* mirrors the selection of articles displayed in the *summary view*, remaining on the canvas even if the user changes from the *summary view* to the *article view*. This provides the user with an overview of the selected article subset at all times, even when the user is analysing individual articles in the *article view*. As discussed in section 4.3.3.5, we make efficient use of the available canvas space by only providing a visual summary of the novel content inside the card-like visualisation (E)(2).

4.3.6.3 The Topic View

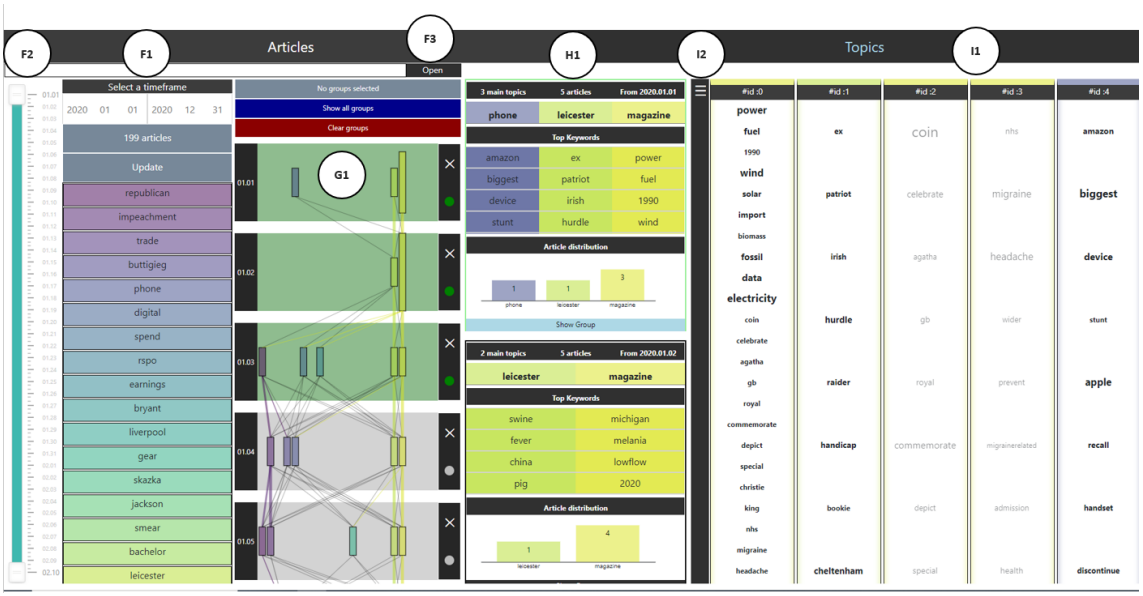


Figure 4.15: The topic view of the developed prototype, presenting the visualisations associated with the *topic graph*, *cluster view* and the *keyword view*.

The user transitions from the text view to the topic view by selecting the corresponding tab in the header of the page. Akin to the text view, the topic view again centers its layout around an efficient workflow: The user starts with a high level overview of the topic evolution in the *topic graph* and progressively descends to lower level abstractions as the *cluster view* and the *keyword view* are opened up, from left to right on the canvas. Figure 4.15 shows the entire topic view for arbitrarily selected and extended nodes in the *topic graph*.

Similar to the text view, the user is provided a *date selection field* (F)(1) to initially define a broad time frame of interest. utilising a slider mechanism in the *time frame selection slider* (F)(2), the user can then further specify the time frame. The slider mechanism was chosen to enable to user to make fine granular time frame selections. Upon changing the broad time frame in the *date selection field* and pressing the button above, the slider ranges automatically update to reflect the selected time range. As the user moves the slider handles and further adjusts the time frame, the system calculates ”on-the-fly” the number of articles falling into the current time frame and displays that number. The expandable menu in (F)(3) allows to specify the parameters of the ensuing topic evolution procedure:

- The user can specify the granularity to cluster the articles by, where the *day* selection groups topics from the same publishing day before finding the prevalent topics and the *month* selection does the same on a per-month basis.
- A variety of input parameters allow to specify restrictions on the clustering. The user can manually specify the maximum number of articles to consider for a single cluster and the number of topics to find in such a cluster. Alternatively, the number of articles per cluster can be determined dynamically, while the procedure algorithmically finds the number of prevalent topics, as described in section 4.3.5.1.

Pressing the button below the article number field then triggers the topic evolution procedure, subsequently displaying the resulting *topic graph* in (G)(1) and its accompanying topic list in (H)(2), as discussed in section 4.3.5.2. As described in the corresponding section, the *topic graph* presents for each time step the resulting topic clusters, i.e. the article groups assigned to the same topic. The topic distribution of each time-dependant cluster is shown in a topic node and the distribution is visualised as a bar chart.

With the topic graph on screen, the user can then utilise the reminder of the expandable menu (F)(3) to augment the presentation of the topic graph:

- A user defined input specifies the number of keywords to assign per topic, the resulting keyword vector representing the corresponding topic.
- A set of buttons allow to define the edge similarity, either calculating the similarity between topic clusters based on the cosine similarity of the topic vectors or based on the number of keywords shared.
- An edge filter enables to restrict the edges, drawing only edges between topic clusters if the computed similarity score exceeds the defined threshold or the minimum required number of shared keywords respectively.
- Two additional options are provided to augment the edges, where edges can be completely removed or only drawn between the most similar topic clusters.

As the graph grows in size, the user can utilise the collapse button on the topic nodes to temporarily remove the node from the visualisation. Conversely, selecting a node for expansion via the circular expansion button on the node, selects the corresponding topic cluster for the *cluster view* in (H)(1), displaying the information discussed in section 4.3.5.3. The user is able to create a custom selected group of arbitrary topic clusters from the time ordered sequence and then choose to display the selection. Alternatively, the user can decide to expand the entire sequence of article clusters, creating a list of *cluster views*. The background of an expanded node is highlighted to emphasise its expansion. In the cluster view, the user can observe the topic statistics and the main keywords representing each corresponding topic. This is followed by the topic keyword vectors presented as side by side lists. The *cluster view* of an expanded node is then completed by a bar chart, displaying the article distribution for each topic.

The final component of the topic view is drawn upon selecting a *cluster view* for expansion, prompting the system to visualise the keyword distribution of the corresponding topic cluster in the *keyword view* (I)(1). The *keyword view*, as discussed in section 4.3.5.4, presents each article of the corresponding cluster as a list of its keywords. The keywords of each list are sorted with respect to their modified tf-idf scores. Each keyword list, associated with a certain article, displays at its top

the id of the corresponding article, enabling the user to track specific articles and for example search for them in the text view. To allow the visualisation to scale for big clusters of multiple articles, the keyword lists are placed inside a *scrollable* container, displaying only a subset of the lists at a time. The expandable settings menu in (I)(2) further provides the interactive means to search and filter for certain keywords:

- The user can specifiy the number keywords to display per article, the number being set to 30 per default.
- Per default, the lists are ordered with respect to their publishing date, to simplify the traction of the evolution keywords. Alternatively, the user can group the articles according to their topic membership. Furthermore, each article list is faintly colored according to the color assigned to the corresponding topic. If the user is more interested in the novelty of the corresponding article, the keyword lists can be colored according to the articles' novelty scores.
- The user is provided multiple keyword filtering options, displaying only keywords occurring once or those occurring across multiple articles.
- Lastly, the user can search for specific keywords, prompting the system to only display keyword lists containing the keyword.

Finally, if a user is interested in the article a specific keyword originates from, clicking on the keyword opens up the miniature *article view*, illustrated in figure 4.16(J)(1). As described in section 4.3.5.4, at first are only shown the passages containing the keyword of interest, with the reminder of the article collapsed (J)(2) and coming into view as the user clicks on the collapsed passages. The header at the top of the view allows the user to cycle through all articles of the currently inspected cluster that contain the specific keyword, without having to manually search for and select the keyword in the keyword view.

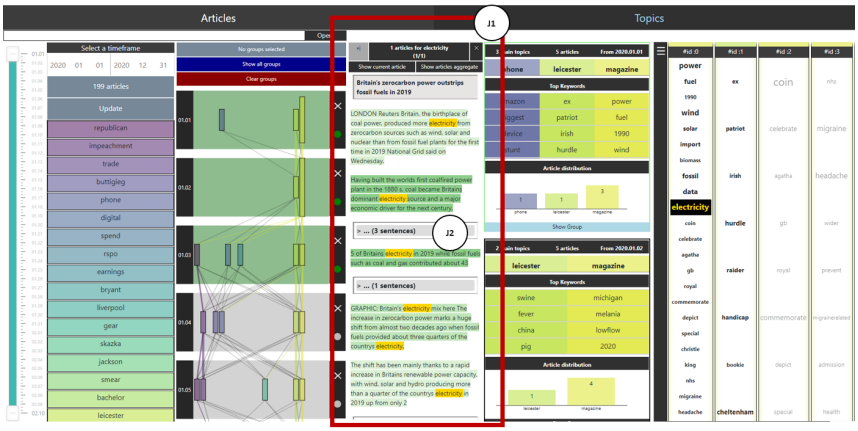


Figure 4.16: Upon clicking on a specific keyword, in this case *electricity*, a view opens up, displaying the passages of articles containing the keyword. The view behaves similar to the *summary view* and provides an interactive header to easily switch between different articles.

5 Technical & Implementation Details

The following chapter provides an overview of the technical details with respect to the implementation of the interactive visualisation framework of this thesis. We first describe the libraries utilised in the data processing pipeline, the format of the articles before and after the processing and the way articles and intermediary results are temporarily stored. Subsequently, we detail the two core technologies of the prototype in GPT-2 and the components of the topic modelling procedure. We then denote the implementation details of the construction of the different visualisations of the framework. The final section of the chapter shortly presents the general set-up of the web application.

5.1 Data Preparation & Processing

The interactive framework was developed on and works on a text corpus consisting of news story articles, each article containing a single story. The news stories where fetched from multiple online news provider sources, examples being *The New York Times* [1], *The Guardian* [2] or *CNN* [3]. The resulting data set consists of 300 000 - 400 000 JSON [4] files of the following structure:

```
{
   ...,
   ...,
   "Title" : "...",
   "Link" : "...",
   "PublishingDateString" : "...",
   "PublishingDate" : "...",
   "ArticlePageFetchedTime" : "...",
   "FetchedArticleContent" : "...",
   "FeedTitle" : "...",
   ...,
   ...
}
```

Detailed are the fields expected by the processing module of the framework and necessary for the subsequent steps of the processing pipeline. The news corpus was unordered at first and has been sorted afterwards, such that the articles are sorted according to their publishing date in ascending order.

[1] https://www.nytimes.com/
[2] https://www.theguardian.com/international
[3] https://edition.cnn.com/
[4] https://de.wikipedia.org/wiki/JavaScript_Object_Notation

5.1.1 Text Preprocessing

Having retrieved a, at first unordered, corpus of news articles, the documents need to be cleaned first. The textual contents are provided as single strings of text, retrieved directly from the corresponding web page. As such, the texts contains various forms of special, non-alphanumerical characters, for example formatting related HTML character codes like ANSI escape sequences to display certain types of punctuation characters. The data is thus cleaned, employing sentence splitting, tokenization, token cleaning and lemmatization. For that, we utilise the *Natural Language Processing Toolkit* (NLTK) [5], a language processing PYTHON library. As the final text processing step, we remove stop words from each article. As the language model still requires the context provided by stop words, we keep two copies of each article, where only one has its stop words removed. We further shorten the article URL and normalize the provider name extracted from the *FeedTitle* field. After the cleaning step, we utilise the textual contents to compute the novelty scores with GPT-2. Having computed the novelty scores, we employ the modified tf-idf procedure to compute the tf-idf vectors for each article.

Before the tool can be used, the processing of the data and the related computations have to be employed in advance. A configuration file *config.json* guides the preprocessing pipeline, as the following parameters can be specified for the ensuing processing pipeline:

```
{
   "data_key" : "",
   "num_per_day" : "",
   "num_lines" : "",
   "num_sent_keywords" : "",
   "start_date" : "",
   "end_date" : "",
   "predict_with_context": "",
   "gen_param : []
}
```

The *date_key* field specifies the data file to store the articles, while the *start_date* and the *end_date* determine the time frame to retrieve articles from. Parameters with the prefix *num_* determine the number of corresponding items to retrieve, while *gen_param* provides the parameter for the text generation. *predict_with_context* is a boolean flag that determines whether the preceding context is utilized in the loss calculation. The processing in advance is mostly motivated by the long processing times to calculate the novelty scores and produce the tf-idf vectors for each article. Employing the processing at *runtime* would decrease the response time of the system, as well as delay any interactions between the user and the system.

5.1.2 Storage

After the processing and computation steps, we store each processed article in a single dedicated JSON file, functioning as a permanent store. The structure of each article entry in the JSON file is as follows:

[5] https://www.nltk.org/

```
ID : {
   "title_cleaned" : [],
   "title_sanitized" : [],
   "content_cleaned" : [],
   "content_sanitized" : [],
   "time_stamps" : {
       "fetched_date" : "...",
       "publication_date" : "...",
    },
   "link" : "...",
   "provider" : "...",
   "tf_idf" : [],
   "novelty" : {
       "doc" : N(d),
       "sentences" : [],
    }
    "alternatives" : []
}
```

where ID denotes the article identifier and $N(d)$ denotes the document novelty score. The title and content entries are stored as arrays of sentences, each sentence again an array of tokens. The suffix *cleaned* denotes the cleaned content with stop words, utilised to display the text of articles in the different views of the framework and for the novelty score computation. The suffix *sanitized* denotes cleaned and lemmatized content without stop words, utilised for the topic modelling. The entry *tf_idf* stores the modified tf-idf vectors for each article, while the entry *novelty* stores the sentence wise and overall article novelty scores. Lastly, the *alternatives* entry stores an array of alternative sentences for each sentence of the article.

5.2 Technologies

Instrumental to the functions and visualisations are the outputs of GPT-2 and the topic modelling procedure. For each of the technologies, we quickly describe the technical details including the source of the tools, parameter and model settings and the concrete usage of the respective tools.

5.2.1 GPT2

The GPT-2 transformer based language model functions as the underlying language model of the prototypical realization of our presented approach. We utilise the OpenAI GPT-2 [6] implementation provided by the open-source artificial intelligence library *Huggingface* [WDS+20] for PYTHON. The library offers multiple pretrained models, corresponding with the different model sizes of GPT-2, from which we use the third largest model GPT2-MEDIUM, with 345 million parameters. To make use of the implementation, we additionally utilise the open source machine learning PYTHON

[6] https://huggingface.co/transformers/model_doc/gpt2.html

frameworks *tensorflow* [7] and *pytorch* [8] and the math library *numpy* [9]. We utilise the medium sized model, as our computational resources restricted our capabilities to fine tune any bigger model. We fine tuned the pretrained GPT2-MEDIUM on 10% of the article news data set.

To compute the novelty scores of the articles with GPT-2, we process each article sentence by sentence. We encode a currently processed article with the fine tuned *GPT-2 tokenizer*, providing both the sentence, as well as the preceding context as sequences of tokens. As the model can at most encode 1024 tokens at a time, we set the context window such that the model utilises a window of 1023 preceding tokens when predicting the current token in the sentence. The model subsequently calculates the loss of the sentence as the average loss of the tokens of the sentence, which we then invert and store as the novelty score of the sentence. We perform the encoding and the model inference on a GPU to significantly speed up the process.

We further utilise GPT-2's text generation function to generate the alternatives for each sentence of each article. We provide the model again with a context window adhering to the input length constraint when generating a single sentence. We choose *top-p sampling* as the token sample method, with $p = 0.95$ and $k = 20$, k denoting the number of most likely tokens to consider for the sampling. The values of p and k have been obtained through testing and observing the resulting sequences, paying attention especially to the coherence of the produced sentences and minimal repetition. We generate and store the alternative sentences at *runtime*, when the user first views the corresponding article in the *article view* and subsequently load the generated alternatives for subsequent usages from *data.json*.

5.2.2 Topic Modelling

The topic modelling procedure of our approach employs Non-Negative Matrix Factorization (NMF) to factorize a provided term-document matrix according to a predefined number of topics. We utilise the NMF implementation [10] provided by SKLEARN, a machine learning library for PYTHON.

5.2.3 Visualisation

The interactive web framework of our approach creates multiple visualisations to present the insights to be gained from the underlying data. We utilise the HTML-document styling language CSS for the novelty-dependant background visualisations in the *text view* of our framework. To create the topic graph, as well as its nodes and edges, we utilise the JAVASCRIPT library D3 [11]. To visualise the the *time frame selection slider* in the topic view of the prototype, we utilise the open source javascript library NOUISLIDER [12]. Both CSS and D3 are further utilised for minor visualisations like the *date heatmap* and for all of the styling of the single-page web application representing the prototypical framework.

[7] https://www.tensorflow.org/
[8] https://pytorch.org/
[9] https://numpy.org/
[10] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html
[11] https://d3js.org/
[12] https://refreshless.com/nouislider/

5.3 Web Application

The interactive visualisation framework of this approach is implemented as a single-page web application, i.e. we do not route the user to different HTML pages upon changing views, but re-render and redraw the current HTML page to display different visualisations. The framework is set-up with a *backend*, the server-side portion handling the data processing and intermediary result computation. The processed data and results are subsequently transferred to the *frontend*, the client-side portion rendering the web page. Frontend and backend communicate asynchronously via AJAX [13], initiated by the client-side.

5.3.1 Backend

We utilise the PYTHON programming language to implement our backend. For that, we set up a development server with the web application framework FLASK [14]. Upon starting the development server, the backend renders and serves the initial, yet empty, HTML web page to the frontend. The backend subsequently listens for client-side AJAX requests, containing parameters and values specifying the required computational results the backend then produces and sends to the frontend.

5.3.2 Frontend

For the client-side web page rendering, the user interaction handling and the styling of the HTML web pages, we utilise JAVASCRIPT in junction with CSS and the open source frontend styling toolkit BOOTSTRAP [15]. We further utilise the JAVASCRIPT library JQUERY [16] for the AJAX communication between backend and frontend. After the backend server has been started and the initial HTML web page has been served, the frontend is tasked with the majority of the page rendering. Switching between the different views of the prototype is handled in majority by the frontend, removing and rendering new HTML components of the web page at *runtime*. Data and intermediary results required for new components and views are fetched via AJAX calls to the backend.

[13] https://en.wikipedia.org/wiki/Ajax_(programming)
[14] https://flask.palletsprojects.com/en/2.0.x/
[15] https://getbootstrap.com/
[16] ://jquery.com/

6 Results

After the development of the prototype, realizing the interactive visualisation approach to novel context detection and topic evolution modelling, we dedicate this chapter to discuss the results of this work. Thus, in the first half of the chapter, we explore a set of real-world use cases and present the different ways the developed prototype can be utilisd to solve each of the problems and tasks. The latter half subsequently thoroughly discusses the utility provided by the interactive visualisation framework of this work: We highlight the advantages of the developed prototype, the disadvantages of our concrete realization and overall discuss the suitability of our framework with respect to the over aching goals we set at the beginning of this thesis.

6.1 Use Cases

Recalling the scenarios we described in chapter 1, we denoted several issues to overcome when searching for new insights in large, sequential text corpora. At the core, we can summarise and convert these issues into three big tasks to solve: Identify documents of interest, recall the information gathered so far and finally extract the new information in the documents given the context. The following sections decompose these major tasks into smaller real-world tasks and demonstrate the capabilities of the prototype developed in this work, to step by step solve each task at hand. For that, we will look at an exemplary use case in the context of examining news stories in a large corpus of news articles.

6.1.1 Task description

We assume a given corpus of news articles, published in the time frame spanning from the beginning of the year 2020 to the end of 2020. In this use case, we are particularly interested in the unfolding of the COVID-19 pandemic [1] during January 2020. For that, we prepare a subset of the articles spanning from the beginning of January, 2020.01.01, to the end of January, 2020.01.31, randomly sampling up to 100 articles containing the keyword *coronavirus* per day. Specifically, we set out the task of gathering an overview about the news progression during the month of January:

- Gather an overview about the general topic of *coronavirus*.
- Identify and examine interesting articles to gather more in-depth information.
- Find out about the general topical evolution during the time frame and the relevance of the topic *coronavirus*.

[1] https://en.wikipedia.org/wiki/COVID-19_pandemic

6.1.2 Gather a general overview of the data

Figure 6.1: The text view provides a general overview of the articles, where we observe that most articles reporting about *coronavirus* have been published at the end of the month, as the *date heatmap* shows.

We begin to solve the aforementioned tasks by examining the prepared data set in the *search page view*. Of the 891 retrieved articles related to the topic *coronavirus*, most are published at the end of the month, as the *date heatmap* and the *list of articles* in figure 6.1 show. Looking at the *list of articles* and the article novelty score dependant coloring of the titles, we can observe the novelty of the articles to fluctuate, with a slight decrease for articles at the and of the month. Figure 6.2 illustrates this by comparing articles from 2020.01.09 and 2020.01.23. We select and open the articles from 2020.01.09 as the first day with multiple reports on the topic *coronavirus*.

6.1.3 Collect the key information

As the articles from the specified day are presented in the *summary view*, we can utilis the important passages highlighted in each article to get a gist of the key information related to the *coronavirus*. Figure 6.3 shows a mashup of the novel text passages retrieved from a few selected articles. From the aggregated passages, as well as the additional information provided in the hidden passage in 6.3(b), we can quickly gather the main information provided by novel sentences as highlighted by the background coloring: It seems to be a *pneumonia outbreak* mainly affecting *China*, with the circulating virus being related to the *SARS respiratory disease*. In 6.3(a) and 6.3(b), we can further observe how two very similar sentences are reported in different articles and the underlying model correctly visualising the second occurrence as less novel. In addition, 6.3(c) shows how general statements about the *coronavirus* are old news to the underlying model.

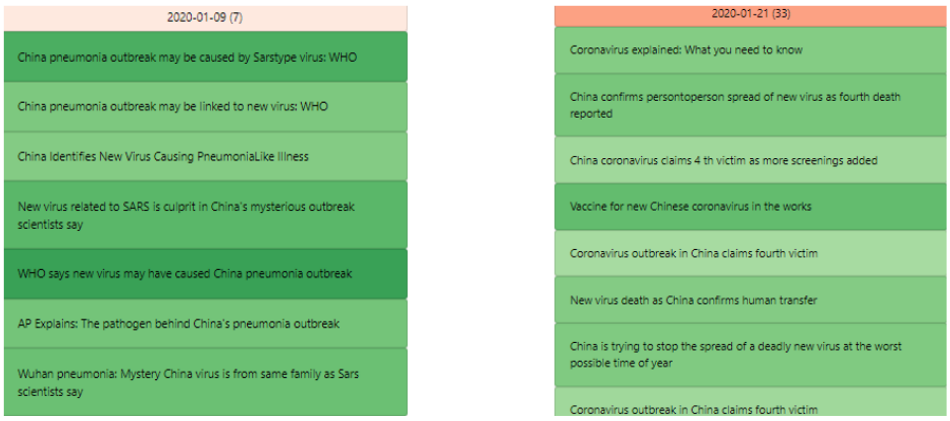
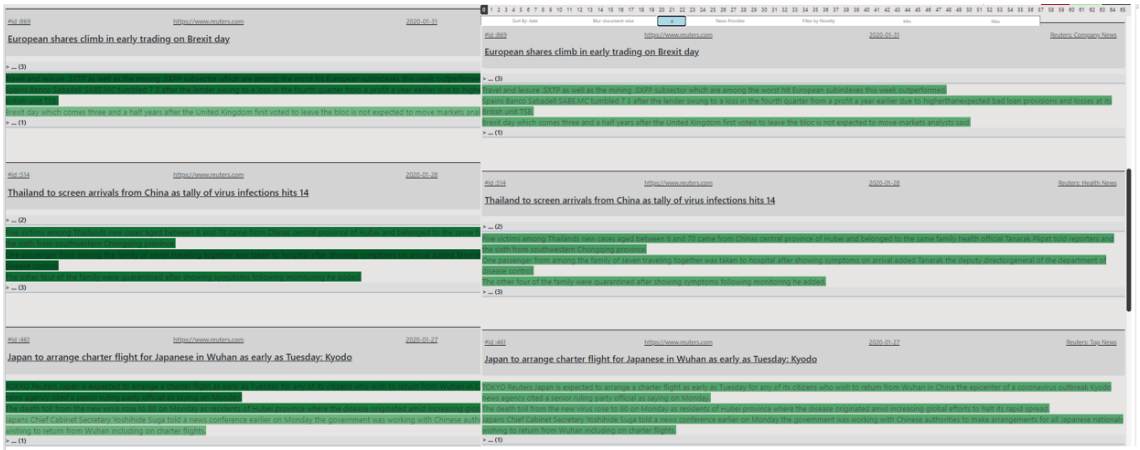


Figure 6.2: The *list of titles* shows, that the novelty of articles published at later dates, here the 01.23., seem to contain less novel content - likely as they report about content covered in the previous days.

Figure 6.3: The mashup shows some of the aggregated article passages presented in the *summary view*. The background coloring draws the attention towards the novel information, from which we can quickly gather information about the *coronavirus*. The sequences outlined in red highlight how the underlying model recognizes already known content.



(a) Sorted by novelty (b) Blur window set to 4

Figure 6.4: The overlapping illustration displays how we first sort the articles by their novelty, resulting in the presentation of highly novel text passages (a). To make the passages more readable, we adjust the Gaussian smoothing window size, resulting in a more fluent background visualisation (b).

6.1.4 identify specific articles of interest

Having roughly determined what the retrieved articles report about, we want to gather specific information about the *coronavirus* topic. For that, we want to identify promising articles to examine in detail. We return from the subset of articles retrieved from January 9, to the overall data set and start our search by sorting the articles by their novelty score, in descending order. This gives us an overview of the most interesting articles after the underlying model. As figure 6.4(a) shows, since the articles contain many highly novel passages, the background highlighting makes it difficult to read some of the passages. Thus, in Figure 6.4, we adjust the Gaussian blurring window size, resulting in more fluent background visualisations. As a side effect, it now becomes difficult to identify particularly interesting articles, as most backgrounds have similar color hue and opacity. Rather than going though each of the aggregated passages, we select a few promising looking articles and restrict the *summary view* to these article, before we consult the *thumbnails* and look out for keywords of interest. Figure 6.5 shows the aforementioned identifier search (a) and the close inspection of a thumbnail (b). Its sentence-wise keywords hint at an interesting article with keywords of different topics like *diseases*, *wildlife* and *climate*, causing us to examine the specific article in the *article view*.

6.1.5 Examine articles of interest

Viewing the selected article in the *article view*, the highlighted text passages guide as towards the more interesting and novel information with respect to the insights we have gathered so far. We can quickly identify the *transmission of diseases between animals and humans* as the main topic. As expected, since we are viewing the 786-th article in the provided data set, passages concerning general information about the *coronavirus* are not highlighted. Meanwhile, passages containing new information about *animal disease transmission* are predominantly put into focus. Looking at

(a) Select articles via their ID (b) *Thumbnail* Keywords

Figure 6.5: To restrict the number of articles to analyse, we cherry pick a small sequence of articles via their identifier and inspect their *thumbnails* (a). The keywords on demand draw interest towards a specific article (b).

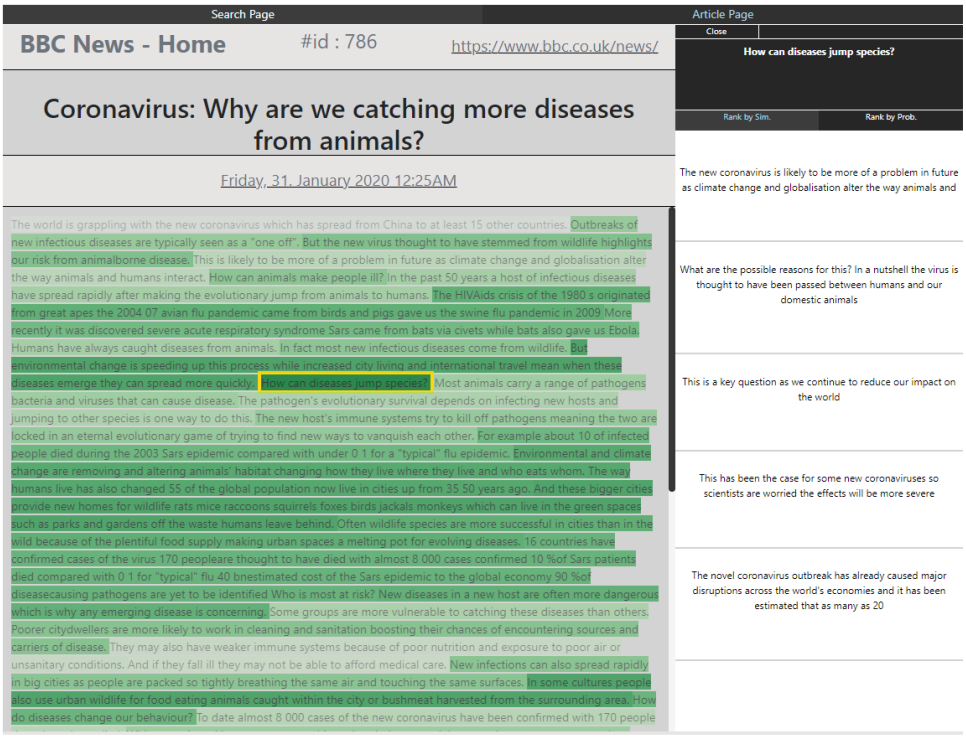


Figure 6.6: The *article view* quickly draws the attention towards the interesting passages of the article, as passages about *animal diseases* are heavily highlighted. Upon clicking on the sentence *How can diseases jump species?*, the model generated alternatives disclose the relation between the topic of this article and the *coronavirus*.

the sentence *How can diseases jump species?*, highlighted in figure 6.6, we would like to know the context of this question and the relation to the topic of *coronavirus*. We can consult the knowledge the underlying model has compiled, by displaying the alternative sentences the model proposes

instead of the question. Figure 6.6 shows alternative sentences outlined in red, illustrating the connection the model has drawn between the preceding context about *animal diseases* and the *coronavirus*.

6.1.6 Identify prevalent topics

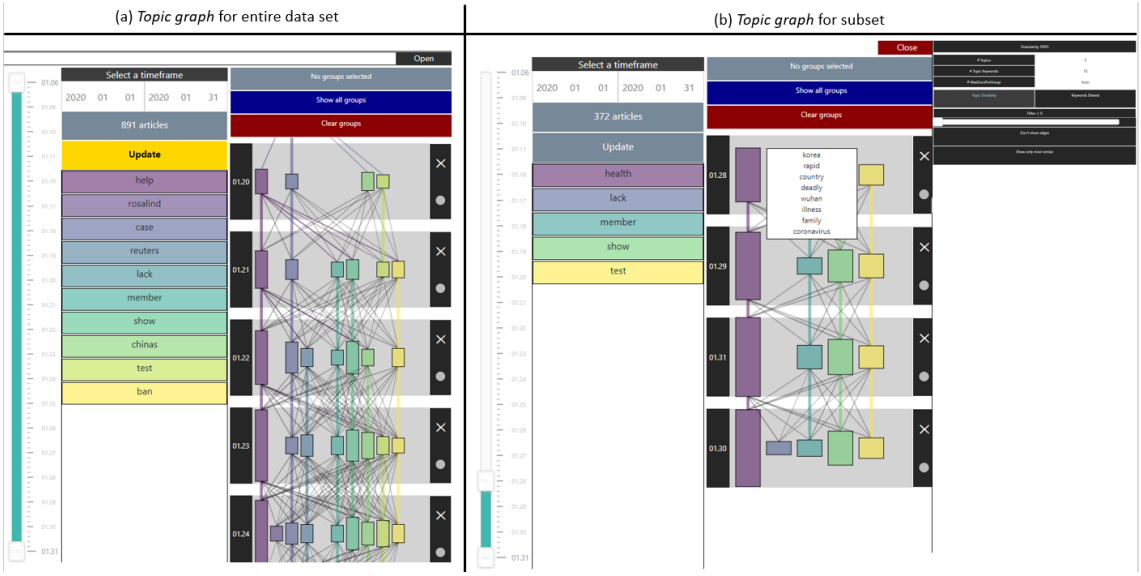


Figure 6.7: The *topic graph* in (a) displays the topics for the entire data set, covering a broad set of topics. The *topic graph* in (b) narrows down the time frame and produces clusters for five topics, as manually set via the menu. Hovering over the purple topic bars associated discloses a general *health* related and especially *coronavirus* related topic.

The selected articles we examined in less or more detail already covered a broad set of topics, which now prompts us to look at the evolution of topics in the data set. Drawing the *topic graph* for the data set, figure 6.7(a) highlights an unfortunate, yet expected side effect of the modified tf-idf vectors we utilis for the topic modelling. While we would expect very frequently covered terms like *coronavirus* to be a major prevalent topic, the discount induced by the modification of the weighting scheme likely penalized such terms. As a result, we have a rather broad set of topics. We thus refine the displayed *topic graph* in figure 6.7(b) by restricting the time frame to the end of march, as we know that the density of articles was high for those days. We further search for five topics overall, producing bigger topic clusters. As we inspect the topic keywords of a selected topic bar, we identify the general topic *health* to be related to the *coronavirus*.

6.1.7 Track the evolution of topics and topic keywords

As we are interested in the evolution of the topic *health* and its relevance, we extend two distant time steps and examine their *cluster views*. Figure 6.8 shows that the general topics of each cluster are the same, while the strength of the topic *health* increases for the later, meaning at a later time step, cluster. The topic keywords seem slightly different, but do not provide enough indication whether the contents of the underlying articles change over time. We thus have a look at the *keyword view*

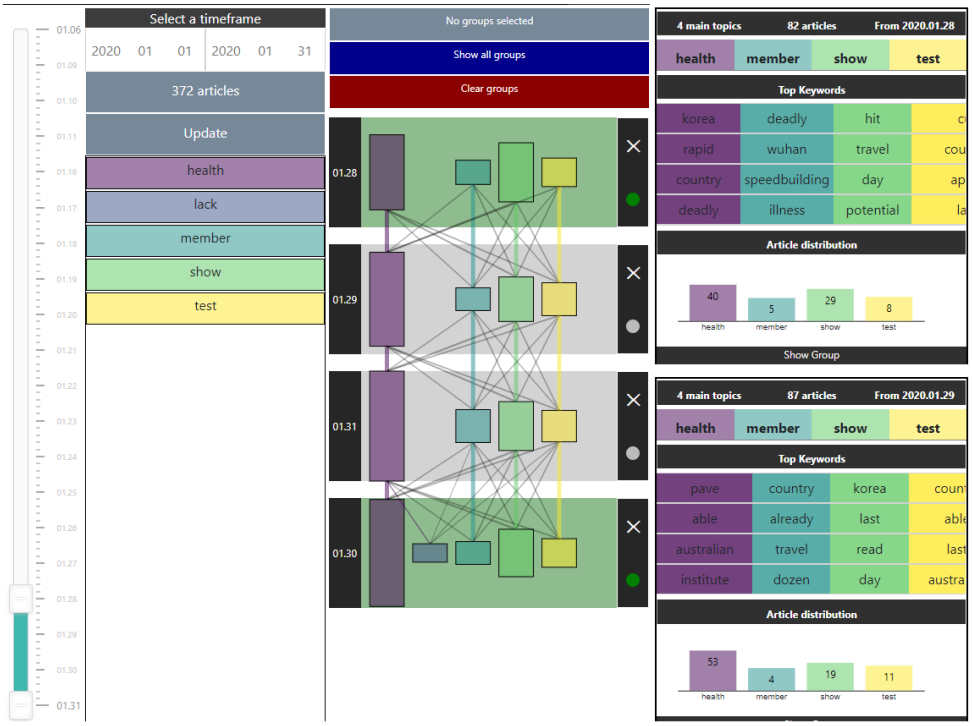


Figure 6.8: The *cluster views* of two distant time steps show that the general topics are similar, while the topic *health* gains relevance in the later cluster.

of each of of the clusters and inspect the keyword evolution in the underlying article sequences. Figure 6.9 side by side displays snippets of the *keyword views* of each cluster. Visible are the first ten articles of the underlying article sequences. Having set the number of keywords to display in each keyword list to ten and displaying only the recurring keywords in each article, we can identify that both clusters report topics closely related to and centered around keywords as *coronavirus*, *restrictions* and *outbreak*. In the earlier cluster in (a), we spot one, outlined in red, article standing out and seemingly reporting about *Kobe Bryant*, catching our interest.

We click on the keyword to open an in-place view of the passages containing the keyword *Bryant*. Reading the passages, we conclude that the anniversary of *Bryant's* death fell into the inspected time frame, thus shorty overshadowing the other topics.

(a) *Keyword view* of **earlier** cluster (a) *Keyword view* of **later** cluster

Figure 6.9: The *keyword views* of each of the clusters illustrate that the underlying article sequences report about content mostly related to *coronavirus*. In the earlier cluster (a), we spot an odd topic revolving around *Kobe Bryant*.

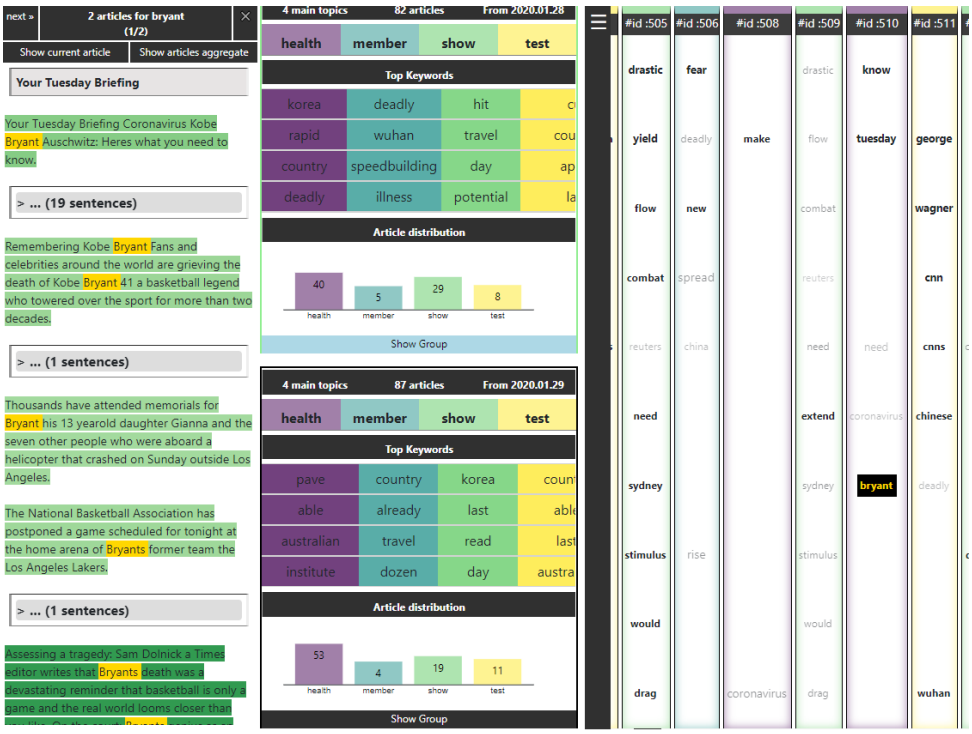


Figure 6.10: Inspecting the selected keyword in the view on the left side, provides us with the interesting passages from the corresponding article reporting about *Kobe Bryant*.

6.2 Discussion

The exemplary use case discussed in the previous section provided a first look into the effectiveness of the developed prototype for novel text exploration and topic evolution analysis in news story corpora, while revealing a range of shortcomings we need to address in the future. This section

takes a closer look at the advantages and disadvantages accompanying the interactive visualisation framework of this work and touches upon potential improvements and additions we concretize in chapter 7.

6.2.1 Novel text content identification

Central to our approach is the identification of novel text regions by utilizing the transformer network based language model GPT-2. Given a sequence of words to asses and a preceding textual context as prior knowledge, GPT-2 can accurately model the likelihood of the sequence, given the context. The likelihood assessment of GPT-2 functions as an intuitive approximation of the novelty of the sequence: As we define the novelty score of a sentence as the loss output of GPT-2 for that sequence, we can interpret the high loss as the model's inability to predict the sequence. This in turn indicates two major factors we desire to reveal, as we search for novel content: The given sequence consists of unusual content the model did not expect and the given context does not provide enough information about similar content or comparable patterns. Thus, the sequence of interest is likely to be novel with respect to the model's knowledge. This has mainly motivated the usage of GPT-2, as the transformer network's trained prior knowledge and likelihood assessments are among the most reliable in the area of neural language models. Additionally, GPT-2's ability to process long input sequences enables to model long distance relationships between the textual inputs, allowing the model to recognize similarities between the sequence to predict and distant sequences in the given context. That said, the likelihood assessments are in no way perfect and the modelling of long input sequences leave room for improvement. The technical restriction set by the maximal input length of 1024 tokens requires to narrow the possible prior context, potentially cutting off valuable information and thus leading to worse likelihood assessments. This is especially detrimental in the context of large news story corpora, as news topics might span over large sequences of news articles, where the first few articles reporting about the topic of interest offer valuable information and context. With the restriction on the context length, the model cannot access the valuable context when processing later articles of the sequence. We alleviated this issue by fine tuning the model, albeit only on 10% of the data set, which improves the model's underlying parameters and the model's performance on the given news corpus and similar text corpora. New application domains with structurally different text data would require to fine tune the initial model again, which is a computationally expensive task and thus not always a possibility. Generally, utilizing GPT-2 to model the novelty of text regions comes at the cost of a more difficult setup, compared to traditional statistical approaches like n-gram models. We decompose the given text documents and passages into sentences and assess the novelty sentence-wise, rather than calculating novelty scores for each token. While the sentence-wise assessment provides certain advantages, which we discussed in chapter 4, a major motivation against computing novelty scores for each word was the prolonged processing speed. As we built an interactive visualisation framework around the computed novelty scores, slow processing speeds are very disadvantageous with respect to the response time and interactivity of the system. As a trade-off for a better interactive framework, we miss out on individual, word level novelty scores, giving potential insight into which words specifically contribute to the novelty of a text region.

6.2.2 Modified tf-idf scores

To summarise individual articles and to extract important and novel keywords for topic modelling, we modified the tf-idf weighting scheme with the insights gained from the novelty assessment of individual text segments. The modification consisted of two added components, each part penalizing terms depending on the novelty of their originating text region: We discount the term frequency of individual tokens by counting each occurrence of a term in a document not with a weight of 1, but with the weight of the corresponding sentence's novelty score. As a result, the term frequency of a term in a document reflects the average novelty of the sentences within the document the term appears in. The document frequency of a term is additionally weighted by the average document novelty of all documents the term appears in. This works as a discount of terms that may appear frequently only in a selected few documents, getting assigned a high document frequency after the standard weighting scheme, but where the documents possess little new novel content. When querying for the top keywords of a document, with respect to the modified tf-idf score, the modified scoring thus prioritizes a mix of representative terms and terms appearing in highly novel text regions. At the same time, as we map documents to tf-idf vectors and cluster multiple documents, the discounted document frequency emphasises on terms appearing in novel documents. As the document space is separated based on the features, i.e. the terms, of the documents, these terms receive a greater influence in defining the resulting document groups. On one hand, this yields keyword-based article summaries with an heavy emphasise on novel and content-bearing keywords. On the other hand, it counteracts and harms the retrieval of representative keywords with respect to an article's topic: Important, topic relevant keywords within articles are likely to appear often, in multiple passages of varying novelty. These keywords hold important information with respect to the overall content of the article, yet might get discounted heavily if some of the text passages or the article overall contain little new information otherwise. Furthermore, articles associated with a topic are likely to share the same topic keywords, while the overall novelty of the articles fluctuate as articles within a time ordered sequence cover the same or similar topics. This can be problematic in the context of topic modelling, as we have seen in the discussion of figure 6.7 in the use cases: We utilis NMF to extract topic and assign these topics to articles based on the decomposition of the article keyword vectors. The discount of broadly appearing terms might yield high variance in the selection and assignment of topics, as the selection of frequent and continuously appearing keywords to represent topics is discouraged. As one of the goals of the prototype is to model the topic evolution and find both newly emerging and continuously developing topics among the given article corpus, the latter requires to either refine the modification or give the user the option to selectively remove the modification. We discuss such refinements in chapter 7.

6.2.3 Novel text content visualisation

To visualise novel text regions in articles and sequences of articles, we proposed and developed the *article view*, the *summary view* and the *thumbnails*. The *article view*, as a dedicated presentation of novel content in individual article documents, allows the user to analyse the novelty of specific text regions in detail. The mapping of the sentence-wise GPT-2 based novelty scores to the background color hue and opacity of each sentence, provides an intuitive visual emphasise on the novelty of text regions. Utilizing a continuous color scale, we are able to map fine granular differences in novelty scores to visually different backgrounds. Composite text regions of highly novel sentences immediately draw the attention due to the more intensive background visualisation. At the same

time, novel and less novel textual regions are easy to distinguish as the latter blends into the background, further emphasising the former. Adding the option to Gaussian blur the transition between the background visualisation of the sentences, further allows to customize the presentation: If the novelty of individual passages fluctuates heavily, leading to stark contrasts in the background coloring and opacity, the user can activate the smoothing to create a more fluent visualisation. If the differences in novelty between regions are not as pronounced, the user can turn of the smoothing, emphasising more on the transitions. The visual results of the smoothing can be dependant on the chosen window size though, requiring either the fine tuning of the window size by the practitioner or an option to manually set the window size for the user. The smoothing results further depend heavily on the overall variance in the novelty scores, where little differences in the initial scores will lead to very similar smoothed scores, subsequently producing homogeneous and hard to distinguish backgrounds. This issue is partly connected to and aggravated by the sentence-wise novelty scoring, as each sentence reflects the average novelty of its tokens, smoothing out high and low value components. As a result, sentences of varying length might be scored similarly, despite longer sequences with a high average novelty score potentially holding more interesting information. The lack of the word-wise background visualisation in the *article view* further removes the ability to identify individual components and terms with the biggest influence on the overall novelty of certain passages. The sole mapping of the sentence-wise scores to the background color hue and opacity provides the opportunity for the visual emphasises of such components, potentially requiring additional visual encodings as discussed in chapter 7.

Thumbnails were introduced as an abstract summarisation of individual articles, providing an intermediary level overview of articles among a time-ordered sequence. The visual encoding of individual sentences as bar glyphs and coloring the bars according to the novelty score of the sentence, works as a space-efficient and easy to interpret representation of the novel content distribution in articles. When viewing multiple articles among a sequence, the *thumbnails* allow to quickly identify and distinguish articles containing interesting passages, without specifically inspecting each of the articles. The space-efficiency of *thumbnails* is coupled with the on demand presentation of novel keywords of each sentence. As the keywords are only shown on interaction with a *thumbnail*, the user can first interactively decide whether the novel content distribution of an article looks promising and individually gather more detailed information about the passages independently. Displaying only a selected few keywords provides the user with a short description of each sentence, abstracting from the content and summarising the important portions. *Thumbnails* thus alleviate the possible overload of information when multiple articles are to be inspected, leaving the decision to present details to the user. Though, the utility provided by the details on demand heavily depend on the extracted keywords. As we utilis the modified novelty scores to sort and extract high value keywords, the emphasises lies on novel keywords, yet it does not guarantee to return representative results, as discussed in the section before. This can conflict with the initial abstract presentation of the *thumbnails*, if the provided keywords on interaction do not match the expectation induced by the color coding of the sentence glyph. The visual encoding of the sentences is kept simple, where each glyph representing a sentence has the same length. As such, the visualisation lacks an indication about the length of individual sentences, which the constant number of keywords do not provide either. The user would thus need to individually inspect an article to distinguish long and short sequences, even if they share similar novelty scores. As we interactively present the keywords in *thumbnails* on hover, the user can only analyse *thumbnails* one

by one, requiring the user to preserve a mental map of multiple *thumbnails* if multiple articles are to be examined. In chapter 7, we thus discuss potential aggregation mechanisms and additional visual encodings to incorporate.

The *summary view* provides the highest level overview with respect to novel content inspection, summarising articles among a time-ordered sequence. Extracting the $k = 3$ most novel passages, i.e. sentences, from each article, the *summary view* constructs a coherent aggregation of the most interesting text passages found in a sequence of articles. Color coding the background of each extracted passage akin to the *text view*, the user is provided a quick to explore overview of the novel text regions in the specified time frame. Complementing each summary of an article with a collapsed, interactive visual indicator of the article content missing in the summary, enables the user to analyse articles of interest without changing views. The indicators were chosen as horizontal bars, where each bar of 1 pixel width encodes the existence of five *hidden* sentences within the collapsed passage. The indicators are thus space-efficient and easy to interpret, providing an overview of the amount of text omitted from each article. Coupling the *summary view* with an interactive time frame selection mechanism, allows for a fine granular restriction and selection of arbitrary article sequences. Complementing the interactive functionality with a search and a filter mechanism, enables the user to search for articles with specific identifier or containing specific keywords and phrases or filter the given sequence with respect to a specified novelty score threshold for each article. The provided search mechanism is kept simple, leaving room for the addition of more complex search patterns, like defining regular expressions. Similarly, the filter mechanism only allows for filtering entire articles based on their average novelty, while a more sophisticated search of specific text passages exceeding a predefined novelty threshold is not possible. The vertical alignment of the extracted text passages in the *summary view* akin to a list provides a simple overview, but lacks a visual indicator towards novel text passages currently not in view. Potentially interesting passages can thus be easily missed, if the specified time frame produces a long sequence of articles. While the horizontal bar indicators for the hidden passages display roughly the amount of collapsed text, the stacking of the horizontal bars does not scale well for long article sequences. This in turn complicates the examination of novel text passages, as they are spread apart by the stacked indicators occupying large portions of the canvas. Potential improvements to the *summary view* thus need to address the interactive functionalities of the view, the presentation of the novel passages and the visual encoding chosen for the indicators, further discussed in chapter 7

Overall the combination of each of three views provides an efficient workflow, starting with the *summary view* as a high level search space to aggregate potentially interesting articles, identify a subset of articles for further examination with the help of the *thumbnails* and analyse individual articles in the *text view*.

6.2.4 Topic evolution visualisation

The visualisation of prevalent topics in the given corpus of news stories was a second major focus of the approach presented in this work. Similar to the text view of the developed prototype, we composed a multi-level approach of different visualisation levels. On the highest level, we provided an overview of the topic distribution in the *topic graph*. After specifying a time range, the *topic graph* produces a vertical alignment of topic nodes, each node corresponding with a time step and visualising the topic distribution for the articles associated with the time step. The vertical alignment akin to a list allows for the scalability of the graph, visualising large sequences of article

clusters by displaying a portion of the graph at a time as the user scrolls through it. Presenting the topic distribution in the node of a time step as a bar chart, provides an intuitive overview of the prevalent topics and an easy to understand encoding of the topic strength in the height of each bar. It further allows to sort the topic bars consistently across all time steps, crucially enabling the traction of specific topics. If topics are prevalent across multiple time steps, the corresponding bar charts visualising each topic distribution will position the associated topic bar the same. This allows to easily track topics continuing over multiple time steps, as well as identify newly emerging topics. Connecting the topic bars of adjacent topic nodes with edges, where the edge strength encodes the similarity of the topics, further simplifies the classification of similar topics, as well as similar topic distributions. Providing the user with two modes to determine the similarity, either by cosine similarity or the set difference of the topic keyword vectors, enables to define varying degrees of similarity. The interactive topic evolution exploration is rounded up by both the edge filtering and topic filtering mechanisms, restricting the visualisation to the emphasis of topic clusters exceeding a minimum similarity score or clusters associated with specific topics of interest. The edge filter and removal functionality coupled with the node collapse functionality allows the user to reduce the visual clutter, if length of the graph or the density of the drawn edges grow too large. While the chosen layout and the filter capabilities secure the scalability with respect to the occupied canvas space and visual clutter, the *topic graph* provides no real way to observe the entirety of the topic evolution if the underlying article subset is too large. The topic evolution visualisation only shows a subset of the topic nodes at any time. In junction with the layer-wise alignment of the nodes, it is not possible to visually compare the topic distributions of two distant time steps without memorization. The bar chart visualisation of the topic distribution at a time step further faces scalability issues if the number of topics grows too large. This produces very thin topic bars or forces to employ a scrolling mechanism, resulting in the same issues as described for the topic nodes. Within this context, connecting each topic bar with edges drawn as straight lines between the bars, inevitably produces visual clutter. Figure 6.7 shown in the discussed use case hints at the issue of densely drawn edges. The provided filter mechanisms allow to reduce the clutter in a coarse grained fashion, but do not allow to restrict the visualisation to a fine grained selection of specific connections. Chapter 7 thus discusses potential improvements on the graph layout, the topic distribution visualisation and the filter mechanisms.

The *cluster view* functions as the natural progression from the *topic graph* to a lower level abstraction, providing an intermediary level overview of specific topic distributions of interest. The user is able to interactively select arbitrary time steps of interest and construct a customized list of *cluster views*, each view displaying information about the topics and articles of the associated time frame. Each *cluster view* provides the necessary information to understand the content of each topic prevalent at a time step and the strength of the topics. As the user can construct arbitrary groupings of time steps, the relevance of topics occurring across distant time steps can be easily compared in the *cluster view*. Apart from the details about a selected topic distribution, the *cluster view* is comparatively ”empty” and leaves room to encode and visualise more information. We have seen glimpses of this in the use case discussing figure 6.8. While the presentation of the topic keywords as lists enables to compare the topic contents, the presentation does not effectively encode similarities, as the user has to read each entry and manually search for shared keywords. The bar chart visualising the article distribution faces the same issues as the bar chart in the topic nodes, providing an intuitive, but not very precise overview of the topic strengths. We could extend the *cluster view* with further details and interactive mechanisms to explore the topics, thus being further discussed in chapter 7.

Lastly, the *keyword view* completes the multi-level visualisation with a low level view of the articles associated with a specific topic cluster. The articles are presented as vertically aligned lists of their keyword vectors, each list ordered along the horizontal axis with respect to the publication date of the article. This enables to easily compare articles along the flow of time and determine the progression of relevant keywords. As we map the tf-idf score and frequency of each word to its font size and font strength, the user can track the relevancy of specific topics shared among multiple articles. By mapping the document frequency of a keyword up until a specific article to its opacity in the corresponding keyword list, the user is able identify emerging and fading keywords. The *keyword view* is completed with a sorting mechanisms to group articles according to their topic association, a filter mechanism to filter keywords occurring at most once or at least twice and a search mechanisms to only display occurrence of a specific keyword. These interactive tools allow the user to search for, identify and track the evolution of specific keywords and keyword subsets and examine their relevance with respect to the article's topic. The interactive, in place presentation of a view displaying the articles a selected keyword appears in, rounds up the user-driven drill down of a topic distribution of interest. Similar to the *cluster view*, the *keyword view* provides little additional information apart from the keyword content and their relevancy progression. The keyword lists additionally suffer from scalability issues, if the corresponding article clusters consist of large article batches. This in turn aggravates the comparison of articles at distant time steps, similar to the issue of topic nodes, which was one of the reasons for the discussed use cases to display only a subset of articles in figure 6.9. Chapter 7 explores some of the potential enhancements to the *keyword view*.

Overall, the developed prototype provides a multitude of interactive visualisations to explore novel text content and the topic evolution of news story corpora. The resulting framework suffers from scalability issues as very large sequences of articles are to be examined and leave room for improvement with respect to the novelty assessment and identification of representative topics.

7 Conclusion and Outlook

This final chapter concludes the thesis by recapping the approach presented in this work, closing the chapter with an outlook on the general future of the developed prototype, including potential additions and improvements.

Summary

The goal of the proposed approach was to build an interactive visualisation framework that combines state-of-the-art novelty detection techniques and topic modelling techniques, to visualise novel text content and the evolution of topics in news story corpora. For that, we first reviewed the fundamental concepts building the basis for the developed prototype. We introduced basic and advanced Machine Learning and Natural Language Processing (NLP) techniques to clean and process the articles of the underlying corpus, producing tf-idf based vector representation of each article. We continued with a look at Language Modelling and Topic Modelling, before we summarised the essential and relevant techniques of Deep Learning. Starting with Deep Neural Networks, we discussed several neural network architectures, leading to the discussion of neural language models. We concluded the foundations with the introduction of the GPT-2 transformer network based language model, which we later utilised in our approach. As the second step towards the realization of the approach, we discussed a range of related works, shining light on different approaches to identify and visualise novel text and the evolution of topics. Drawing valuable insights from the reviewed works, we then concluded a set of requirements the prototypical realization of our approach has to fulfill. The defined requirements guided the subsequent discussion of and decision about different concepts and designs of the components to incorporate into the prototype. The first requirement defined the need for a component to asses the novelty of a text region, given a preceding context. We utilised the previously mentioned GPT-2 language model to predict the likelihood of an input sequence, given a context, deriving a novelty score function to identify novel text passages. Building on the novelty scores, we defined the requirement to visualise the novel text passages in individual articles, as well as sequences of articles. For that, we introduced the concepts of the *article view*, *summary view* and *thumbnails* as different views visualising the novel content in specific articles, in article summaries and in aggregations of sequences of articles. As the third requirement, we identified the task of finding representative and novel topics among a given sequence of articles. To solve the task, we first introduced a modified tf-idf weighting scheme, utilising the novelty scores. With the emphasise on novel and representative keywords, we mapped each article to a vector of its keywords, subsequently applying Non-negative Matrix Factorization to extract prevalent topics from the articles. Tying in on the topic modelling, we fulfilled the fourth requirement of visualising the topic evolution by designing a multi-level visualisation component. The component consists of a *topic graph* for the general topic overview, the *cluster view* to inspect specific topic clusters of interest and the *keyword view* to examine the keyword evolution among the articles of a cluster.

The fifth requirement defined the need to provide multiple perspectives on the visualisations of the approach, which we achieved by integrating Multiple Coordinated Views, combining each of the views described. As the last requirement, we concluded the need to interactively manipulate the data presentation, for which we included interactive selection, search and filter mechanisms in each of the views. We then presented the resulting interactive visualisation framework, before detailing the technicalities of the implementation. Lastly, we presented a real-word use case to showcase the application of the developed prototype, followed by a discussion of the advantages and shortcomings of the prototype.

Outlook

During the discussion of the prototype in chapter 6, we have shone light on issues and potential points of improvement. The novelty scoring with GPT-2 is currently based on an approximation of the conditional likelihood, due to the context size restriction of 1024 token for GPT-2. The scores thus do not reflect the actual likelihood of the sentence provided a sufficiently long context, but rather based on a ”short sighted” window. We could improve the novelty scores by utilising a bigger model like XLNet [1], which allows for longer input sequences. That said, arbitrarily long input sequences are not quite feasible yet, at least if the computational resources are limited. The fast technical advancement of the field of Deep learning might provide a different solution in the future. We calculate sentence-wise novelty scores in the developed prototype, due to inefficient computation of token-wise likelihoods, which could influence the response time of the framework. We could define minimum system requirements for the execution of the prototype, enabling the computation of token-wise likelihoods without a drop in performance. Alternatively, we could look to outsource the heavy compute components, like the novelty scoring, by hosting the prototype in the cloud for example. Token-wise novelty scores would further enable to employ a more fine grained modification of the tf-idf scores of article vectors, discounting tokens based on the token scores. The proposed modification requires a slightly different approach to dissolve the conflicting weighting of representative keywords and novel keywords. One possible solution could be to base the modification on a different weighting scheme or even a different word vector representation, for example by utilising modified word embeddings. In the *article view*, we can further improve the novelty score dependant background visualisation. For that, we can construct a more sophisticated background visualisation by not only mapping the novelty score to the color hue and opacity, but also the font size for example. We can further incorporate additional smoothing techniques, emphasising on outliers for example. The *summary view* provides several points for improvement, starting with the encoding of the volume of the hidden text passages. We can employ a more space-efficient, but less direct encoding, for example utilising a constant sized glyph or object, whose background color is determined by the volume of the hidden text passage. Alternatively, we could simplify the display by showing the actual number of words or sentences, without any visual indicator. In terms of interactive functionalities, the *summary view* could be extended by a mechanism to search based on regular expressions, as well as search for topics, allowing for a more fine grained search. The latter point would require to couple the text view and the topic view of the prototype, such that intermediary results can be shared without recomputation. Allowing to fully collapse or remove passages of specific articles would further provide a solution if the *summary view* presents many

[1] https://huggingface.co/transformers/model$_doc/xlnet.html$

articles of little interest for the user. For the *thumbnails*, we could employ a selection mechanism to specify multiple *thumbnails* to enlarge, solving the issue of only being able to enlarge one *thumbnail* at a time. To incorporate the length of sentences in the glyph based sentence representation, we could map the length of the sentence to the length of the bar or display the length as textual label. The *topic graph* visualisation could be refined by selecting a different layout, relocating the graph into a separate view and utilising the entire canvas space to position topic nodes according to similar topic distributions for example. The scalability issue of the topic distribution bar charts could be solved by choosing a different, more space-efficient visualisation like Treemaps [2] for example. An important interactive enhancement of the topic graph could be a search mechanism to search for specific topics, extending the search restriction to the *cluster view* and the *keyword view*. The *cluster view* can be extended to display more information, for example show a statistical summary of the novel content found in the articles of the selected topic cluster. The *keyword view* could add an additional, alternative display of the article keywords, for example a StreamGraph [3] themed visual flow of the keywords over time. Such a differently aligned visualisation can alleviate the issues of scalability, while providing a different perspective on the keyword progression. In terms of utility, a future task is to move away from the requirement to process the data in advance and allow to flexibly select arbitrary time frames, providing the fast computation of the article related data. This can then be extended to a streaming approach, processing articles as they are fetched from the web. Lastly, we could extend the interactive visualisation framework with additional views. One such potential view could be an *evaluation view*, where multiple articles can viewed, examined and compared side by side. Additionally, a *statistics view* could visualise statistical properties of the corpus, for example the average novelty score across arbitrary time frames or a numerical assessment of the topics prevalent across such time frames.

[2] https://en.wikipedia.org/wiki/Treemapping
[3] https://en.wikipedia.org/wiki/Streamgraph

Bibliography

[BGN08] S. Bateman, C. Gutwin, M. Nacenta. "Seeing Things in the Clouds: The Effect of Visual Features on Tag Cloud Selections". In: Association for Computing Machinery, 2008, pp. 193–202. DOI: 10.1145/1379092.1379130. URL: https://doi.org/10.1145/1379092.1379130 (cit. on p. 40).

[BKS+19] M. Biswas, V. Kuppili, L. Saba, D. R. Edla, H. S. Suri, E. Cuadrado-Godia, J. R. Laird, R. T. Marinhoe, J. M. Sanches, A. Nicolaides, J. S. Suri. "State-of-the-art review on deep learning in medical imaging". In: *Frontiers in bioscience (Landmark edition)* 24 (Jan. 2019). DOI: 10.2741/4725. URL: https://doi.org/10.2741/4725 (cit. on p. 26).

[BL06] D. M. Blei, J. D. Lafferty. "Dynamic Topic Models". In: ICML '06. Association for Computing Machinery, 2006, pp. 113–120. DOI: 10.1145/1143844.1143859. URL: https://doi.org/10.1145/1143844.1143859 (cit. on p. 42).

[BLC19] I. Beltagy, K. Lo, A. Cohan. "SciBERT: A Pretrained Language Model for Scientific Text". In: Association for Computational Linguistics, Nov. 2019, pp. 3615–3620. DOI: 10.18653/v1/D19-1371. URL: https://www.aclweb.org/anthology/D19-1371 (cit. on p. 33).

[Ble12] D. M. Blei. "Probabilistic Topic Models". In: *Commun. ACM* 55 (Apr. 2012). DOI: 10.1145/2133806.2133826. URL: https://doi.org/10.1145/2133806.2133826 (cit. on p. 24).

[BM97] E. Brill, R. J. Mooney. "An Overview of Empirical Natural Language Processing". In: *AI Magazine* 18.4 (Dec. 1997). DOI: 10.1609/aimag.v18i4.1318. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/1318 (cit. on p. 15).

[BNJ03] D. M. Blei, A. Y. Ng, M. I. Jordan. "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3 (Mar. 2003) (cit. on pp. 24, 42).

[CG99] S. F. Chen, J. Goodman. "An empirical study of smoothing techniques for language modeling". In: *Computer Speech Language* 13 (1999). DOI: https://doi.org/10.1006/csla.1999.0128. URL: https://www.sciencedirect.com/science/article/pii/S0885230899901286 (cit. on p. 22).

[CLRP13] J. Choo, C. Lee, C. K. Reddy, H. Park. "UTOPIAN: User-Driven Topic Modeling Based on Interactive Nonnegative Matrix Factorization". In: *IEEE Transactions on Visualization and Computer Graphics* 19 (2013). DOI: 10.1109/TVCG.2013.212 (cit. on p. 44).

[DCLT19] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: 2019. DOI: 10.18653/v1/N19-1423 (cit. on pp. 33, 34).

[EBA19] S. Edunov, A. Baevski, M. Auli. “Pre-trained language model representations for language generation”. In: Association for Computational Linguistics, June 2019, pp. 4052–4059. DOI: 10.18653/v1/N19-1409. URL: https://www.aclweb.org/anthology/N19-1409 (cit. on p. 33).

[EHR+14] A. Endert, M. Hossain, N. Ramakrishnan, C. North, P. Fiaux, C. Andrews. “The human is the loop: new directions for visual analytics”. In: *Journal of Intelligent Information Systems* 43 (Dec. 2014). DOI: 10.1007/s10844-014-0304-9 (cit. on p. 51).

[FFB18] C. Felix, S. Franconeri, E. Bertini. “Taking Word Clouds Apart: An Empirical Investigation of the Design Space for Keyword Summaries”. In: *IEEE Transactions on Visualization and Computer Graphics* 24 (2018). DOI: 10.1109/TVCG.2017.2746018 (cit. on p. 41).

[FHY19] H. Fujiyoshi, T. Hirakawa, T. Yamashita. “Deep learning-based image recognition for autonomous driving”. In: *IATSS Research* 43 (2019). DOI: https://doi.org/10.1016/j.iatssr.2019.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0386111219301566 (cit. on p. 26).

[Fra06] L. A. Francis. “Taming Text: An Introduction to Text Mining”. In: 2006 (cit. on p. 11).

[GBC16a] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on pp. 16, 27, 29).

[GBC16b] I. Goodfellow, Y. Bengio, A. Courville. “Deep Learning”. In: MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on p. 29).

[GFC04] M. Ghoniem, J.-D. Fekete, P. Castagliola. “A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations”. In: 2004, pp. 17–24. DOI: 10.1109/INFVIS.2004.1 (cit. on p. 37).

[GH11] E. Gedraite, M. Hadad. “Investigation on the effect of a Gaussian Blur in image filtering and segmentation”. In: Jan. 2011, pp. 393–396. ISBN: 978-1-61284-949-2 (cit. on p. 36).

[GJG+15] S. Gad, W. Javed, S. Ghani, N. Elmqvist, T. Ewing, K. N. Hampton, N. Ramakrishnan. “ThemeDelta: Dynamic Segmentations over Temporal Topic Models”. In: *IEEE Transactions on Visualization and Computer Graphics* 21 (2015). DOI: 10.1109/TVCG.2014.2388208 (cit. on p. 45).

[GS04] T. L. Griffiths, M. Steyvers. “Finding scientific topics”. In: *Proceedings of the National Academy of Sciences* 101 (2004). DOI: 10.1073/pnas.0307752101. URL: https://www.pnas.org/content/101/suppl_1/5228 (cit. on p. 24).

[HBWP13] M. D. Hoffman, D. M. Blei, C. Wang, J. Paisley. “Stochastic Variational Inference”. In: *Journal of Machine Learning Research* 14.4 (2013). URL: http://jmlr.org/papers/v14/hoffman13a.html (cit. on p. 24).

[Hei17] F. Heimerl. “Exploratory visual text analytics in the scientific literature domain”. In: 2017. DOI: 10.18419/OPUS-9218 (cit. on pp. 15, 19).

[HHN00] S. Havre, B. Hetzler, L. Nowell. “ThemeRiver: visualizing theme changes over time”. In: 2000, pp. 115–123. DOI: 10.1109/INFVIS.2000.885098 (cit. on p. 45).

[HLLE14] F. Heimerl, S. Lohmann, S. Lange, T. Ertl. “Word Cloud Explorer: Text Analytics Based on Word Clouds”. In: 2014, pp. 1833–1842. DOI: 10.1109/HICSS.2014.231 (cit. on pp. 40, 41).

[HPP+20] M. A. Hearst, E. Pedersen, L. Patil, E. Lee, P. Laskowski, S. Franconeri. "An Evaluation of Semantically Grouped Word Cloud Designs". In: *IEEE Transactions on Visualization and Computer Graphics* 26 (2020). DOI: 10.1109/TVCG.2019.2904683 (cit. on p. 41).

[HS97] S. Hochreiter, J. Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997). DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 30).

[HSG20] B. Hoover, H. Strobelt, S. Gehrmann. "exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models". In: Association for Computational Linguistics, July 2020, pp. 187–196. DOI: 10.18653/v1/2020.acl-demos.22. URL: https://www.aclweb.org/anthology/2020.acl-demos.22 (cit. on p. 41).

[HSS15] K. Hara, D. Saito, H. Shouno. "Analysis of function of rectified linear unit used in deep learning". In: 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280578 (cit. on pp. 27, 28).

[HTF09] T. Hastie, R. Tibshirani, J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009. URL: http://www-stat.stanford.edu/~tibs/ElemStatLearn/ (cit. on p. 18).

[Hua08] A. Huang. "Similarity measures for text document clustering". In: *Proceedings of the 6th New Zealand Computer Science Research Student Conference* (Jan. 2008) (cit. on pp. 20, 21).

[Hug20] Huggingface. *How to generate text using different decoding methods*. 2020. URL: https://huggingface.co/blog/how-to-generate (visited on 07/01/2021) (cit. on p. 35).

[ID10] N. Indurkhya, F. J. Damerau. *Handbook of Natural Language Processing*. 2nd. Chapman amp; Hall/CRC, 2010. ISBN: 1420085921 (cit. on p. 18).

[JM09] D. Jurafsky, J. H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, 2009. ISBN: 9780131873216 0131873210. URL: http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y (cit. on pp. 18, 19, 22, 27–29).

[KB19] P. Kherwa, P. Bansal. "Topic Modeling: A Comprehensive Review". In: *EAI Endorsed Transactions on Scalable Information Systems* 7.24 (July 2019). DOI: 10.4108/eai.13-7-2018.159623 (cit. on p. 11).

[KKE18] J. Knittel, S. Koch, T. Ertl. "Highlighting Text Regions of Interest with Character-Based LSTM Recurrent Networks". In: 2018 (cit. on pp. 12, 40, 42).

[KMDC18] F. Karim, S. Majumdar, H. Darabi, S. Chen. "LSTM Fully Convolutional Networks for Time Series Classification". In: *IEEE Access* 6 (2018). DOI: 10.1109/ACCESS.2017.2779939 (cit. on p. 31).

[KNMK13] M. Krstajić, M. Najm-Araghi, F. Mansmann, D. A. Keim. "Story Tracker: Incremental visual text analytics of news story development". In: *Information Visualization* 12 (2013). DOI: 10.1177/1473871613493996. URL: https://doi.org/10.1177/1473871613493996 (cit. on pp. 12, 44, 45).

[LRKC10] B. Lee, N. H. Riche, A. K. Karlson, S. Carpendale. "SparkClouds: Visualizing Trends in Tag Clouds". In: *IEEE Transactions on Visualization and Computer Graphics* 16 (2010). DOI: 10.1109/TVCG.2010.194 (cit. on p. 40).

[LTD+16] L. Liu, L. Tang, W. Dong, S. Yao, W. Zhou. "An overview of topic modeling and its current applications in bioinformatics". In: *SpringerPlus* 5 (2016). URL: https://doi.org/10.1186/s40064-016-3252-8 (cit. on p. 24).

[LW15] X. Li, X. Wu. "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition". In: 2015, pp. 4520–4524. DOI: 10.1109/ICASSP.2015.7178826 (cit. on p. 31).

[MCCD13] T. Mikolov, K. Chen, G. Corrado, J. Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013 (cit. on p. 33).

[Mit97] T. M. Mitchell. *Machine Learning*. 1st ed. McGraw-Hill, Inc., 1997. ISBN: 0070428077 (cit. on p. 16).

[MMO+20] L. Martin, B. Muller, P. J. Ortiz Suárez, Y. Dupont, L. Romary, É. de la Clergerie, D. Seddah, B. Sagot. "CamemBERT: a Tasty French Language Model". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020). DOI: 10.18653/v1/2020.acl-main.645. URL: http://dx.doi.org/10.18653/v1/2020.acl-main.645 (cit. on pp. 33, 34).

[MRS08] C. D. Manning, P. Raghavan, H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511809071 (cit. on pp. 23, 27).

[MSH+13] S. Malik, A. Smith, T. Hawes, P. Papadatos, J. Li, C. Dunne, B. Shneiderman. "TopicFlow: Visualizing Topic Alignment of Twitter Data over Time". In: ASONAM '13. Association for Computing Machinery, 2013, pp. 720–726. DOI: 10.1145/2492517.2492639. URL: https://doi.org/10.1145/2492517.2492639 (cit. on pp. 44, 45).

[MTG09] S. Meyn, R. L. Tweedie, P. W. Glynn. *Markov Chains and Stochastic Stability*. 2nd ed. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511626630 (cit. on p. 22).

[Mur21] K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2021. URL: probml.ai (cit. on pp. 17, 27, 29).

[PCCT14] M. A. Pimentel, D. A. Clifton, L. Clifton, L. Tarassenko. "A review of novelty detection". In: *Signal Processing* 99 (2014), pp. 215–249. DOI: https://doi.org/10.1016/j.sigpro.2013.12.026. URL: https://www.sciencedirect.com/science/article/pii/S016516841300515X (cit. on p. 11).

[PCZ+19] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, Q. V. Le. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Interspeech 2019* (Sept. 2019). DOI: 10.21437/interspeech.2019-2680. URL: http://dx.doi.org/10.21437/Interspeech.2019-2680 (cit. on p. 26).

[PHJG19] R. Pokharel, P. D. Haghighi, P. P. Jayaraman, D. Georgakopoulos. "Analysing Emerging Topics across Multiple Social Media Platforms". In: ACSW 2019. Association for Computing Machinery, 2019. DOI: 10.1145/3290688.3290720. URL: https://doi.org/10.1145/3290688.3290720 (cit. on p. 43).

[PNI+18] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer. "Deep contextualized word representations". In: 2018. arXiv: 1802.05365 [cs.CL] (cit. on p. 33).

[PSM14] J. Pennington, R. Socher, C. Manning. "GloVe: Global Vectors for Word Representation". In: Association for Computational Linguistics, Oct. 2014. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162 (cit. on p. 21).

[PT94] P. Paatero, U. Tapper. "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values". In: *Environmetrics* 5 (1994). DOI: https://doi.org/10.1002/env.3170050203. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/env.3170050203 (cit. on pp. 25, 43).

[Rob07] J. C. Roberts. "State of the Art: Coordinated Multiple Views in Exploratory Visualization". In: 2007, pp. 61–71. DOI: 10.1109/CMV.2007.20 (cit. on p. 36).

[RWC+18] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. "Language Models are Unsupervised Multitask Learners". In: (2018). URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf (cit. on p. 40).

[SOR+09] H. Strobelt, D. Oelke, C. Rohrdantz, A. Stoffel, D. A. Keim, O. Deussen. "Document Cards: A Top Trumps Visualization for Documents". In: *IEEE Transactions on Visualization and Computer Graphics* 15 (2009). DOI: 10.1109/TVCG.2009.139 (cit. on p. 41).

[SVL14] I. Sutskever, O. Vinyals, Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: NIPS'14. MIT Press, 2014, pp. 3104–3112 (cit. on p. 31).

[TZ11] F. S. Tsai, Y. Zhang. "D2S: Document-to-sentence framework for novelty detection". In: *Knowledge and Information Systems* 29 (2011). DOI: 10.1007/s10115-010-0372-2. URL: https://doi.org/10.1007/s10115-010-0372-2 (cit. on p. 39).

[VSP+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). URL: http://arxiv.org/abs/1706.03762 (cit. on pp. 31, 41).

[WDS+20] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush. "Transformers: State-of-the-Art Natural Language Processing". In: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6 (cit. on p. 89).

[WLS+10] F. Wei, S. Liu, Y. Song, S. Pan, M. X. Zhou, W. Qian, L. Shi, L. Tan, Q. Zhang. "TIARA: A Visual Exploratory Text Analytic System". In: KDD '10. Association for Computing Machinery, 2010, pp. 153–162. DOI: 10.1145/1835804.1835827. URL: https://doi.org/10.1145/1835804.1835827 (cit. on pp. 43, 45).

[WT04] P. C. Wong, J. Thomas. "Visual Analytics". In: *IEEE Comput. Graph. Appl.* 24.5 (Sept. 2004). DOI: 10.1109/MCG.2004.39. URL: https://doi.org/10.1109/MCG.2004.39 (cit. on p. 11).

[XLG03] W. Xu, X. Liu, Y. Gong. “Document Clustering Based on Non-Negative Matrix Factorization”. In: SIGIR ’03. Association for Computing Machinery, 2003, pp. 267–273. DOI: 10.1145/860435.860485. URL: https://doi.org/10.1145/860435.860485 (cit. on p. 43).

[YC00] J. Y. Yam, T. W. Chow. “A weight initialization method for improving training speed in feedforward neural network”. In: 30 (2000). DOI: https://doi.org/10.1016/S0925-2312(99)00127-7 (cit. on p. 29).

[ZCM02] Y. Zhang, J. Callan, T. Minka. “Novelty and Redundancy Detection in Adaptive Filtering”. In: SIGIR ’02. Association for Computing Machinery, 2002, pp. 81–88. DOI: 10.1145/564376.564393. URL: https://doi.org/10.1145/564376.564393 (cit. on p. 39).

All links were last followed on July 08, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature