Master's thesis
Master's Programme in Data Science

Contrastive pretraining in discourse change detection

Mikko Lipsanen

March 13, 2022

Supervisor(s): Dr. Lidia Pivovarova

Examiner(s): Dr. Lidia Pivovarova, Professor Hannu Toivonen

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Mikko Lipsanen

Työn nimi — Arbetets titel — Title

Contrastive pretraining in discourse change detection

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	March 13, 2022	81

Tiivistelmä — Referat — Abstract

The thesis presents and evaluates a model for detecting changes in discourses in diachronic text corpora. Detecting and analyzing discourses that typically evolve over a period of time and differ in their manifestations in individual documents is a challenging task, and existing approaches like topic modeling are often not able to reach satisfactory results.

One key problem is the difficulty of properly evaluating the results of discourse detection methods, due in large part to the lack of annotated text corpora. The thesis proposes a solution where synthetic datasets containing non-stable discourse patterns are generated from a corpus of news articles. Using the news categories as a proxy for discourses allows both to control the complexity of the data and to evaluate the model results based on the known discourse patterns.

The complex task of extracting topics from texts is commonly performed using generative models, which are based on simplifying assumptions regarding the process of data generation. The model presented in the thesis explores instead the potential of deep neural networks, combined with contrastive learning, to be used for discourse detection.

The neural network model is first trained using supervised contrastive loss function, which teaches the model to differentiate the input data based on the type of discourse pattern it belongs to. This pretrained model is then employed for both supervised and unsupervised downstream classification tasks, where the goal is to detect changes in the discourse patterns at the timepoint level. The main aim of the thesis is to find out whether contrastive pretraining can be used as a part of a deep learning approach to discourse change detection, and whether the information encoded into the model during contrastive training can generalise to other, closely related domains.

The results of the experiments show that contrastive pretraining can be used to encode information that directly relates to its learning goal into the end products of the model, although the learning process is still incomplete. However, the ability of the model to generalise this information in a way that could be useful in the timepoint level classification tasks remains limited. More work is needed to improve the model performance, especially if it is to be used with complex real-world datasets.

ACM Computing Classification System (CCS):
Computing methodologies → Artificial intelligence → Natural language processing → Discourse, dialogue and pragmatics

Computing methodologies → Machine learning → Machine learning approaches → Neural networks

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

We are surrounded by an ever growing amount of textual data, large part of which is now available in electronic form. As the sheer volume of the data grows, it becomes increasingly difficult to extract information that concerns a large set of documents by going through them manually.

Therefore it is not surprising that the increase in computing power and the simultaneous development of new machine learning methods during the last decades has led to active research on methods and algorithms that can help to reveal linguistic patterns, shared topics, changing discourses, and even prevalent emotions, in large text corpora.

Often these corpora are structured diachronically, making the change and permanence of different elements over time a relevant research interest for historians and social scientists, among others. In these fields of study, the focus is typically on analysing the ways different phenomena (religion, politics, culture, sports etc.) are discussed in the texts, and discovering how the contents and patterns of the discussions have changed over time.

Topic modeling is an active field of research under Natural Language Processing (NLP). It focuses on developing methods that aim to reveal topics, defined by a specific vocabulary, that are shared by groups of documents. One key challenge for the topic models has been posed by diachronic text corpora, where the contents of the topics change over time, while still preserving some unifying elements.

Dynamic topic models have improved the ability to extract changes in the contents of the topics over time [23]. They do not, however, directly give answers to questions on the prevalence of topics at specific time points. To discover changes in topic frequencies over time, further analysis of the results is still needed. One solution is a two-stage model proposed by Duong et al. [7], which combines topic modeling with neural network based classification. This model also provides a starting point for the research that is presented in the following chapters.

The aim of this thesis is to contribute to the work of developing new tools and models that could facilitate the analysis of large scale text corpora. The hypothesis of the thesis is that a properly structured neural network model could be able to detect

changes in discourses over time when trained in an end-to-end fashion, while avoiding the pitfalls related to the alternative, two-stage model, presented by Duong et al. [7]. This hypothesis is tested in practice by developing a deep neural network model and testing its performance with synthetic datasets.*

The choice of the deep learning approach stems from the complexity of the data from the perspective of the proposed learning task. Besides their complexity, real world text corpora are also rarely, if ever, annotated in a way that would allow to use them for evaluating the performance of a discourse detection model. The solution proposed in the thesis is to use synthetic datasets, which are generated from a corpus of Finnish news articles and applied for training and evaluating the performance of the developed model. The use of synthetic data makes it possible to adjust the noisiness of the model input, and it also enables to view discourse change detection as a classification problem, as will be explained in more detail in Chapter 2.

A specific focus in the model training phase is the use of contrastive pretraining, which is an approach commonly implemented in a self-supervised setting, especially for image classification tasks, but which is still less common in the natural language processing field. In this thesis, the goal is to find out whether supervised contrastive pretraining can be used to improve the performance of the developed model in the discourse change detection task, and whether this approach could provide the basis for a model performing succesfully also in an unsupervised setting.

Chapter 2 discusses the definition of discourse and discourse change in the context of this work, and presents earlier research that is relevant as a background for the approach chosen in the thesis. Chapter 3 outlines the main features of contrastive learning, and explains the motivation for using it in the discourse change detection task. The three following chapters focus on the data and the models used in the experiments. Chapter 4 explains and illustrates the different stages of synthetic data generation, while Chapter 5 looks at the models developed for contrastive pretraining and Chapter 6 focuses on the downstream classification tasks. Chapter 7 presents the methods used for exploring and evaluating the learning results, and the results of the experiments conducted with the presented models are detailed in Chapter 8. Chapter 9 concludes the thesis with a brief outline of the presented model and experimental results, and discusses also the open questions and challenges related to the approach to discourse change detection presented in the preceding chapters.

This work has been partly supported by the European Union's Horizon 2020 research and innovation programme under grant 770299 (NewsEye).

*The code used in the experiments is available in GitHub at https://github.com/MikkoLipsanen/Discourse-change.

2. Detecting topics and discourses

This chapters begins with a brief definition of the meaning of discourse and discourse change in the context of the thesis. Research on discourse change detection is closely linked to the field of digital humanities, and topic models are the tools most commonly used for this purpose. For this reason topic models are discussed in their own section, followed by an outline of the research by Duong et al. [7], which provides a basis for the approach chosen in the thesis and the specific research questions that it aims to answer.

2.1 Defining discourse and discourse change

Defining what is meant by the term 'discourse' is a complex and contested issue, and providing a single, comprehensive description of a vast field of study is a challenging task [19]. However, from a more limited perspective that is relevant to the approach adopted in this thesis, discourse can be understood as a theme that is shared across individual texts, often related to a specific vocabulary and even linguistic forms, but not entirely reducible to them. Discourses can be analysed using a wide variety of sources, that can include historical documents, fiction, messages in social media platforms or, as in this study, news articles.

Following the example of Duong et al. [7], discourse is here defined in the framework of a diachronic text corpus, which consists of a large number of individual documents that can be placed on a timeline. Synthetic datasets built from news articles are presented as a compromise solution when properly annotated text corpora are lacking. While the use of synthetic data can negatively impact the generalizability of the model, it also makes it possible to give a concrete and specific definition of discourse in the context of the thesis. The category labels ('politics', 'sports', 'food' etc.) that journalists and other other experts have assigned to the articles are used as proxies for discourses, defining elements of shared semantic contents across individual documents. As the articles usually cover several categories, and texts belonging to different categories also share large amount of vocabulary and other features, the expression of the categories in the documents is 'noisy' in a way that is typical for discourses also in other domains.

This means that there is necessarily also a subjective element impacting the category assignments that are used to evaluate model performance. One way to compensate for this is to select easily distinguishable categories to be used in the model training, as will be explained in more detail in Chapter 4.

The major practical benefit of this definition of discourse is the possibility to use the news data for synthetically creating diachronic text corpora that can be utilized in the experiments. The process of data generation is described in detail in Chapter 4. As the prevalence of different discourses in synthetic datasets can be controlled up to a certain point, it is possible to evaluate and measure the performance of the discourse detection model in a much more systematic way than what is allowed by alternative approaches, where such labels are not available.

2.2 Topic models

Although the model developed in the thesis, presented in detail in Chapter 5, is largely based on different traditions than the generative approach commonly used in topic modeling, the work on discourse change detection builds on research done in this field.

Topic models are a tool commonly used in Natural Language Processing (NLP) for exploring latent, or 'hidden', topics from large text data sets [2]. They are, for the most part, based on generative probabilistic modeling, where the assumption is that the observed data has been created by a generative process, and the topics are defined as distributions over a vocabulary defined by hidden random variables that the model tries to infer.

One of the most commonly used topic modeling approaches is Latent Dirichlet Allocation (LDA) [2, 23], which has also been the basis for many later developments in the field. Figure 2.1 shows a probabilistic graphical model representation of the generative process assumed in LDA. First, for each of the M documents in the data set a topic distribution θ over a predefined number of topics K is randomly sampled from a Dirichlet distribution parameterized by α. The distribution of words in each topic is also sampled from a Dirichlet distribution, this time using the hyperparameter β. Then a topic is sampled for each 'potential' word in the document based on θ, and finally the actual word is sampled from the distribution of words in each topic. As is commonly the case with probabilistic models, the posterior distribution, or the conditional distribution over the hidden variables (such as topic proportions) given the data, needs to be approximated using for instance Monte Carlo sampling or variational methods.

LDA and similar topic models are built on some simplifying assumptions, which set limits to their utility in many use cases [2, 23]. First, both the order of the words

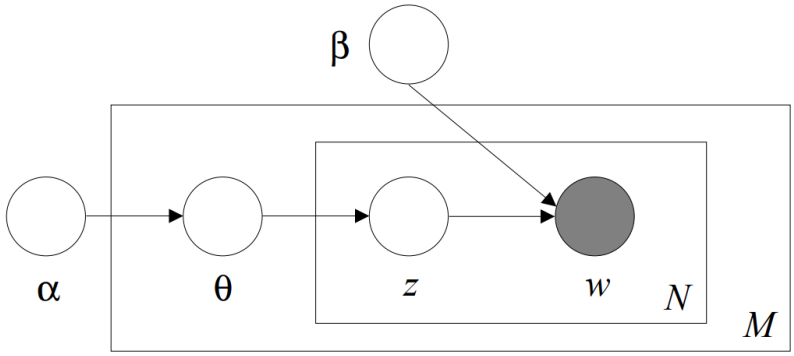


Figure 2.1: The generative process of the LDA topic modeling method. Image from [3].

in the documents as well as the order of the documents in the corpus are not taken into consideration in the model, which of course is a crude simplification. Also, the topics, defined as distributions over the vocabulary, are taken to be uncorrelated, and the number of topics in the input data is expected to be known beforehand .

Since the publishing of the paper describing LDA almost 20 years ago [3], numerous new topic modeling methods have been developed, and many of them are specifically designed to tackle some of the challenges mentioned above [23]. A recent example is Dynamic-Embedded Topic Model (D-ETM), which aims to improve the LDA approach by allowing dynamic changes in topics over time. D-ETM also uses word embeddings that reflect the semantic meanings of words in their contexts [5], thus bringing the word (and consecutively topic) representations closer to the real world, where independence assumptions, such as the ones made in LDA, rarely apply.

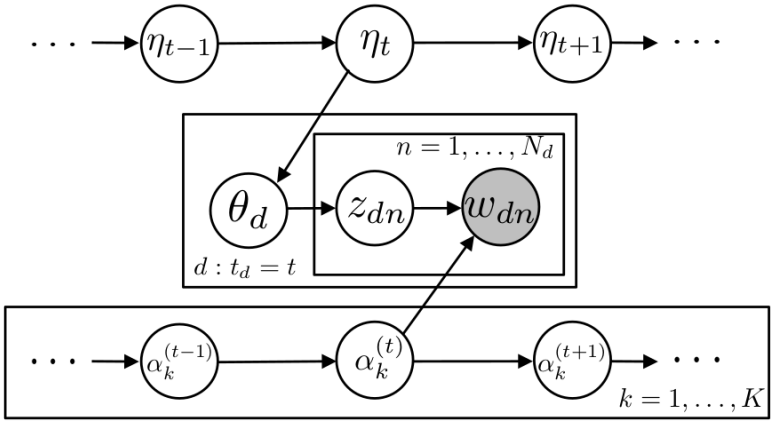


Figure 2.2: The generative process of the Dynamic-Embedded Topic Model (D-ETM). Both the distributions α defining the topics k and the topic proportion means η evolve over time. Image from [5].

As can be seen in Figure 2.2, in D-ETM both the topic proportions and the topic contents evolve over time, making it a suitable model for analysing time series

data. However, with D-ETM, and topic models more generally, the topic frequencies at each time point have to be inferred from the topic proportions, and further analysis is required to discover how different topics have changed over time. Therefore, when the aim is specifically to detect changes in the discourse trends, a two-stage model is required to achieve the goal.

2.3 Two stage model for discourse change detection

Synthetic datasets and LDA topic model are both part of the experiments conducted by Duong et al. [7] using a two-stage model that is illustrated in Figure 2.3. In the first part of the model, the documents are assigned into 'discourses' using two alternative methods, K-means clustering of the document embeddings, or LDA topic modeling. In the latter case, documents with a topic probability higher than a defined threshold are assigned to that topic's cluster. Since standard, non-dynamic LDA is used, separate models have to be built for each timepoint in the dataset. The output of this part of the model is a time series where each timepoint contains information about the number of documents assigned to the different topics.

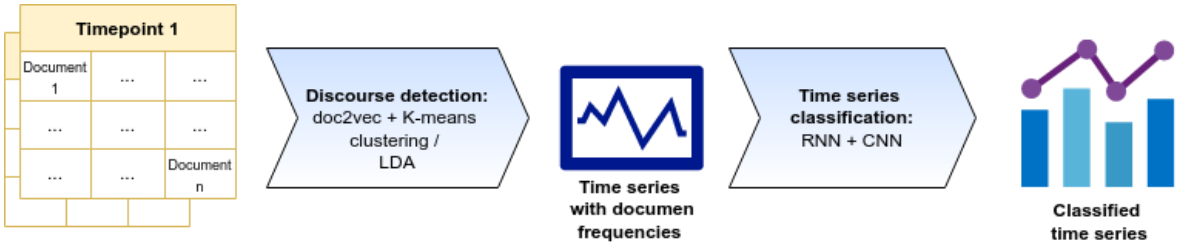


Figure 2.3: Two-part model for discourse change detection presented in Duong et al. [7]. Image by the author.

The second part of the discourse detection pipeline introduced by Duong et al. [7] uses the time series data to detect timepoints that belong to an non-stable discourse pattern. This module, presented in more detail in Figure 2.4, combines a Recurrent Neural Network (RNN) and a Convolutional Neural Network (CNN). The input to both networks consists of vectors of frequency values, one for each timepoint in the data. The output of the networks is split into two parts: one output layer predicts the probability that the input contains a non-stable discourse pattern, and the other output layer predicts a probability for each time point of being part of a non-stable discourse pattern.

Figure 2.5 gives a concrete example of what this means in practice. The two lines in the plot define the number of articles belonging to the two example categories, 'fires' ('tulipalot) and 'employment' ('työllisyys'), at each of the 100 timepoints. Some

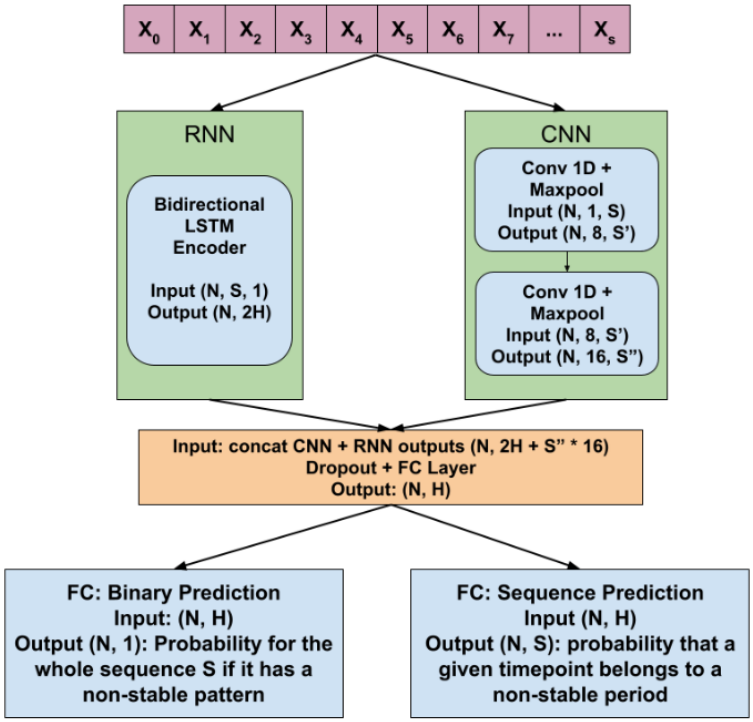


Figure 2.4: The combined RNN and CNN network architecture presented in Duong et al. [7]. Image from [7].

noise has been added to the article frequencies, but it can still be easily observed that the blue line makes an up-down curve in the middle of the timeline. All of the timepoints covered by this curve, highlighted with the red shading, are counted as part of a non-stable discourse pattern.

There are in total seven discourse patterns used by Duong et al., six non-stable patterns and a stable one, where noise creates the only variation in the document frequency. The non-stable patterns are presented in Figure 2.6 without added noise. The same patterns are also utilized in the synthetic data generation process for this study, explained in more detail in Chapter 4.

Although the model introduced in Duong et al. [7] is able to produce some promising results, there is still much room for improvement. The fact that the model contains two separate parts creates an additional challenge, since finding an optimal combination of model variants and parameters can be very difficult. The first part of the model, where LDA or K-means clustering is used for detecting the discourses from the document representations, is also prone to suffer from the limitations that were discussed earlier in relation to LDA. Developing a unified model for discourse change detection that is based on deep neural networks is, therefore, an attractive option, as it allows the backpropagation of the loss through the entire model, and consequently

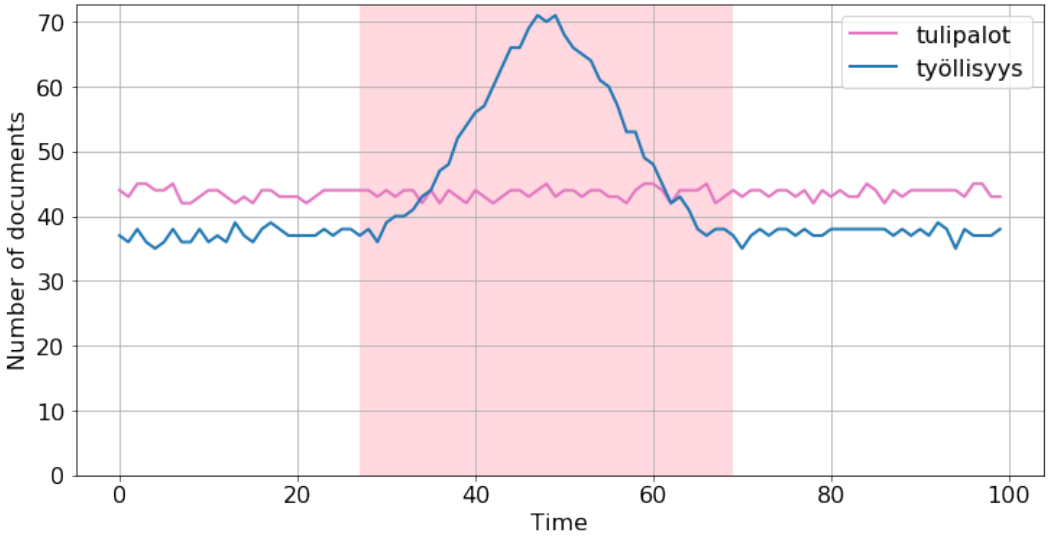


Figure 2.5: Example of the definition of an unstable timepoint. In the image, the area shaded with red (between timepoints 27 and 69) is defined to belong to a non-stable discourse pattern. Image by the author.

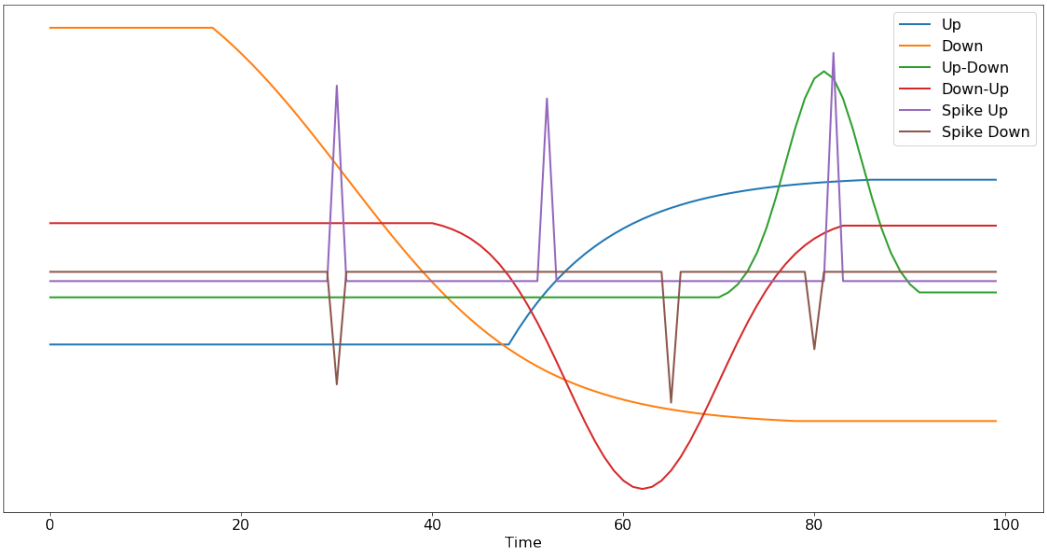


Figure 2.6: Illustration of the six non-stable discourse patterns ('Up', 'Down', 'Up-Down', 'Down-Up', 'Spike Up', 'Spike Down') used in the synthetic datasets. Image by the author.

can be trained in an end-to-end fashion.

After initial attempts, based on a straightforward supervised classification approach using synthetic datasets, to build such a model failed to deliver positive learning results, additional elements were needed in the model and the training process to boost learning. A key feature turned out to be contrastive learning.

3. Contrastive learning

The basic goal of contrastive learning is to make the encoded representations of data samples belonging to the same class more similar to each other, while distancing them from the representations of samples belonging to different classes [14]. This has proven to be a useful approach especially when pretraining deep learning models to perform better in classification-based computer vision tasks. The application of contrastive learning in a self-supervised setting, where it has been used to compensate for the lack of labeled training data, has been especially popular [11].

To take an example from computer vision, typically pretext tasks like color transformations, geometric transformations or jigsaw puzzles (splitting the original image into patches) have been used to create augmented versions of the original image that function as positive samples, forming a category that can be now defined by pseudo-labels [11]. Figure 3.1 illustrates this basic setting.

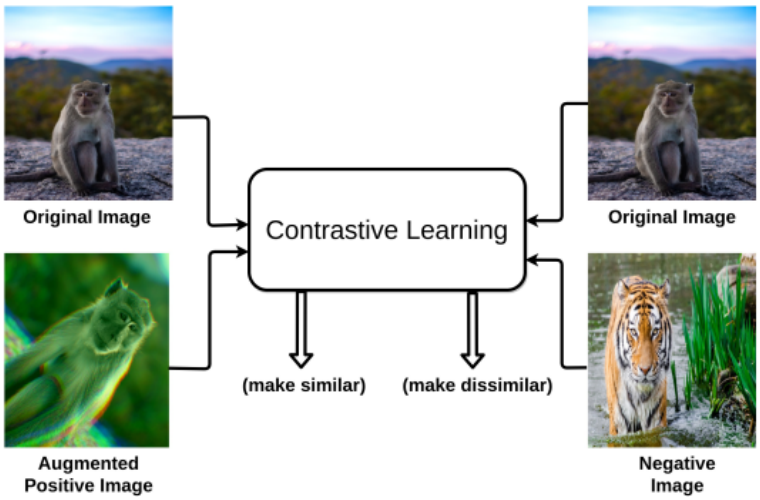


Figure 3.1: The basic idea of contrastive learning, applied here to the computer vision field. Contrastive learning increases the similarity between the encoded representations of the original image and its augmented modification, while making the original image dissimilar to the negative example. Image from Jaiswal et al. [11].

In the NLP framework, contrastive pretext tasks like predicting the neighboring word or sentence, have been used successfully for example in creating word embeddings

(word2vec) and in training language models (BERT) [11]. Different augmentation methods have also been developed for creating modified versions of the original text input in order to obtain positive samples for self-supervised training [17].

Contrastive learning is typically applied when the goal is to pretrain a model in order to improve its performance in some downstream task, like image or text classification [17]. By learning to differentiate the similar and dissimilar samples from each other, the model that is trained using contrastive loss accumulates information in its parameter weights that can be useful when applied to a downstream task. It has been shown that the information learned during contrastive pretraining can generalise beyond the specific data and categories used during training. For instance, pretrained models have been used in downstream tasks where they have been able to predict previously unseen classes (zero-shot learning), and they have also performed better than other models in predicting classes with very few samples in the data (few-shot learning).

This is also a key motivation for implementing contrasive pretraining in the discourse change detection task. The idea is to test whether training the model to differentiate the various discourse patterns (see Figure 2.6 above) used in the synthetic data can help in downstream classification tasks. The goal is both to classify whole datasets based on the discourse patterns that they contain, and to classify individual timepoints based on whether they do or do not belong to a non-stable discourse pattern (see Figure 2.5 above).

As the use of synthetic data makes available the actual labels of these categories, data augmentations and pretext tasks for creating pseudo-labels are not used in the experiments described in the following chapters. Instead, the experiments take advantage of a supervised version of contrastive learning. This means that the data samples having the same class label are pulled closer together in the representation space than those with different class labels [12].

The specific model structure used for achieving this, as well as the actual loss function used in contrastive pretraining, are discussed in detail in Chapter 5. Figure 3.2 illustrates the basic setting, where the representations of datasets containing the same discourse pattern, recognised by their class labels, are made more similar to each other than to representations with other class labels.

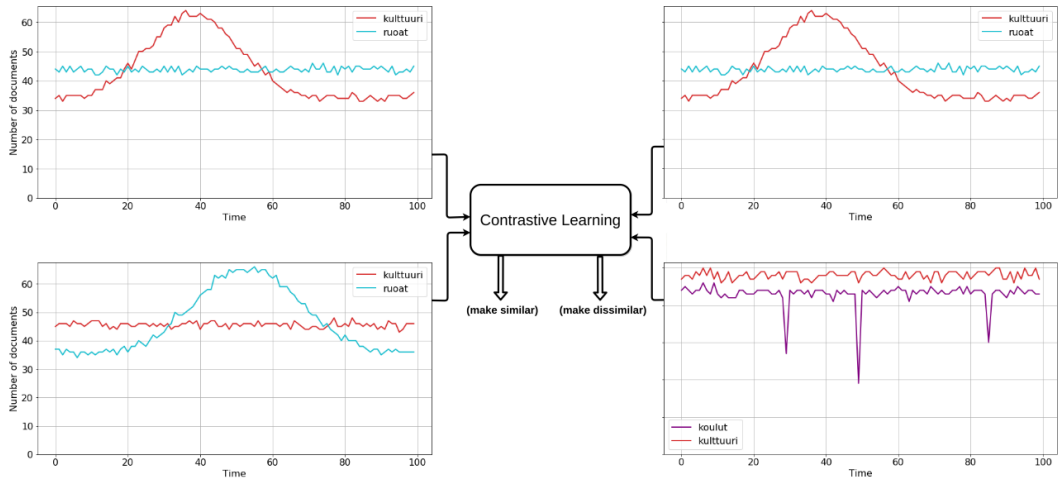


Figure 3.2: Basic idea of contrastive learning applied to discourse pattern recognition. Images of timelines on the left are variations of the same unstable discourse pattern, while the patterns in the timelines on the right are clearly different. Image modified from Jaiswal et al. [11].

4. Data

This chapter describes the data used in the experiments, data preprocessing and the generation of synthetic datasets. Emphasis is put on explaining the process of creating synthetic data, as this is one of the key elements in the experimental setup and necessary for understanding the scope and generalisability of the results. This part of the thesis corresponds in many aspects with the work of Duong et al. [7], but there are also some significant differences which are pointed out in the following sections.

4.1 Dataset

The 'raw' data used for creating synthetic datasets comes from a corpus of Finnish news articles by the Finnish Broadcasting Company YLE[†], and contains in total over 700 000 articles from the years 2011-2018. The articles cover a wide variety of themes, and each article in the corpus includes usually several category labels that describe the specific topics discussed in the article. These category labels are of key importance in enabling the creation of synthetic datasets, a process that is described in detail in Section 4.4 below.

4.2 Data preprocessing

Some basic preprocessing operations were performed to the article data. Stopwords included in the Finnish stopword list of the nltk[‡] library (229 words) were removed from the articles, as well as very rare and very common tokens. Rare tokens were defined to appear in less than 5 documents in the corpus, while tokens that appear in more than 70% of the documents were also removed. In addition, tokens shorter than 3 characters were removed from the corpus texts. The motive behind this preprocessing scheme was to reduce the number of tokens that are likely to have least impact on the semantic meaning of the articles from the point of view of semantic change detection. As the discourses cover a large number of individual documents, extremely rare words

[†]Freely available for academic use at http://urn.fi/urn:nbn:fi:lb-2017070501

[‡]For more information see https://www.nltk.org/

are likely to be of small relevance for their shared content. Similarly, very common tokens are usually words that are not specifically related to the discourses, and removing them can thus help the model to focus on semantic information that is most useful for the task at hand.

4.3 Document embeddings

Text preprocessing is followed by the transformation of the documents into numerical vectors. In natural language processing, the representation of the input text data has a big impact on the performance of the model. LDA and many other topic models represent the input documents in a Bag-of-Words (BoW) form, where only the number of word occurrences is counted and semantic information that is contained in word and sentence order is disregarded [1].

As was discussed above, in later topic model variants there is often an attempt to improve the model output by using a richer representation of the content of the documents. In D-ETM this is achieved with word embeddings [5], while the developers of a recent neural topic model implementation [1] use contextualized document embeddings to improve the coherence of the topics. The general idea behind embeddings is to generate word or sentence representations that are close in vector space to semantically similar words or sentences: this closeness can be measured for example with cosine similarity or Euclidean distance. One common way to generate sentence embeddings is provided by Sentence-BERT (SBERT), which is a modification of the BERT language model, designed specifically to generate sentence embeddings that contain meaningful semantic information [16].

To achieve this, Sentence-BERT uses pre-trained BERT* network that is modified slightly to get the desired output: fixed-sized sentence embeddings for input texts of varying size. Sentence-BERT has achieved good results in tasks that measure the semantic similarity of the sentence embeddings [16]. There are also various pretrained models available for implementations of Sentence-BERT, making it possible to use and fine-tune models that have been already trained with extensive data sets. This can not be done with alternative document embedding methods like doc2vec [13], which was used by Duong et al. [7] in their experiments.

For the above reasons, Sentence-BERT was used for creating the document embeddings that are the input to the neural network model, which is presented in Chap-

*Bidirectional Encoder Representations from Transformers (BERT) is a popular multi-purpose language model based on the transformer architecture. See the original publication by Devlin et al. [4] for more information.

ter 5. Specifically, a general purpose model 'all-distilroberta-v1'* trained with over 1 billion training pairs was used as the pretrained embedding model. Although Sentence-Bert models are able to embed larger sequences than just single sentences, the capacity of the pretrained models is limited to a specific number of tokens. The chosen model is able to process documents with maximum length of 512 words, which is more than most of the other pretrained models.

The quality of the embeddings can be tested by finding documents that are closest (by cosine similarity) to a selected document in the embedding space and checking manually whether they are also closest in semantic meaning. This method was used for an approximate comparison, and the performance of the 'all-distilroberta-v1' model was estimated to be similar or better than that of the other tested models.

After the embedding model was chosen, it was used to transform each article in the news corpus into an embedding vector of 300 dimensions. The 512 word limit set to the input sequence by the model caused some of the articles to be truncated to that length, but since the average length of the articles in the (not preprocessed) corpus is 250 tokens, this limitation was not deemed to be of crucial importance.

4.4 Synthetic data generation

Following the example of Duong et al. [7], articles belonging to 12 large and distinct categories were selected as the basis for synthetic dataset creation. The chosen categories contain 'autot' (cars), 'musiikki' (music), 'luonto' (nature), 'vaalit' (elections), 'taudit' (diseases), 'työllisyys' (employment), 'jääkiekko' (ice-hockey), 'kulttuuri' (culture), 'rikokset' (crimes), 'koulut' (schools), 'tulipalot' (fires) and 'ruoat' (food). Each article was grouped into at least one of these categories, and when an article had more than one of the category labels, it was allocated to each of the relevant categories. The slightly over 200 000 articles belonging to the above mentioned categories were the basis for creating most of the synthetic datasets.

The one exception to this was the dataset variant where the noisiness of the data was increased by adding a random selection of documents from a large number of categories as 'background noise' to the generated unstable discourse pattern. In this case, the categories of the 'noisy' documents were selected from a group where each category contained at least 8000 documents, leaving in total 137 categories from which the documents could be sampled. The purpose of this procedure was to test the performance of the model with more noisy, and therefore more realistic, data, while the size limit guaranteed that the categories contained enough documents for dataset

*More information on the model is available at https://www.sbert.net/docs/pretrained_models.html#.

generation. Figure 4.1 shows as an example a visualisation of a dataset with an unstable discourse pattern and background noise consisting of articles belonging to several news categories.

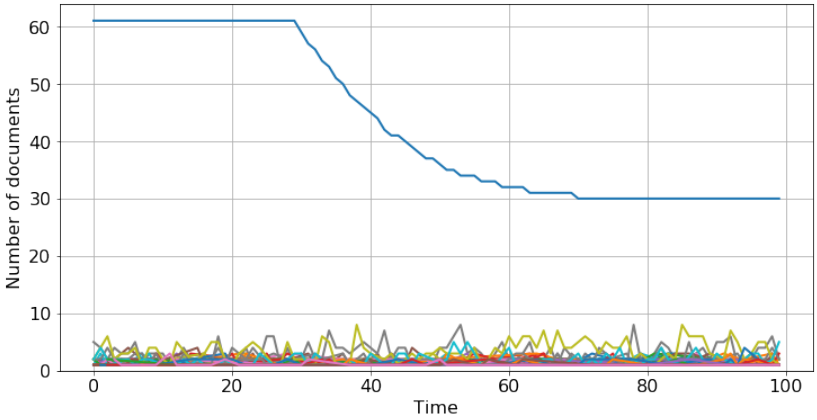


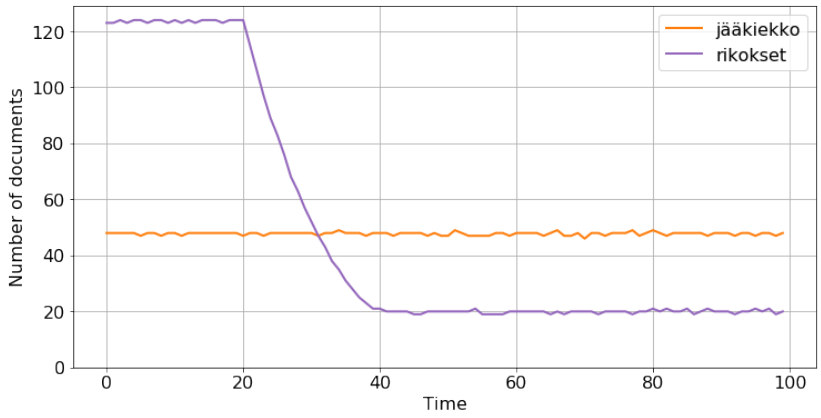
Figure 4.1: Synthetic dataset where the blue line signifies an unstable pattern ('Down') that contains no added noise. The lines with small spikes are 'background noise', consisting of articles that cover a wide variety of news categories, indicated in the plot by the different colors.

Distinct train, validation and test datasets were used for model training and evaluation. In order to emphasize the independence of the test data, five of the above categories were used in creating the test dataset, and the rest were kept for the train and validation datasets.

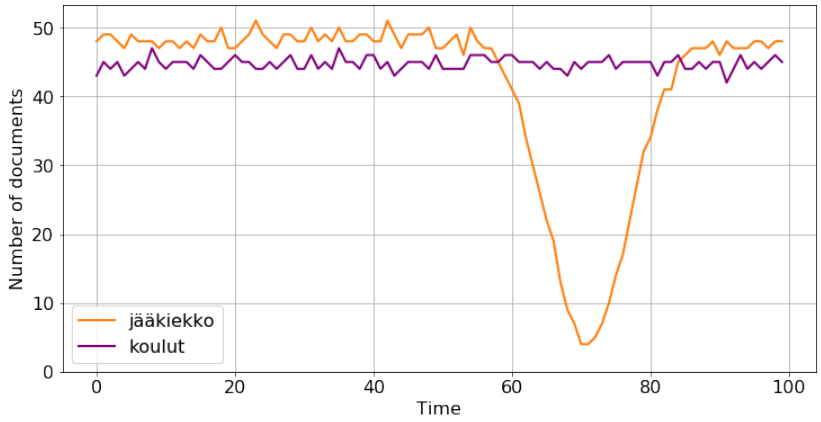
The synthetic datasets were generated using 100 timepoints. Randomly sampled documents from the selected categories were assigned to each timepoint, so that one category followed a specific 'unstable' pattern. In practice, this means that the number of documents belonging to that category at each timepoint is defined by the pattern in question. In most test cases, a simplified setup with only one other category, following a stable pattern with varying amount of noise, was used. By varying the amount of noisiness in the data, it is possible to observe how well the model is able to tolerate complexity, and also to see whether noise has a positive effect on the generalizability of the model in cases where it is overfitting to the training data. Figure 4.2 shows three example datasets with different levels of added noise.

Similarly to Duong et al. [7], sigmoid and Gaussian functions were used for generating the unstable discourse patterns, which are illustrated above in Figure 2.6. Due to its characteristic form, the sigmoid function is well suited for modeling the increasing and decreasing patterns, while the other unstable patterns can be created by varying the mean and standard deviation of the Gaussian function. The pivot points, or the starting and ending timepoints of the unstable pattern, are selected randomly between specified limits, and so are also the change rates of the patterns, the number of spikes in a sample sets in the 'Spike Up' and 'Spike Down' patterns, and

(a) 'Down'-pattern with added noise, using the lowest noise level.



(b) 'Down-Up'-pattern with medium noise level.



(c) 'Spike Down'-pattern with the highest noise level.

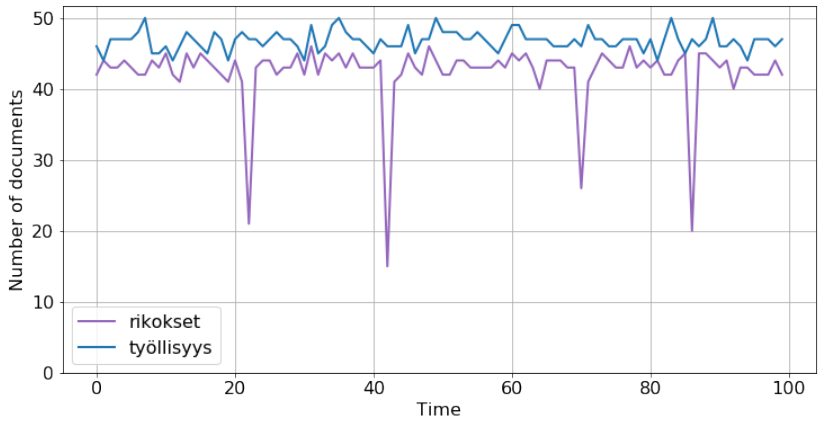


Figure 4.2: Example of three sample sets with different noise levels and unstable discourse patterns.

the number of documents used for each category in a sample set.

Despite the above mentioned random elements in the data creation, the synthetic

datasets are very simple when compared with most real world text corpora. The upside, however, is that this highly controlled method of dataset generation makes it possible to observe and test the model behaviour in a much more nuanced way than would be possible when using unannnotated real world diachronic text corpora. When developing a new model for discourse change detection, it is extremely useful to be able to start with very simple 'base level' data to see whether the model architecture allows it to learn anything that could be relevant for the task at hand. The synthetic data generation model presented above makes this possible, and it is also easy to increase the complexity of the data as the model is developed further. In addition, it should be remembered that at the document level, there is still plenty of ambiguity and noise, as the articles usually belong to more than one category. Also inside each category there can be significant variation in vocabulary and style.

When developing the neural network model, training and testing was done with a few variations of very simple data generation procedure. The number of timepoints in a sample dataset was set to 100, and each dataset was constrained to contain only two patterns: these could be either a pair of one unstable and one stable pattern, or alternatively two stable patterns. Each pattern in a dataset contained articles from only one of the assigned categories, with the exception of the 'background noise' data generation method described earlier. These synthetic data variations used in the experiments are summarized in Table 4.1.

Dataset type	No. patterns	No. categories	No. documents in category	Amount of noise in the patterns
Basic	2	2	4000-5000	No added noise
Noise 1	2	2	4000-5000	Gaussian noise with std 0.0001 is added to the document frequency pattern
Noise 2	2	2	4000-5000	Gaussian noise with std 0.0002 is added to the document frequency pattern
Noise 3	2	2	4000-5000	Gaussian noise with std 0.0003 is added to the document frequency pattern
Random	2	≤ 137	1-5000	No added noise

Table 4.1: Types of synthetic datasets used in the experiments.

5. Contrastive pretraining

This chapter presents the deep neural network model, which is trained to differentiate the synthetically generated discourse patterns from each other. The pretrained model is then used in downstream tasks for detecting the start and end points of unstable discourse patterns and also for classifying timepoints based on whether they belong to such pattern. The results of these experiments are discussed in detail in Chapter 8.

Section 5.1 presents the main elements of the network architecture, including the key deep learning techniques that are implemented in the model, Self-attention and Long short-term memory (LSTM). Section 5.2 presents the different variations of the model used in the experiments.

5.1 Model architecture

As mentioned above, the model presented in this thesis is not based on the probabilistic generative approach that is commonly used with topic models. Instead, the aim has been to design a deep neural network architecture that can be trained in an end-to-end fashion to recognise unstable discourses in input data. The key elements of the model and its functioning are described here.

After the data preprocessing and synthetic dataset generation have been accomplished, the input data for the model consists of sample datasets. Each dataset contains 100 timepoint,s and each of the timepoints consists of varying number of documents belonging to different categories. Thanks to the fact that the data is created synthetically, we also know what category each document belongs to, what type of discourse pattern each dataset contains, and, if the dataset has an unstable discourse pattern, we know whether a timepoint is part of this pattern or not. These are the three types of labels available to be used in model training and evaluation.

The general structure of the contrastive model is shown in Figure 5.1. The input to the model consists of the document vectors and the labels indicating the type of discourse pattern present in the dataset. One input data sample contains the document vectors that belong to one timepoint. Since the number of documents varies between timepoints and the input shape for the neural network needs to be constant, padding is

used to add zero vectors to timepoints when needed. If n is the number of documents (including possible paddings) in a timepoint and $m = 300$ is the size of the document embedding vector, the input to the first part of the model, the Transformer Encoder layers, is a matrix of size $n \times m$.

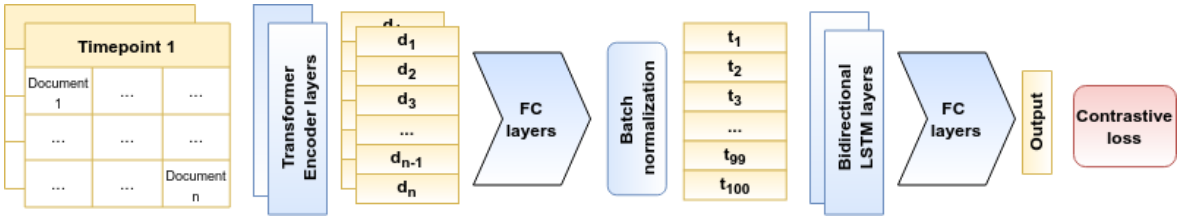


Figure 5.1: Architecture of the model used for contrastive pretraining. The input consists of document respresentations that belong to one timepoint in the sample dataset. The output of the transformer encoder layers, $d_1, ..., d_n$, has the same dimensions as the input. $t_1, ..., t_{100}$ are the timepoint level representations from the entire sample set, used as an input to the LSTM layers. The final output of the model is a vector of 100 dimensions that corresponds to an input sample set with 100 timepoints.

An important feature of the model structure is the way it allows to extract outputs from specific intermediate layers. The output of the transformer encoder layers, indicated in Figure 5.1 as $d_1, ..., d_n$, preserves the dimensions of the input data, but has already been transformed by the trained model. These vectors are referred to in the following as 'document representations', and they are used for exploring the impacts of model training at the document level. The 'timepoint representations' $t_1, ..., t_{100}$ are the result of further processing of the data by fully connected layers, followed by batch normalization. They are used for exploring the impacts of model training at the timepoint level, and provide also the input for the unsupervised classification and pivot point detection models used in the downstream tasks.

The model output, a 100 dimensional vector, is a representation of an entire dataset. Even though the output dimensions equal the number of timepoints in the input dataset, the learning goal defined by the loss function does not require the output vectors to contain information on individual timepoints. The output vectors are used for estimating the contrastive loss, and it can thus be expected that they contain information that relates directly to the contrastive learning task. The choice of dimensions, however, allows for the model to embed also additional information to the outputs that could be useful for the downstream classification tasks.

The rest of this chapter describes in detail the key elements of the model, starting from the Transformer Encoder layers and ending with the contrastive loss function. The final section also introduces the different variations of the model that are used in the experiments.

5.1.1 Transformer Encoder

When attempting to extract information on a discourse pattern from time series data with embedded documents, it would be useful to bring the representations of similar documents at each time point closer to each other. In the model presented above in Figure 5.1, this is achieved by using a Transformer Encoder layers that implement multi-head self-attention.

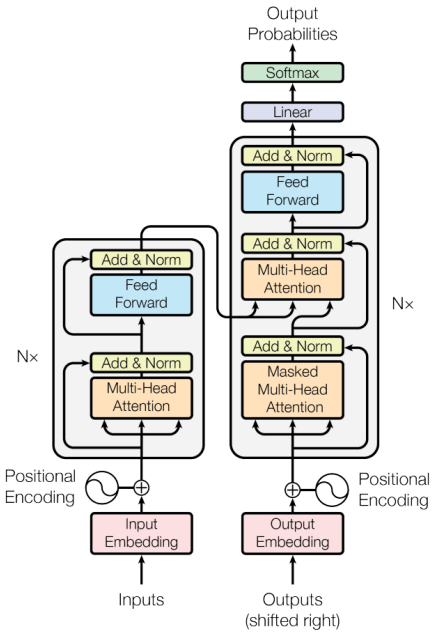


Figure 5.2: Architecture of the transformer model from Vaswani et al. [22]. Left part of the model is the encoder, and the right part the decoder.

This encoder model, illustrated on the left in Figure 5.2 and first introduced as part of the Transformer architecture in Vaswani et al. [22], passes the input through N consecutive layers, each containing multi-head self-attention and a fully connected feed-forward network.

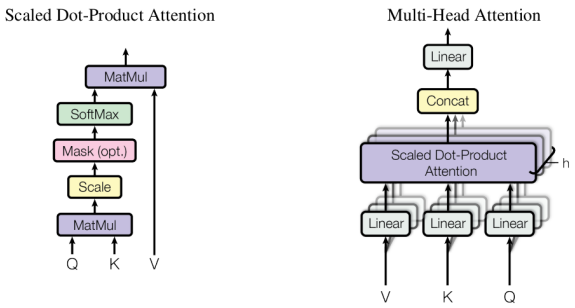


Figure 5.3: Scaled dot-product attention and Multi-head attention from Vaswani et al. [22]

The key element here is the attention function. It receives as input three matrices,

queries Q, keys K and values V, which in self-attention are formed by multiplying the input data X with weight matrices, so that $Q = W_Q X$, $K = W_K X$ and $V = W_V X$ [22]. In the model presented here, X is a matrix formed by the document vectors belonging to a single timepoint. The attention process starts by calculating a dot product between the queries and keys and scaling it with a square root of their dimension d_k, which corresponds here with the size of the document embeddings e. Then the softmax function is used to form a score for each document, which is then applied as a weight to the value vectors:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}V\right)$$

The weighted value vectors are then summed together to form the output of self attention for each position in the input [22]. This allows the model to encode dependencies between different positions in the input sequence into their representations. With multi-head self-attention, additional representation sub-spaces are introduced by implementing the attention function simultaneously with multiple versions of the queries, keys and values. These are all multiplied by weight matrices specific for each attention head, and in the end the results of each head are concatenated together to form the output for the given positions.

Attention is a flexible and efficient way to assign importance to relevant parts of the input sequence by the weighting mechanism described above, and it is a common element in neural models used for a variety of NLP tasks [8]. Whereas in tasks like machine-translation the individual input elements are usually word embeddings, in the model used here for discourse change detection they are embedded documents. The general aim, however, is the same: to assign weights to the different inputs based on the connections between them that are deemed relevant for the specific task.

PyTorch library was used for building the neural network model, and the Transformer encoder layers were realized using the PyTorch 'TransformerEncoderLayer'* implementation. The key parameters of this architecture, the number of the transformer encoder layers used and the number of attention heads implemented in each layer, were chosen as a compromise between performance gains and computational costs. After some initial testing with different values, in the final implementation four transformer encoder layers and two attention heads were used.

*More information about these and the other PyTorch elements mentioned in the text can be found at https://pytorch.org/docs/stable/nn.html

5.1.2 Batch normalisation

The transformer encoder layers do not transform the dimensions of the timepoint data ($n \times m$), which makes it possible to still observe the outputs of this part of the model at the document level. This possibility is explored further in the experiments described in Chapter 8, where the impacts of contrastive training at the document level are discussed.

After the transformer encoder layers, the dimensionality of the data is reduced by using two fully connected layers, the output of which is passed through ReLU (Rectified Linear Unit) activation function

$$f(x) = \max(0, x),$$

which outputs 0 if the input $x \leq 0$, and x otherwise. ReLU is also used as the activation function after the first linear layer element in each of the transformer encoder layers.

The output of the linear layers is then passed through batch normalization. The input data enters the network in batches, which contain a defined amount of individual data samples. The larger the batch size, the more processing power is required. On the other hand, the gradient update of the model parameters is estimated based on the samples in the batch, so the more they are, closer to the true value the estimate will be. Based on tests done with different batch sizes, the one chosen for the final model was 30 sample sets, or 3000 timepoints.

Batch normalization is a method developed to counter what its developers, Ioffe and Szegedy [10], term internal covariate shift. This refers to the fact that during model training, the weights of the different layers in a deep neural network are constantly changing, which keeps also changing the distribution of the input data, especially after several layers. The changing distribution of the inputs requires constant adaptation from the hidden layers, which can slow down learning and require complicated tuning of the hyperparameters, like the learning rate.

The solution proposed by Ioffe and Szegedy [10] is to normalize the samples in a batch so that the inputs to the following layer has a stable distribution. If $\mathbf{H} \in \mathbb{R}^{y \times z}$, where y is the batch size and z is the size of the output of the previous layer, is a matrix containing the data to be normalized, μ is a vector containing the columnwise means of $\mathbf{H}$, and σ is a vector with the columnwise standard deviations of $\mathbf{H}$, then

$$\mathbf{H}' = \frac{\mathbf{H} - \mu}{\sigma}$$

gives the normalized inputs to the following layer [9]. In addition, to maintain the expressive power of the model, learned parameters γ and β are added to the normalization procedure

$$\gamma \mathbf{H}' + \beta,$$

so that now β defines the mean and γ the standard deviation of the normalized values [9].

In the experiments done for this thesis, the use of batch normalization during the contrastive training of the model was able to stabilize learning, so that a PyTorch implementation of the function ('BatchNorm1d') was used also in the final version of the network.

5.1.3 Long Short-Term Memory (LSTM)

The data representation after the fully connected layers is a vector with 300 dimensions for each timepoint in the original dataset, so that the size of the timepoint representation now equals the size of the document representation in the input data. The fact that this intermediate part of the model outputs single vector representations for each timepoint is an intentional design choice that makes it possible to observe the impacts of contrastive learning at the timepoint level and also facilitates the use of the timepoint vectors in downstream classification tasks.

The aim of the architectural choices described above is to enable the model to pass information that is relevant for recognising unstable discourse patterns from the documents to the timepoint representations. But as the contrastive model is trained to learn to distinguish different discourse patterns, which occur at the level of individual sample sets, a further level of abstraction is needed to obtain desired results. As the discourse patterns in a time series data by nature cross over multiple time points, in order to pass forward information that is relevant for discourse pattern recognition, the time point level representations need to be tied closer together as a sequence.

There are different ways to achieve this task, which is not unlike the one faced previously with the document level representations. The method applied here is called Long short-term memory (LSTM), which is a specific type of recurrent neural network (RNN) architecture [9]. Recurrent neural networks are well suited for processing sequential data, since their output at each step is based not only on the current input, but also on contextual information from previous steps that is preserved in the 'memory', or hidden state. The basic version of RNN, however, suffers from the vanishing and exploding gradient problems, which, among other things, limits the range of the contextual information that can be effectively maintained in the hidden state.

The LSTM structure offers a way to mitigate these problems by introducing gated memory cells, which provide a more nuanced way to control the content and impact of the state at each time step. In the LSTM model presented in Figure 5.4, x_t is the

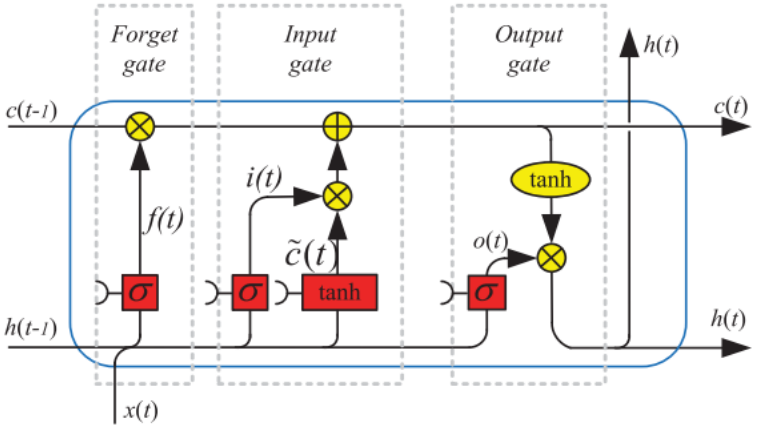


Figure 5.4: LSTM cell architecture. Image from Yu et al. [24]

input to the cell at time t, h_{t-1} is the hidden state information from the previous cell, and c_{t-1} is the cell state or memory passed also from the previous cell. The Forget gate controls which information will not be kept in the cell state by using the function

$$f(t) = \sigma\left(W_{fh}h_{t-1} + W_{fx}x_t + b_f\right),$$

where W_{fh} and W_{fh} are the weights applied to the hidden state h_{t-1} and current input x_t, b_f is the bias term and σ is the sigmoid function, which outputs values between 0 and 1. The Input gate, on the other hand, decides how much new information is added to the cell state based on h_{t-1} and x_t by performing pointwise multiplication between

$$i_t = \sigma\left(W_{ih}h_{t-1} + W_{ix}x_t + b_i\right)$$

and

$$\tilde{c}_t = tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}},$$

where $tanh$ is the hyperbolic tangent function

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

and the results are summed pointwise to the old cell state, resulting into the new cell state

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t.$$

Finally, the Output gate controls how much of this cell state is included in the output

$$h_t = o_t \cdot tanh(c_t)$$

where

$$o_t = \sigma\left(W_{oh} h_{t-1} + W_{ox} x_t + b_o\right).$$

In the standard LSTM model, the flow of information is one-directional, so that only the earlier steps in the sequence of data can impact the current output. In the discourse change detection setting this is a problem, as the representation of the current timepoint should be related to both past and future timepoints. A solution is provided by bidirectional LSTM, which processes the input sequence starting from both ends, so that at each time step between the end points contextual information from both directions is available [24]. This allows the timepoint representations to encode relevant information from other points in the time series, so that they can better reflect the discourse dynamics in the data. Figure 5.5 presents a model of a multilayer bidirectional LSTM network, where the final output contains the hidden state information from the last forward layer $\overrightarrow{h}_t$ and the last backward layer.

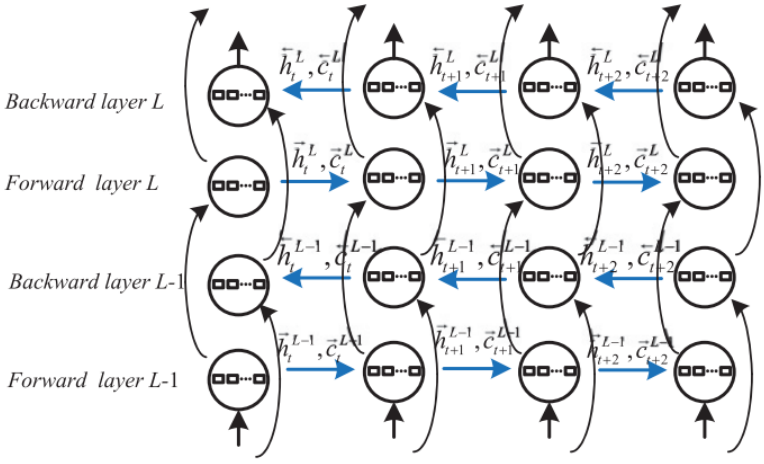


Figure 5.5: Model of a multilayer bidirectional LSTM presenting the connections between the forward and backward layers. Image from Yu et al. [24]

The key parameters for the LSTM model are the number of layers used and the size of the hidden state h. Using multiple layers and large hidden state can bring better learning results, but, especially when using bidirectional LSTM, adding new layers also multiplies the computational cost. After testing different options as part of this thesis work, a balance between these two considerations was reached with hidden state size 10 and two bidirectional LSTM layers.

5.1.4 Loss function

The input to the LSTM layers is a sequence of timepoint vectors corresponding to the length of the sample set, which is here 100 timepoints. The output consists of the hidden state information of the last forward and backward layers, in both cases of size 10, for all of the timepoints t. The model uses two feedforward layers with ReLU activations to further reduce the dimension of this data into a 100 dimensional vector representing one sample set of 100 timepoints. This, then, is the final level of abstraction in the hierarchy of data representations, which started from the 300 dimensional document vectors and was followed by the 300 dimensional timepoint vectors.

The basic aim of the training is to incentivize the model to encode information relevant for discourse pattern recognition into the final output vector. In order to achieve this goal, a proper loss function is required. As was mentioned in Chapter 3, a supervised contrastive loss function is used for guiding the model as it learns to bring the representations of data samples belonging to the same class closer to each other, while increasing the distancing between them and the representations of samples belonging to different classes. The classes that are explicitly used in the training of the network are the discourse patterns, and the loss function receives as its input the final output of the model together with the corresponding pattern label.

The idea behind the use of contrastive pretraining is that a) the results of contrastive training would also be visible in the document and timepoint representations, besides the output vectors, and b) that contrastive pretraining using the pattern labels could also help the model perform downstream classification tasks where timepoint vectors are classified based on the timepoint labels.

To achieve these goals, a loss function

$$L^{sup} = \sum_{i \in I} L_i^{sup} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} log \frac{exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a / \tau)}$$

introduced in Khosla et al. [12] as SupCon Loss, is used. Here z_i defines the anchor, or an output vector that belongs to a specific category based on its pattern label. $P(i)$ is the group of positive pairs to z_i in the input batch, and $A(i)$ the group of negative pairs, or all output vectors in the batch belonging to different classes than z_i.

The numerator of the SupCon loss function is an exponent of the dot product of the positive pair of vectors, scaled by the temperature parameter τ, and the denominator is a sum over the exponents of the scaled dot products between the anchor z_i and the negative pairs $z_a,\ a \in A(i)$ [12].

Unlike many of the earlier contrastive loss functions, SupCon loss is able to calculate the loss also in cases when there are more than one pair of samples in the batch

belonging to the same positive class. This is achieved by summing the loss over the samples i, so that each one in turn is the anchor, and the loss calculation is averaged over all of the positive pairs, $|P(i)|$ being the cardinality of $P(i)$. [12]

The SupCon loss function achieves excellent results when applied to image classification tasks by Khosla et al. [12], and it also performed well when compared with some alternative contrastive loss functions in the discourse detection pretraining task. In this thesis work, an implementation of the SupCon loss available in the Pytorch Metric Learning-library* was used in the model, with the temperature parameter τ set to 0.5 after some comparative tests.

After the loss has been calculated, it is propagated through the model using gradient descent, and the optimization algorithm that is applied in the model is Stochastic Gradient Descent (SGD). When θ are the model parameters, at each iteration t the gradient

$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}),$$

is calculated, $f_t(\theta_{t-1})$ signifying the loss averaged over the training data batch. The parameter update

$$\theta_t \leftarrow \theta_{t-1} - \gamma \; g_t$$

then weights the new gradient based on the learning rate parameter γ. The learning rate $\gamma = 0.01$ proved to be suitable for both the contrastive learning and classification tasks, and it was consequently applied to all of the neural network models. Momentum was not used with SGD, and other optimization algorithms were also not tested in a systematic way. Fine-tuning of the model was, to an extent, seen as a secondary concern when compared to the primary task of ensuring its basic functioning.

5.2 Model variations

Minor variations to the above described data processing pipeline inside the model were also tested during the contrastive prelearning process. The basic idea was to enrich the timepoint representations, used as input to the LSTM layers, with information on the neighboring timepoints, and to find out whether this could help the model to find relevant connections in the data across broader time spectrum.

Enrichment of timepoint representations is illustrated in Figure 5.6. A sliding window, which in the image has size 3, was used to create for each timepoint vector in

*For more information, see https://kevinmusgrave.github.io/pytorch-metric-learning/losses/#supconloss

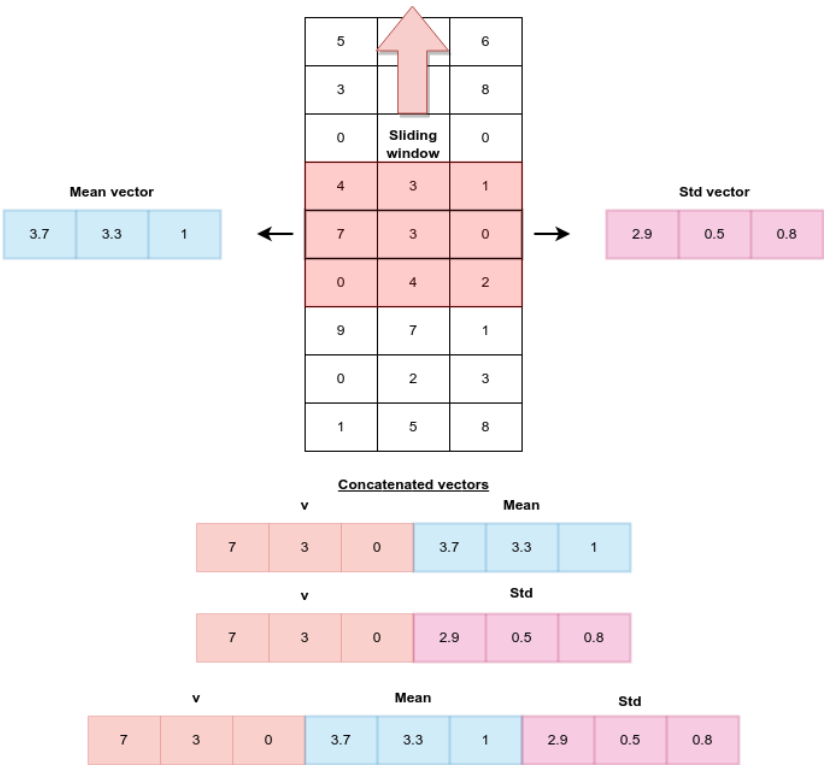


Figure 5.6: Illustration of the method used for enriching the timepoint representations with mean and standard deviations of timepoints in a sliding window.

the time series an accompanying 'smoothing' vector. This contained either the mean or standard deviation calculated over the columns of the vectors covered by the sliding window. The size of the sliding window that was implemented in the experiments is 9 timepoints, and zero padding was used at the beginning and end of the time series. Other values, like the vector mean, could have also been used for this purpose. The impacts on the values of the smoothing vector and, consecutively, model outputs, of the different padding options were not, however, compared in the experiments.

After the smoothing vectors were calculated for the whole time series, they were concatenated with the original vector (**v** in the image), and the extended vector was used as an input to the LSTM layers. The model variants, listed in Table 5.1, are based on different combinations of these vectors that were tested during contrastive pretraining.

Model type	Smoothing	Window size	Padding	LSTM input vector size
Basic	No smoothing.	-	-	300
Mean smoothing	Mean	9	Zero padding	600
Std smoothing	Standard deviation	9	Zero padding	600
Double smoothing	Mean and standard deviation	9	Zero padding	900

Table 5.1: Contrastive model variants.

6. Change detection

The aim of contrastive pretraining in the context of this thesis is to teach the neural network model to encode information in the data that can facilitate discourse change detection. This is formulated as both as a supervised and an unsupervised classification problem, which use different methods for achieving the same goal: classifying timepoints based on their affiliation to an unstable discourse pattern. By comparing the results achieved using these different approaches it is then possible to estimate the potential of unsupervised methods, which generalise better to more realistic settings where labeled data is not always available.

This chapter discusses in detail the specific settings and methods used in these classification tasks. Section 6.1 focuses on supervised classification of timepoints and datasets, and Section 6.2 describes the methods used for unsupervised timepoint classification and pivot point detection.

6.1 Supervised classification: model architecture and losses

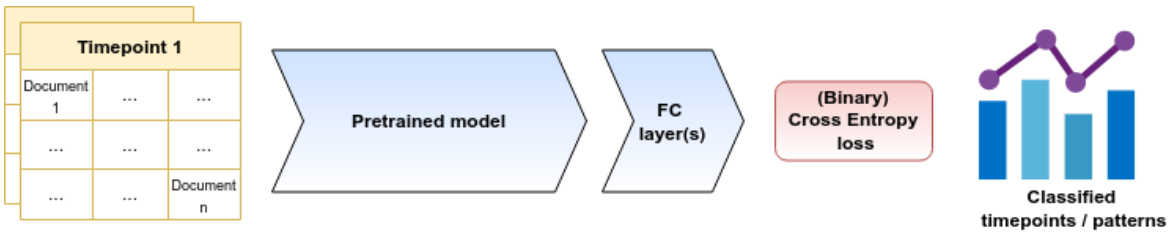


Figure 6.1: Architecture of the model used for downstream supervised classification tasks. The 'Pretrained model' component contains the parameter values of the contrastive model, learned during the pretranining. Their output is passed to two fully connected layers, which are trained to minimize classification loss.

The supervised classification task takes direct advantage of the information the model has gained during contrastive pretraining. Following the example of Khosla et al. [12], the model for the downstream classification task is built by adding a classifier

element on top of the pretrained model. This consists of two linear layers, which transform the output to proper size, and a loss function suitable for a classification task. During the training of the supervised network, the gradients are propagated only through this classifier element, while the parameter weights of the pretrained network are left 'frozen': this way the knowledge from pretraining is directly transferred to the new task.

The same method is applied to two types of classification settings, and the loss function that is used depends on the specific task. When the outputs of the network are classified based on the discourse pattern of the sample dataset, it is a multi-class classification problem where the commonly used loss function is Cross-Entropy loss..

If C defines the number of classes in the data and N is the batch size, then the input to the loss function, x, has the size $N \times C$ and consists of unnormalized scores for each of the classes. When y are the target labels, the cross entropy loss for the batch is

$$L_{CE} = \frac{1}{N} \sum_{n=1}^{N} -log \frac{exp(x_{n,y_n})}{\sum_{c=1}^{C} exp(x_{n,c})},$$

where x_{n,y_n} is the score corresponding to the correct class label for sample n, and the denominator contains the sum of the exponents of the scores given to all of the classes in the sample n by the classifier. This definition follows the PyTorch implementation, which is used also in the experiments.

In the second variation of the model, the task is to classify the model output based on the timepoint labels, which define whether a timepoint in the sample set is part of a stable or an unstable discourse pattern. This distinction was illustrated above in Figure 2.6.

Here there are only two classes, so the classification task requires a loss function suitable for the binary context. The typical choice is binary cross entropy loss, and a PyTorch implementation 'BCEWithLogitsLoss' is used here. The inputs x are raw scores produced by the network, which are mapped to the range $[0, 1]$ using the sigmoid function σ. The output of the linear layers is a vector with 100 dimensions per each sample set of 100 timepoints. Each element in the vector, after being transformed by the sigmoid function, is a probability for the corresponding timepoint belonging to an unstable discourse pattern.

One notable feature of the sample datasets is the imbalance between the two timepoint labels: in most of the discourse patterns described above, the unstable pattern covers only a small part of the entire timeline, so that the 'stable' category dominates in the data. As the primary interest is in identifying the timepoints that belong to an unstable pattern, this can cause problems for learning if it is not taken into account

in the loss function. A simple way to correct the imbalance is to use an extra weight term p when calculating the loss for the positive class.

When D is the number of elements in the output vector and N is the batch size,

$$L_{BCEL} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{D}\sum_{d=1}^{D} - (p\ y_{n,d} \cdot log\ \sigma(x_{n,d}) + (1 - y_{n,d}) \cdot log(1 - \sigma(x_{n,d})))\right)$$

defines the binary cross entropy loss for the batch. The value of the label $y_{n,d}$ is either 0 or 1, and the closer the predicted probability $x_{n,d}$ is to this, the smaller the loss. The weight term p is used for adding emphasis to the correct classification of the unstable timepoints, which have the label value 1. The guideline* is to calculate value of p based on the actual or estimated class proportions, so that if a sample set has 80 stable and 20 unstable timepoints, $p = \frac{80}{20} = 4$. After some testing, $p = 3$ was chosen as the weight value used in the experiments.

6.2 Unsupervised classification

In addition to supervised classification, unsupervised classification methods were also tested in the experiments. The term 'unsupervised', however, needs to be justified, since these methods also depend on the preceding supervised contrastive pretraining. Contrastive training, however, had a different training goal, defined by the loss function, and discourse level information was used in the model training instead of timepoint labels. Therefore, in order to contrast this approach from the fully supervised method described above, it is referred to as unsupervised classification in the following sections.

The goal of this approach was to find out whether contrastive pretraining with the discourse pattern labels could help in identifying timepoints belonging to an unstable pattern even without knowledge of the timepoint labels. The methods presented below take advantage of the timepoint vectors produced at an intermediate stage by the contrastive model, marked with $t_1, ..., t_m$ in Figure 5.1. The assumption, which is tested in the experiments, is that the pretrained model has encoded information into the timepoint representations that can be used for distinguishing timepoints belonging to an unstable discourse pattern from those belonging to a stable pattern.

6.2.1 Unsupervised timepoint classification

The unsupervised timepoint classification method proposed here takes advantage of statistical significance testing to assess whether the shifts in distance between consecutive timepoint vectors in the time series are big enough to be statistically significant.

*See https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

The method, then, focuses on detecting changes at the level of individual timepoints. The key idea is to generate random variations of the timeline by resampling the order of the timepoint vectors, and then measure the average distances between the resampled timepoints and their neighbors. P-value is used to assess whether the differences in the measured distances between timepoints in the original data and the generated variations can be considered significant. If the obtained p-value is below a set threshold (0.05), the timepoint is assigned to an unstable discourse pattern, otherwise it is considered to belong to a stable part of the timeline. One example of the use of this technique in a somewhat similar context is presented in Liu et al. [15], where permutation tests are performed to assess the significance of semantic change in word meanings between different timepoints.

At a more detailed level, the classification procedure is the following. First the data is processed by the pretrained contrastive model, and the timepoint vectors, which are an intermediate product of the original model, are used for the classification task. The goal is to take advantage of the information that the model has encoded in the vectors when performing classification.

Each sample set of 100 timepoints is represented by one hundred 300 dimensional timepoint vectors. Calculating the cosine distance* between neighboring timepoint vectors $\mathbf{a}$ and $\mathbf{b}$

$$D_C(\mathbf{a}, \mathbf{b}) = 1 - \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||_2 ||\mathbf{b}||_2},$$

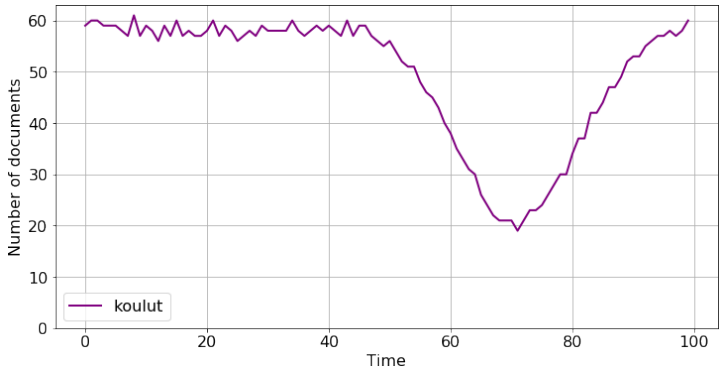
where the denominator defines the product of Euclidean norms $\sqrt{\mathbf{a} \cdot \mathbf{a}}\sqrt{\mathbf{b} \cdot \mathbf{b}}$, produces a timeline on pairwise distances when performed to each timepoint. Figure 6.2 shows an example of such a timeline, where the shaded area marks the actual start and end points of an unstable discourse pattern (here a 'Down-up'-pattern). We can see that the distances between the vectors are clearly higher, and fluctuate more, in the time period belonging to the unstable pattern.

Pairwise distance measures are, however, susceptible to high random variation that can cause problems in the classification stage. Therefore a smoothed version of the distance timeline is generated by giving each timepoint a value based on the distance between the mean values of n vectors on both sides of the timepoint. Mean padding is used for adding n vectors to both ends of the timeline, so that all 100 timepoints can be included.

Figure 6.3 shows a smoothed version of the timeline, where the size of the smooth-

*Cosine distance = 1 - cosine similarity, so that zero cosine distance equals cosine similarity value one, which signifies complete similarity. Values of cosine similarity are in the interval $[-1, 1]$, and values of cosine distance in the interval $[0, 2]$.

(a) 'Down-Up'-pattern with added noise.



(b) A timeline of pairwise cosine distances between timepoint vectors.

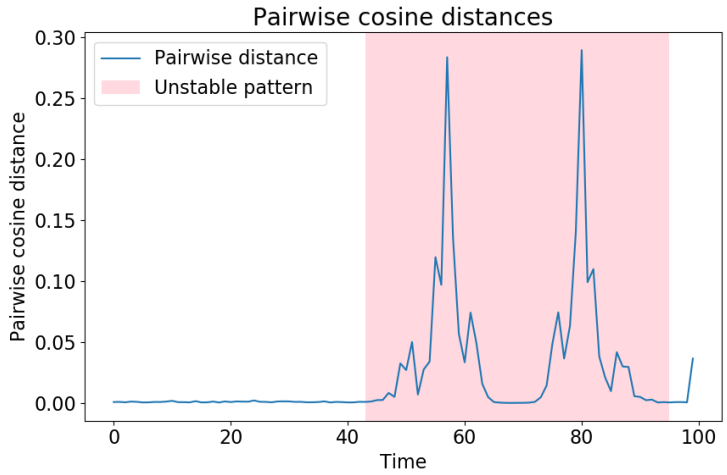


Figure 6.2: The upper image shows an example of a 'Down-Up'-pattern, and the lower image presents a timeline of distances between processed vectors from a dataset with similar unstable pattern. The location of the unstable pattern in the timeline is indicated here with the area shaded in red.

ing window $n = 10$. Now the general form of the discourse pattern can already be clearly observed from the shape of the line, where the two peaks reflect the fall and rise of the document frequency, and the valley in between corresponds with the low point in the 'Down-Up'-pattern.

This smoothed version of the timeline of vector distances is used as the basis for the permutation tests. The hypothesis that is tested states that the distance between timepoints does not equal the average obtained from random permutations, while null hypothesis claims that there is no difference between the random and actual distances.

The Fisher-Pitman permutation test can be used for estimating the distribution of the test statistic by calculating it for all possible permutations of the available data [6]. As each sample dataset contains 100 timepoint vectors, this would mean calculating the averaged cosine distances for all 100! groupings of the 100 vectors. This approach is obviously not feasible, so instead an approximate method is used. A defined number

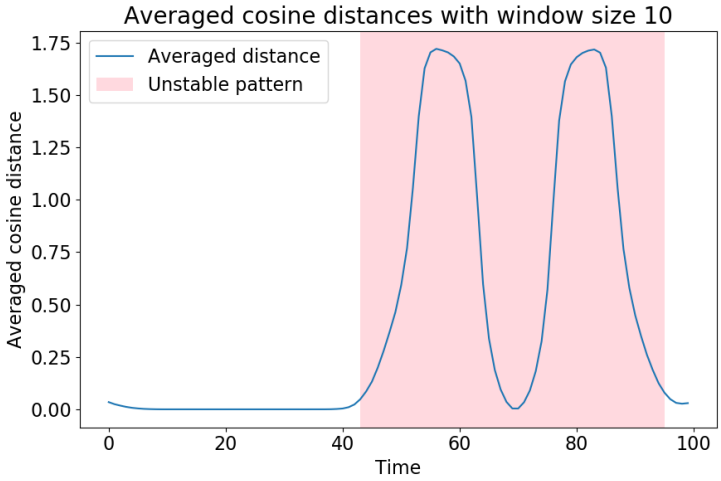


Figure 6.3: A timeline of averaged cosine distances between timepoint vectors.

of sample permutations are drawn from the group of all possible permutations, and the (statistically weaker) conclusions are based on this approximation [6].

The permutations of the timeline are generated by randomly shuffling the order of the timepoint vectors, and the smoothed timelines based on these permutations are compared with the original timeline. Whether the null hypothesis is rejected or not depends on the p-value for each time point t, defined here as the probability of obtaining averaged distances in the permutations of t at least as extreme as the distance actually observed in the original timeline. If n_t is the number of instances where the permutation value at timepoint t is larger or equal to the distance at t in the true timeline, and n_p is the total number of permutations, then the p-value for timepoint t is

$$p_t = \frac{n_t}{n_p}$$

Repeating this calculation for all timepoints t in the sample set produces a timeline of p-values. Now the null hypothesis can be tested by comparing the p-value at t with a significance threshold: the smaller the p-value is, the more there are grounds for rejecting the null hypothesis. How small exactly the p-value has to be depends on the chosen significance level α. Here, a commonly used value 0.05 is applied, so that the null hypothesis is rejected when the p-value is smaller than $\alpha = 0.05$. Unsupervised timepoint classification is now performed by assigning all of the timepoints for which the null hypothesis is rejected into an unstable discourse pattern.

Figure 6.4 shows the p-value timeline for the same sample dataset with the 'Down-Up' pattern. In the image the horizontal dotted line defines the p-value threshold 0.05. Clearly, in this case the timepoints where the p-value falls below the threshold

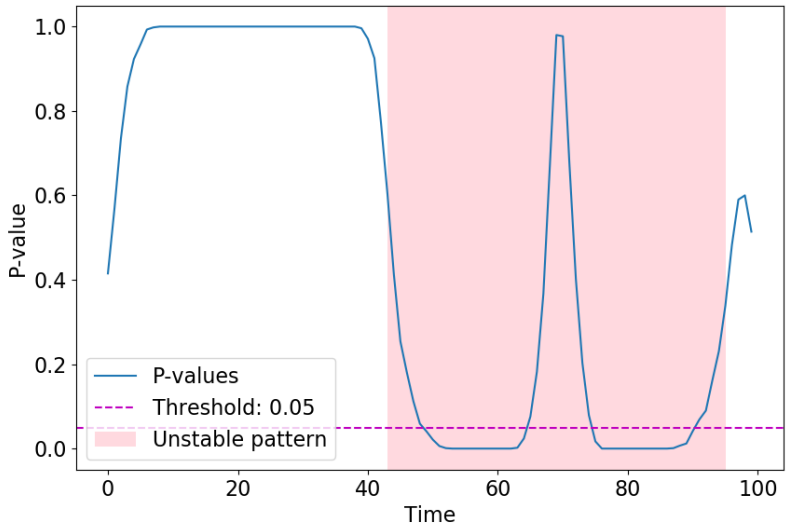


Figure 6.4: A timeline of p-values, with the horizontal dotted line showing the threshold value 0.05.

level correspond well with the timepoints that actually belong to the unstable pattern, marked by the area shaded in red.

6.2.2 Unsupervised pivot point detection

The distance and p-value timelines created for the timepoint classification task can also be used for another unsupervised downstream task, namely pivot or change point detection. The aim here is to detect the start and end points of an unstable discourse pattern, so it is in practice an alternative approach for detecting the location of timepoints belonging to an unstable discourse pattern.

Pivot point detection is a much studied problem especially in statistics and signal processing, and multiple methods and algorithms have been developed for solving the task [20]. Here a method called window-sliding algorithm* is applied to the timeline of averaged cosine distances in order to. The algorithm is based on a function

$$d(y_{a..t}, y_{t..b}) = c(y_{a..b}) - c(y_{a..t}) - c(y_{t..b}), \ (1 \leq a < t < b \leq T),$$

which uses two windows, $y_{a...t}$ and $y_{t...b}$, that slide along the signal $y = \{y_1, ..., y_T\}$, and a cost function $c(\cdot)$ is applied to measure the discrepancy d between the sub-signals represented by the two windows [20].

The size of the window should be a compromise between a value small enough to capture relevant local changes in the timeline, and at the same time big enough to

*More information on the ruptures-library implementation of the algorithm, which was used in the experiments, is available at https://centre-borelli.github.io/ruptures-docs/user-guide/detection/window/

avoid labeling every small peak caused by noise as a pivot point. In the experiments presented below, window size value of 5 has been used.

After the discrepancy has been calculated for all timepoints t, peaks in this curve indicate the points where the difference in the signal segments before and after t is the largest [20]. These, then, are the most probable change points in the signal. The basic idea of the window-sliding algorithm is illustrated in Figure 6.5, where the two sides of the blue box depict the two sliding windows.

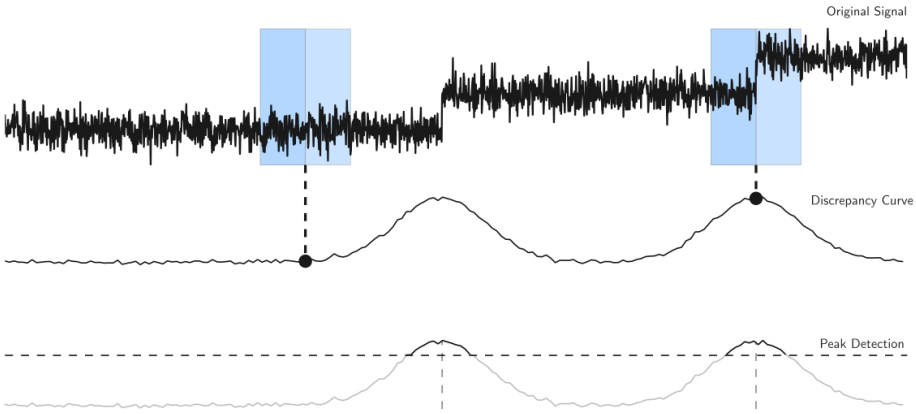


Figure 6.5: Illustration of the window-sliding algorithm, where the two parts of the blue box are the sliding window and the middle point is timepoint t. Image from Truong et al. [20].

When testing the algorithm in practice with the sample data, Gaussian process change* turned out to be the best performing option for the cost function. In the function definition

$$c(y_I) = |I| log\ det\ \hat{\sum}_I,$$

I is the interval of the sub-signal y_I and $\hat{\sum}_I$ is an empirical covariance matrix for $y_I = \{y_t\}_{t \in I}$. As its name suggests, the function uses Gaussian distribution to model the timeline and detect shifts in the estimated mean and scale in order to locate change points in the signal. What this means in practice when applied to the discourse data depends somewhat on the specific pattern: one example case is shown below in Figure 6.6.

There is an important feature in the data that has to be taken into account when implementing the algorithm in an unsupervised setting: the number of change points is not known in advance. In such a case, when $\mathcal{T}$ is the segmentation or split of the timeline based on the identified change points and $pen(\mathcal{T})$ is a penalty for too complex segmentation,

*Information on the ruptures-library implementation of this cost function is available at https://ctruong.perso.math.cnrs.fr/ruptures-docs/build/html/costs/costnormal.html

$$min_{\mathcal{T}} V(\mathcal{T}, y) + pen(\mathcal{T})$$

is the optimization problem to be solved. The penalty used here,

$$pen(\mathcal{T}) := \beta|\mathcal{T}|,$$

has a smoothing parameter β which defines how much the growth in the number of detected change points, $|\mathcal{T}|$, is allowed to impact the result. In this implementation, the penalty term grows linearly with additional change points.

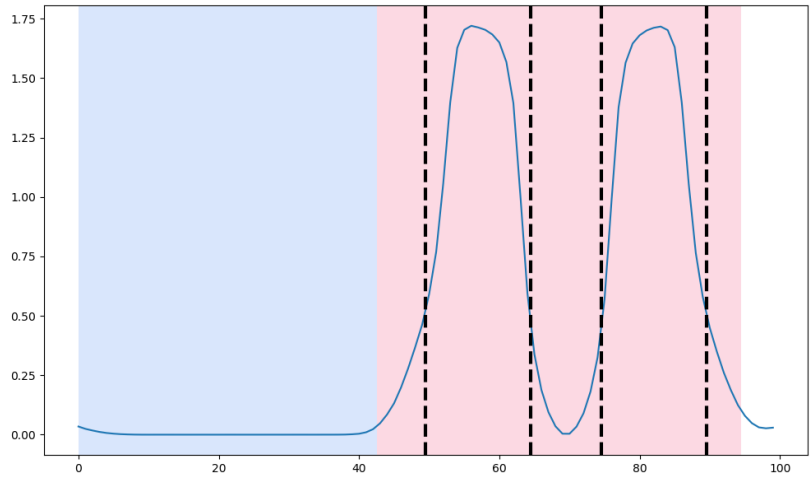


Figure 6.6: Example of pivot point detection with the window-sliding algorithm, using $\beta = 0.5$ in the penalty term. The dotted vertical lines mark the pivot points predicted by the algorithm.

Figure 6.6 presents an example of the window-sliding algorithm applied to the timeline of averaged cosine distances. The timeline is the same one shown above in Figure 6.3, based on a sample dataset with the 'Down-Up' pattern. The shaded red area in the image indicates the segment of the timeline between the actual pivot points, while the dotted vertical lines mark the pivot points predicted by the algorithm. As can be seen in this example, the drop in the distance timeline corresponding to the valley in the discourse pattern has an impact on the results, so that even when using penalty term $\beta|\mathcal{T}|$ with $\beta = 0.5$, the algorithm predicts there to be too many pivot points in both timelines. Based on experimentation with different values of β, it is hard to find a penalty weight that would produce balanced results with all types of timelines resulting from the different discourse patterns. Therefore, the value $\beta = 0.5$ was used in the tests as a compromise solution.

7. Methods for evaluating, visualizing and exploring the results

A variety of approaches are used for assessing the impact of contrastive pretraining and the performance of the methods applied in the downstream tasks. As evaluation metrics like Rand index and F-score and methods like t-SNE and K-means clustering are considered to be well known in data science context, they are not described here in detail. Instead the focus is on explaining how they are used for exploring, visualizing and evaluating the results of the models presented above.

7.1 Exploring the results of contrastive pretraining

A basic indicator of the learning ability of the model in the training phase is provided by the training and validation losses. At the beginning of model training, the training data is split into train, validation and test sets. During the training, individual samples are chosen randomly so that each epoch the order of input is random while also making sure that no dataset is used as input twice in a single epoch. Large validation set size helps to get a better idea of the model performance during training. Therefore, training set of 3000 samples[†] and validation set of 2000 samples were used. An example of a typical loss behaviour is shown in Figure 7.1, where high fluctuations during the early epochs stabilize over time, and the decrease of both training and evaluation losses slows down and then halts after approximately 30 epochs.

Observing only the loss, however, gives a very narrow idea of what and how the model is actually learning. The loss figure is based on the contrastive loss function, which gets as input the final output of the model. As was described above, and illustrated in Figure 5.1, the neural network model used in contrastive pretraining is

[†]Here sample is a synthetic timeline consisting of 100 timepoints, and timepoints contain varying number of document vectors.

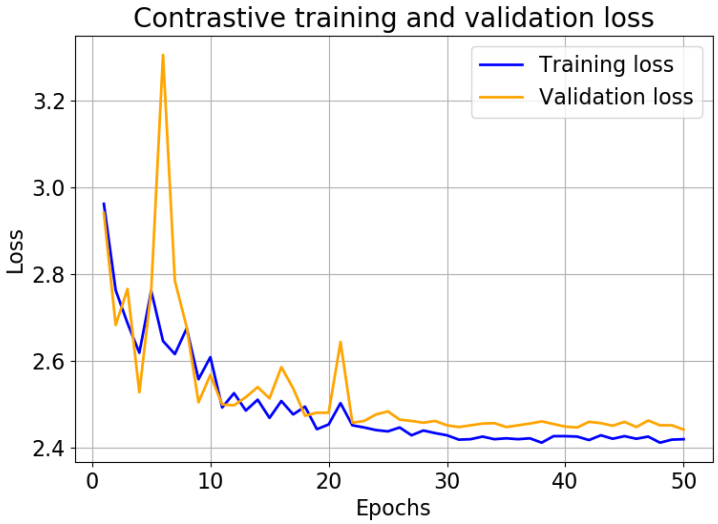


Figure 7.1: Example of the development of training and validation losses during 50 epochs of training. The contrastive model in questions used datasets ('Noise 2') where noise was added to the discourse patterns.

constructed in a way that makes it possible to extract also intermediate products that roughly correspond to document and timepoint presentations. These, together with the final output vectors, are used for examining and evaluating the model. The following sections present methods that are applied for this task, as well as the metrics with which the classification performance is evaluated in the downstream tasks.

7.1.1 K-means clustering and Rand index

As stated above, the goal of supervised contrastive learning is to bring the representations of datapoints belonging to the same category closer to each other while distancing them from samples belonging to other categories. One of the aims of this research is to find out whether this is visible already at the intermediate stages of the model, especially at the timepoint level. Ideally, the model would transform the document and timepoint representations in a way that would allow distinguishing timepoints belonging to unstable and stable patterns from each other.

Clustering provides one general approach that can be used for assessing to what extent this kind of transformation actually takes place. Here, the sklearn library* implementation of the general purpose K-means (or Lloyd's) clustering algorithm is used.

Clustering is applied to both the document vectors and the timepoint vectors,

*More information available at https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

which are obtained as intermediate outputs of the trained contrastive models. Document vectors are clustered by first selecting randomly 500 datasets from the test data, and then implementing K-means twice, using different number of clusters. First the number of cluster is set to two, and the results are compared to the timepoint labels (stable/unstable) of the timepoints which the documents belong to. In addition, the document vectors are clustered so that the number of clusters corresponds with the number of discourse categories in the dataset. In both cases, the final results are averaged over all of the 500 datasets in the sample.

Timepoint vectors are clustered similarly, so that the number of clusters is based both on the binary timepoint labels, where the results are averaged over the datasets, and on the discourse patterns of the datasets where the timepoints belong to. In the latter case, all of the timepoints in the test data are clustered as one group, so that averaging of the results is not needed.

Since the labels defining the 'true clustering' of the data are available, it is possible to compare them with the clustering results. There are different metrics available for this purpose*, out of which the adjusted Rand index was chosen to be used here.

An important challenge for evaluating clustering results is the fact that the specific labels assigned to the datapoints by the clustering algorithm can not be assumed to be the same as the true labels. To give an example, we can define label '0' to signify that a timepoint belongs to a stable discourse pattern, and assign label '1' for timepoints in unstable patterns. If we then have the following true labels for a dataset of 10 timepoints

$$y = [0, 0, 1, 1, 1, 1, 0, 0, 0, 0],$$

the clustering algorithm might well output the correct cluster assignments but use the opposite labels

$$\hat{y} = [1, 1, 0, 0, 0, 0, 1, 1, 1, 1].$$

In such a case, measures like accuracy, which look at the absolute label values to evaluate correct label assignments, can not be used. Rand index avoids this problem by comparing instead the cluster assignments of all of the pairs of elements in the set of true and predicted labels, while ignoring the specific cluster labels. If a pair of datapoints is assigned to the same cluster 0 in y and it is also assigned to the same cluster 1 in $\hat{y}$, the resulting Rand score goes up. This means that, for instance, the Rand score of the clustering

*See for instance https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

$$\hat{y} = [0, 0, 0, 1, 1, 1]$$

is the maximum, or 1.0, whether the actual clustering is

$$y = [0, 0, 0, 1, 1, 1],$$

or

$$y = [1, 1, 1, 0, 0, 0],$$

regardless of the specific labels. If we take two different partitions of the data, U and V, where

- a is the number of pairs of datapoints that belong to the same cluster in U and V
- b is the number of pairs where the items in the pair belong to the same cluster in U and in different clusters in V
- c is the number of pairs where the items in the pair belong to the same cluster in V and in different clusters in U
- d is the number of pairs where the items in the pair belong to different clusters in both U and V,

then the Rand index can be defined as

$$RI = \frac{a+b}{a+b+c+d}.$$

A problem with this unadjusted version of Rand index is that it does not necessarily assign score 0 for random cluster assignment [18]. To ameliorate this, adjusted Rand index uses instead the formula

$$ARI = \frac{\frac{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\frac{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]},$$

where n is the number of items in the data and $\frac{n}{2}$ gives the number of possible pairs that can be formed from them [18]. With this formulation, the expected value of the score is now zero, and the score is one when the cluster assignments match perfectly*.

*For more details on the sklearn implementation of the adjusted Rand score, which is the one used in the experiments, see https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html

7.1.2 Permutation testing

Another method used for exploring the impact that contrastive pretraining has on the document and timepoint vectors takes advantage of the permutation testing approach already discussed in Section 6.2. In this case, the hypothesis that is tested states that the distance between timepoint vectors is dependent on their class labels (stable / unstable). The null hypothesis, on the other hand, claims that all of the distances come from the same distribution. The test statistic used for assessing whether the null hypothesis can be rejected is the cosine distance between the mean vectors of the two groups.

An approximate method of sampling a defined number of permutations is used also here, and the p-value is calculated simply as the fraction of the sampled groupings of the timepoints for which the cosine distance is as large or larger than that between the label groupings. The null hypothesis is rejected when the p-value is smaller than $\alpha = 0.05$.

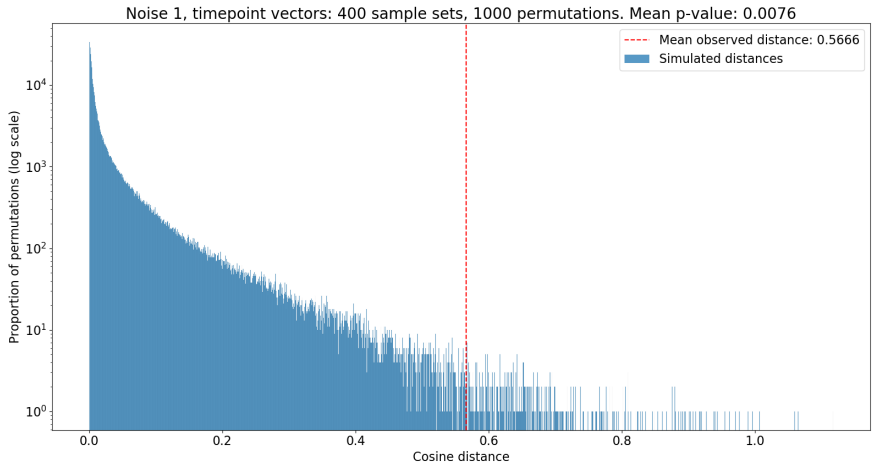


Figure 7.2: Results of a permutation test performed to timepoint vectors. The values of cosine distance are in the interval [0, 2], where zero distance indicates complete similarity. The red vertical line in the plot marks the ground truth distance.

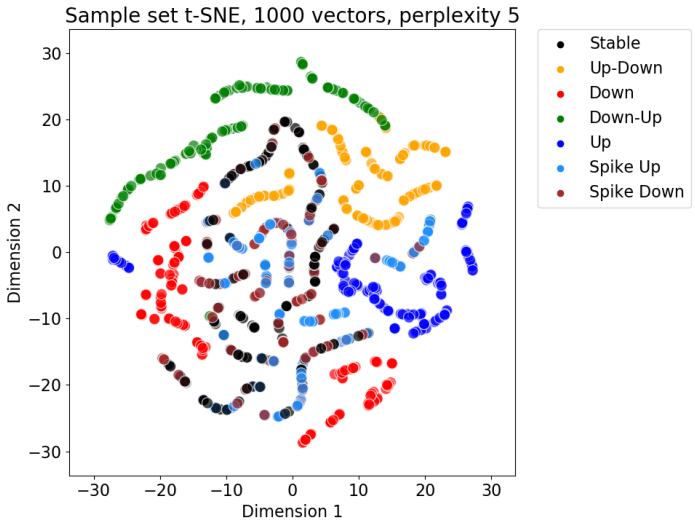
Figure 7.2 illustrates the combined results of 1000 permutation tests performed to timepoint vectors from 400 datasets. The histogram shows the logarithm of the numbers of distance values for different distances obtained in the permutation tests, and the red vertical line marks the mean distance measured using the actual classes defined by the labels. In this example case, the mean of the p-values obtained from the tests is ≈ 0.007, so clearly below the chosen significance level of 0.05. This result indicates that here the null hypothesis can be rejected and it can be assumed that the distance between the vectors depends on whether they belong to a stable or unstable

timepoint.

t-SNE

t-SNE is a method developed by van der Maaten and Hinton [21] for reducing the dimensions of high-dimensional data and visualizing it in a low-dimensional space. It is used here to provide one possible translation of the input data into a low-dimensional form, where the impacts of the contrastive pretraining on the vectors can be observed visually.

(a) t-SNE with perplexity 5.



(b) t-SNE with perplexity 50.

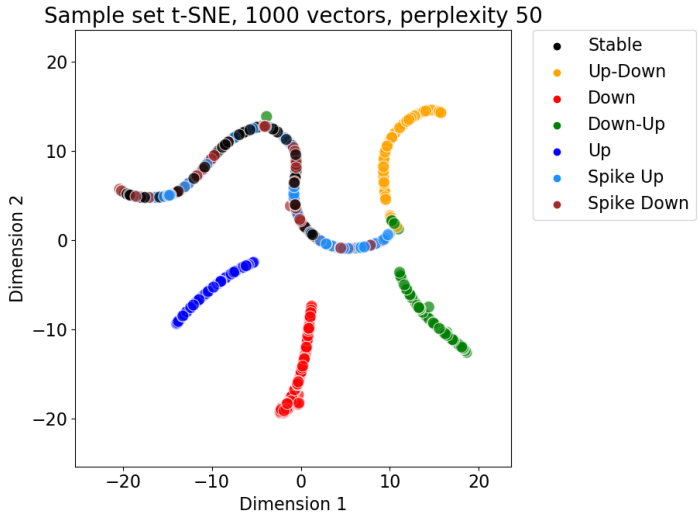


Figure 7.3: Example of the impact of different perplexity values on the t-SNE results.

Due to both the unavoidable loss of information when reducing data with hundreds of dimensions into two dimensional form, as well as the high variability of the outputs of t-SNE, the results obtained with this method should not be given excessive

interpretative weight. One important factor in defining the shape of the results is the perplexity parameter, for which there is no predefined optimal value. Figure 7.3 shows an example of t-SNE performed to the model output vectors using two different perplexity values. Each dot in the image represents one output vector (when input is a sample dataset), and the colors are based on the discourse pattern labels. It is clear that the chosen value has a big impact on the end result, even when it is within the typical bounds (5-50) defined by Maaten and Hinton [21]. The intuitive understanding of perplexity, whereby the higher the perplexity value for datapoint x_i, the more local neighboring points it has, is illustrated quite well by this example.

Same perplexity value (30), which is a compromise close to the midpoint of the typical range, is used in all of the t-SNE plots presented in Chapter 8.

7.2 Evaluating and exploring classification results

The results of supervised and unsupervised classification are assessed with metrics that are commonly used in classification tasks. In the supervised setting, the loss on the train and validation datasets, as well as the accuracy of the classification results of the model, are measured after each epoch of the training process.

Accuracy measures simply the number of correctly predicted class labels n_c as a proportion of the total number of predictions n

$$accuracy = \frac{n_c}{n}$$

Accuracy can, however, provide misleading results in cases where the size of different classes in the data is very unbalanced. This applies to the binary classification of timepoints, where the number of timepoints belonging to an unstable discourse pattern can be a very small proportion of the total timepoints. To take a practical example, if we have a dataset of 100 timepoints and only 10 timepoints belong to the unstable category, then the accuracy of a prediction where all timepoints would be classified as stable is

$$\frac{90}{100} = 0.9$$

or 90%. This is a good result, but it obviously does not give an accurate picture of the model's performance. Therefore, additional metrics are needed. We can formulate the previous example prediction also using different terms:

Positive class: 'unstable'	
True Positives (TP): 0	False Positives (FP): 0
True Negatives (TN): 90	False Negatives (FN): 10

Positive class: 'stable'	
True Positives (TP): 90	False Positives (FP): 10
True Negatives (TN): 0	False Negatives (FN): 0

Here, TP defines how many timepoints belonging to the 'positive' class in the data were classified correctly by the model, FP measures how many 'negative' were incorrectly classified as 'positive', and TN and FN measure the same for the 'negative' category. Now we can define precision

$$precision = \frac{TP}{TP + FP}$$

as the proportion of the model classifications for the 'positive' category that were correct, and recall

$$recall = \frac{TP}{TP + FN}$$

as the proportion of the timepoints actually belonging to the 'positive' category in the data that were correctly assigned to that class by the model. With the example classification results, we get $precision = 0$ and $recall = 0$ when 'unstable' is the positive category, and $precision = 0.9$ and $recall = 1$ when 'stable' is the positive category.

Clearly, in this example case precision and recall scores are able to capture the problem with the classification results that was missed by the accuracy measure. However, in the binary classification scenario, the focus is on the model's ability to correctly classify timepoints belonging to the 'unstable' category, and therefore only those scores will be used when evaluating the results. In the multiclass classification case, where classes consist of the unstable discourse patterns, an average of the scores for different categories, weighted by the number of datapoints belonging to each class, is used.

An additional measure, called F-score, which combines the information of the precision and recall scores, is also used for evaluating the classification results. The F-score is the harmonic mean of the precision and recall scores,

$$F = \frac{2}{precision^{-1} + recall^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

In the binary classification case, if the model had been able to classify one of the ten unstable timepoints correctly, we would get $precision = 1$ and $recall = 0.1$, and the F-score would then be

$$F = 2 \cdot \frac{1 \cdot 0.1}{1 + 0.1} \approx 0.18,$$

which balances between the highly divergent precision and recall scores.

7.2.1 Evaluating pivot point detection

The results of both supervised and unsupervised pivot point detection are evaluated using the already familiar metrics Rand index, precision, recall and the F-score. Due to the nature of the task, however, the implementations of these methods are somewhat different from what was described earlier.

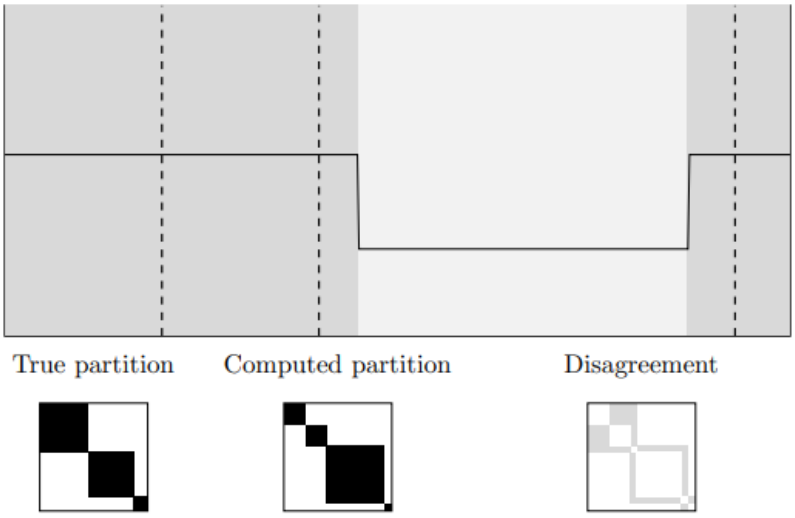


Figure 7.4: The upper image depicts the results of a pivot point prediction, where the light gray area is the segment between the true pivot points and the dashed vertical lines are the predicted pivot points. The lower images show the membership matrices depicting the different partitions. Image from Truong et al. [20].

Figure 7.4 shows an example of a segmentation of a dataset by the predicted pivot points, indicated with dashed vertical lines, and a true segmentation, indicated with a lighter gray color. The true pivot points split the data into three partitions, while the predicted pivot points produce four segments. In the discourse detection context, we can think of both segmentations as placing each timepoint in the data into one of the segments. If $\mathcal{T}^*$ is the set formed by the true pivot points and $\hat{\mathcal{T}}$ is the set of predicted pivot points, we can define

$$gr(\mathcal{T}) = \{(s,t), 1 \leq s < t \leq T\}$$

as the set of grouped indices, where T is the number of timepoints in the data and s and t belong to the same segment according to the pivot point set $\mathcal{T}$. Similarly,

$$ngr(\mathcal{T}) = \{(s,t), 1 \leq s < t \leq T\}$$

is the set of non-grouped indices, where timepoints s and t do not belong to the same segment according to the pivot point set $\mathcal{T}$. Now the Rand index evaluating the

pivot point prediction can be formulated as

$$RI(\mathcal{T}^*, \hat{\mathcal{T}}) := \frac{|gr(\hat{\mathcal{T}}) \cap gr(\mathcal{T}^*)| + |ngr(\hat{\mathcal{T}}) \cap ngr(\mathcal{T}^*)|}{T(T-1)},$$

which is a measure of the uniformity of the two segmentations that compares the way that pairs of timepoints are allocated in each of them [20].

When defining precision and recall for pivot point detection, the main difference from what was discussed before is the introduction of a margin of error variable M. It defines how far a predicted pivot point needs to be from a true pivot point to be counted as a true positive [20]. If a pivot point is predicted to be at timepoint 10, but the true one is at timepoint 15, the prediction is counted as a true positive if $M > 5$. More formally,

$$TP(\mathcal{T}^*, \hat{\mathcal{T}}) := \{t^* \in \mathcal{T}^* | \exists\ \hat{t} \in \hat{\mathcal{T}}\ s.t.\ |\hat{t} - t^*| < M\}$$

With this formulation of true positives, precision and recall are defined as

$$precision(\mathcal{T}^*, \hat{\mathcal{T}}) := \frac{TP(\mathcal{T}^*, \hat{\mathcal{T}})}{\hat{K}}$$

$$recall(\mathcal{T}^*, \hat{\mathcal{T}}) := \frac{TP(\mathcal{T}^*, \hat{\mathcal{T}})}{K^*},$$

where $\hat{K}$ is the true and K^* the predicted number of pivot points. Now the F-score, or the harmonic mean of precision and recall, can be calculated with the familiar formula

$$F(\mathcal{T}^*, \hat{\mathcal{T}}) := 2 \cdot \frac{precision(\mathcal{T}^*, \hat{\mathcal{T}}) \cdot recall(\mathcal{T}^*, \hat{\mathcal{T}})}{precision(\mathcal{T}^*, \hat{\mathcal{T}}) + recall(\mathcal{T}^*, \hat{\mathcal{T}})}$$

The value chosen for the error variable M can have a big impact on the precision, recall and F-scores. If the margin is very large, even far off predictions will be counted as true positives, whereas an excessively small margin can lead to underestimating the capacity of the model. Here, $M = 5$ was chosen as a compromise between the two extremes.

8. Experimental results

This chapter presents the main results of the experiments with the models and data that have been described in the preceding chapters. First, the results of the contrastive model training are introduced in Section 8.1, and the model learning is assessed using the methods described above in Section 7.1. Then, in Section 8.2, the performance of both the supervised and unsupervised classification methods are analysed and compared with each other.

8.1 Contrastive pretraining

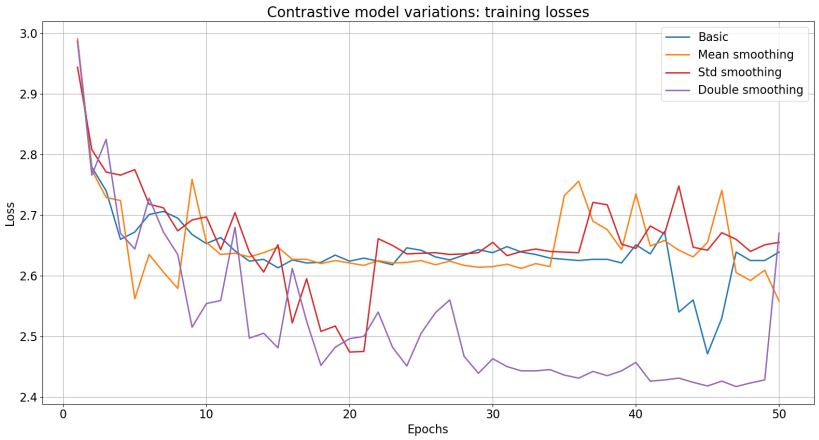
8.1.1 Training, validation and test losses

Contrastive pretraining was performed using four variations of the neural network model, listed above in Table 5.1. Three of the models add information obtained from the immediate neighborhood of each timepoint vector in the form of smoothing vectors to the inputs of the LSMT layers, while the 'Basic' model does not use any smoothing techniques. Figure 8.2 shows the development of contrastive loss during model training, using both training and validation datasets, as well as the loss for each model variation when running the trained model with the test dataset.

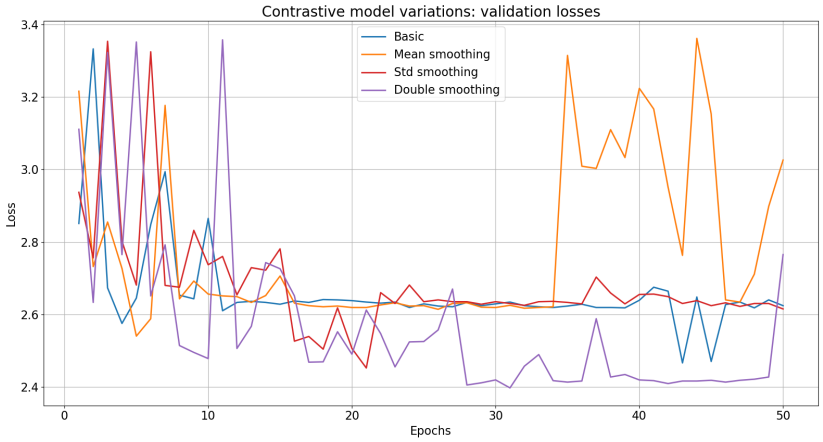
The training losses show a generally decreasing trend, with some fluctuation during the early epochs followed by a more stable curve. In most cases, the model converges after approximately 20 epochs, while the model that uses double smoothing keeps on improving longer and reaches the lowest loss level of the compared models. Validation losses fluctuate even more strongly at the beginning of the training, and for all models except 'Double smoothing', the losses stop decreasing already after approximately 10 training epochs. Test losses confirm the superior performance of the 'Double smoothing' model variation, although the differences between the models are relatively small.

Interestingly, the 'Mean smoothing' model performs generally worse than the model that does not use smoothing, but when information of the mean and standard deviation in the neighborhood of the timepoint vectors is combined, best overall per-

(a) Training losses.



(b) Validation losses.



(c) Test losses.

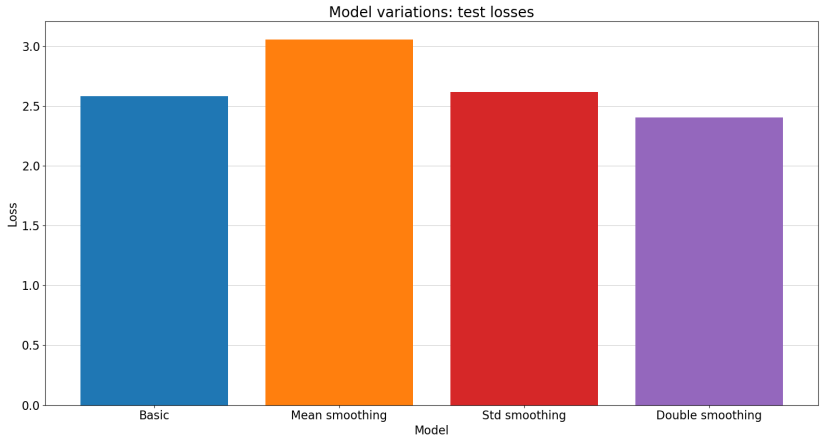


Figure 8.1: Training, validation and test losses for contrastive prelearning with different model variations.

formance is reached. This indicates that the bidirectional LSTM layers are better able to filter and encode information that is relevant at the level of the time series when they are assisted in this task by the enriched input data. The method of enrichment, however, makes a difference, as can be seen from the results obtained with the different

smoothing approaches.

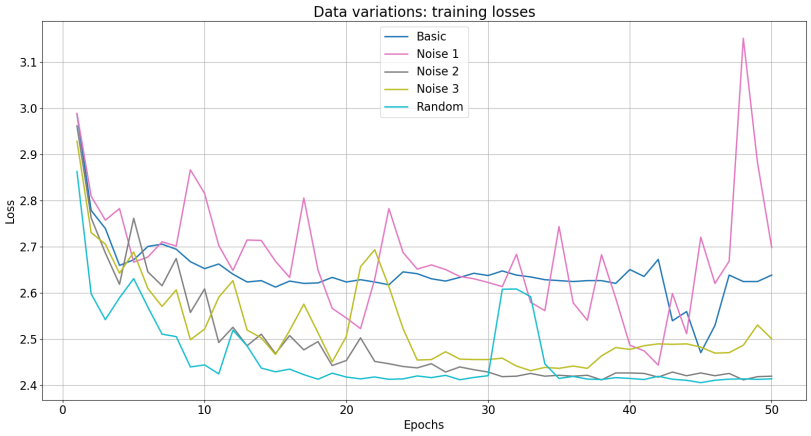
The impact of adding different amounts and types of noise to the data were tested using the synthetic dataset variations, listed in Table 4.1 above. Noise was added to the two discourse patterns, each containing documents belonging to a single category, by increasing variation in the numbers of documents at different timepoints, as illustrated in Figure 4.2. Other kind of noise was generated by increasing the number of news categories covered by the articles used in the stable discourse pattern, thus simulating a more realistic setting where the text corpus consists of documents covering a large number of themes. This dataset type is visualised above in Figure 4.1.

Training with the noisy dataset variations was done using the 'Basic' contrastive learning model, where no smoothing was applied to the timepoint vectors. Resource limitations are the main reason for not testing all data and model variations. Training a single model for 50 epochs requires over 120 hours of computing time when using a high performance computing environment, which set a limit to the amount of testing that could be done. In addition, considering the scope and purpose of the thesis, it was not deemed necessary to include additional model and data variations in order to obtain a general idea of the performance of the contrastive model.

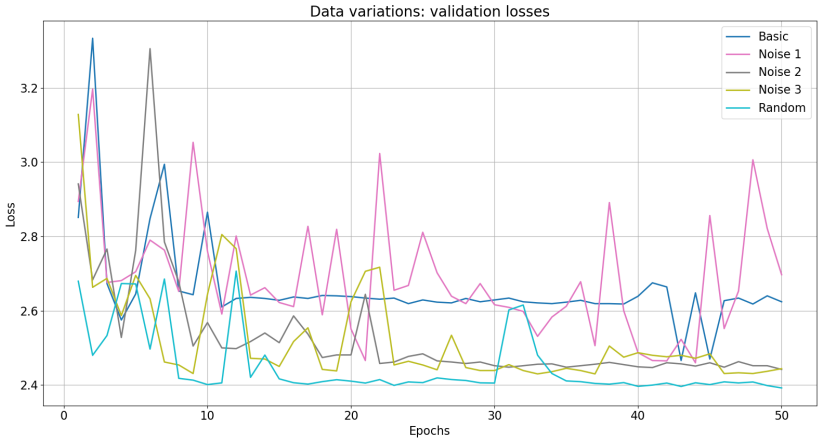
Figure 8.2 shows the training, validation and test results for the different data variations. The results for a dataset without added noise are also included here to provide a reference point. At a general level, the same trends that were detected above are also visible here. Losses fluctuate more during the first 10-20 epochs, and converge at a relatively early stage of the training. The loss curve is most stable when the model is trained with noiseless data, but, interestingly, the loss of this model also stabilises at the highest loss level. All of the models using noisy data attain lower losses, and the differences between them are very small. Somewhat surprisingly, the 'Random' model that uses most complex data, mixing randomly documents from dozens of categories in the stable 'background' pattern, performs best in this test group. Added noise can make the model more robust against overfitting, which is a plausible explanation for the results observed when using the validation and test datasets. The fact that the models that use noisy datasets obtain lower loss levels also with the training data is, however, more surprising.

The lowest loss values reached with the different model and data variations are very similar, close to 2.4. The losses are not decreasing below this level, but instead begin to fluctuate again after around 40 epochs. This is one indication of the limitations of the model.

(a) Training losses.



(b) Validation losses.



(c) Test losses.

Figure 8.2: Training, validation and test losses for contrastive prelearning with different dataset variations.

8.1.2 Evaluating model learning

Document vectors, timepoint vectors and the output vectors of the contrastive model were described above in Section 5.1, where they are presented in relation to the model

architecture. Now it is time to use these outputs for exploring model learning and in particular for testing whether the different model layers in fact encode the kind of information to the data that we expect them to.

Output vectors

Out of the three kinds of data representations listed above, the final output of the model is in a sense the most straightforward case, and it is therefore discussed only briefly here. Based on the way that contrastive loss is implemented in the model, the representations of the model output vectors where the input sample sets have the same discourse pattern should be more similar to each other than to outputs with different pattern labels. The training, validation and test losses described in the previous section give an indication of how well this goal is achieved. Keeping in mind the limitations of the t-SNE method, discussed above in Section 7.1, it can be used for obtaining a visual idea of the functioning of contrastive loss in the different models.

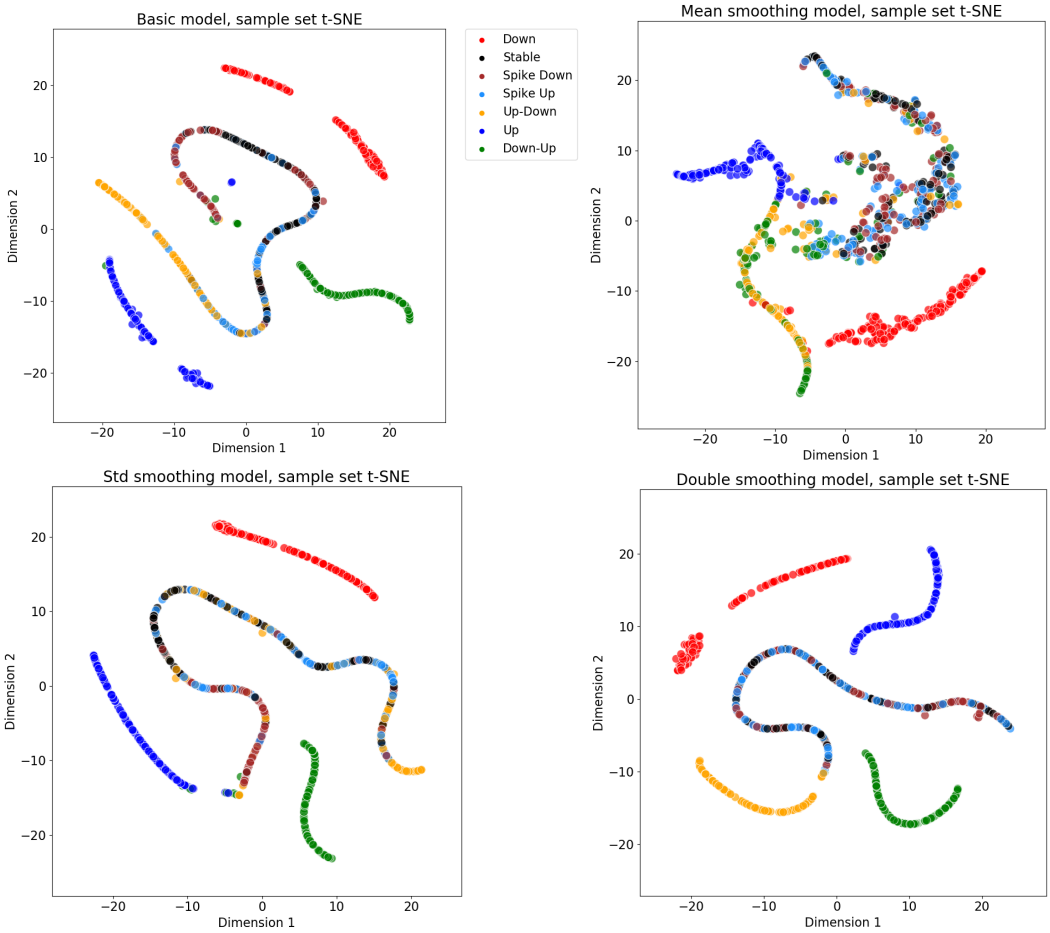


Figure 8.3: Results of t-SNE performed for 1000 model output vectors of the different model variants. Perplexity value 30 is used in all of the t-SNE models.

The results of t-SNE performed for outputs of different model variations, shown

in Figure 8.3, indicate that there are differences both between the performance of the different model variants as well as in the ability of the models to contrast different discourse patterns. Comparison between the models resembles the ranking based on the losses. Double smoothing model performs best, with four discourse patterns being quite clearly separated from the others in the t-SNE image. Mean smoothing model, on the other hand, shows the weakest results: it is barely able to separate two of the seven patterns.

Of the different discourse patterns, 'Up' and 'Down' appear to be the ones most easily distinguishable from the others, followed by 'Down-Up' and 'Up-Down'. None of the models, however, were able to properly differentiate between 'Spike Up', 'Spike Down' and 'Stable' patterns. Based on the t-SNE results, this is clearly one limitation of the contrastive model.

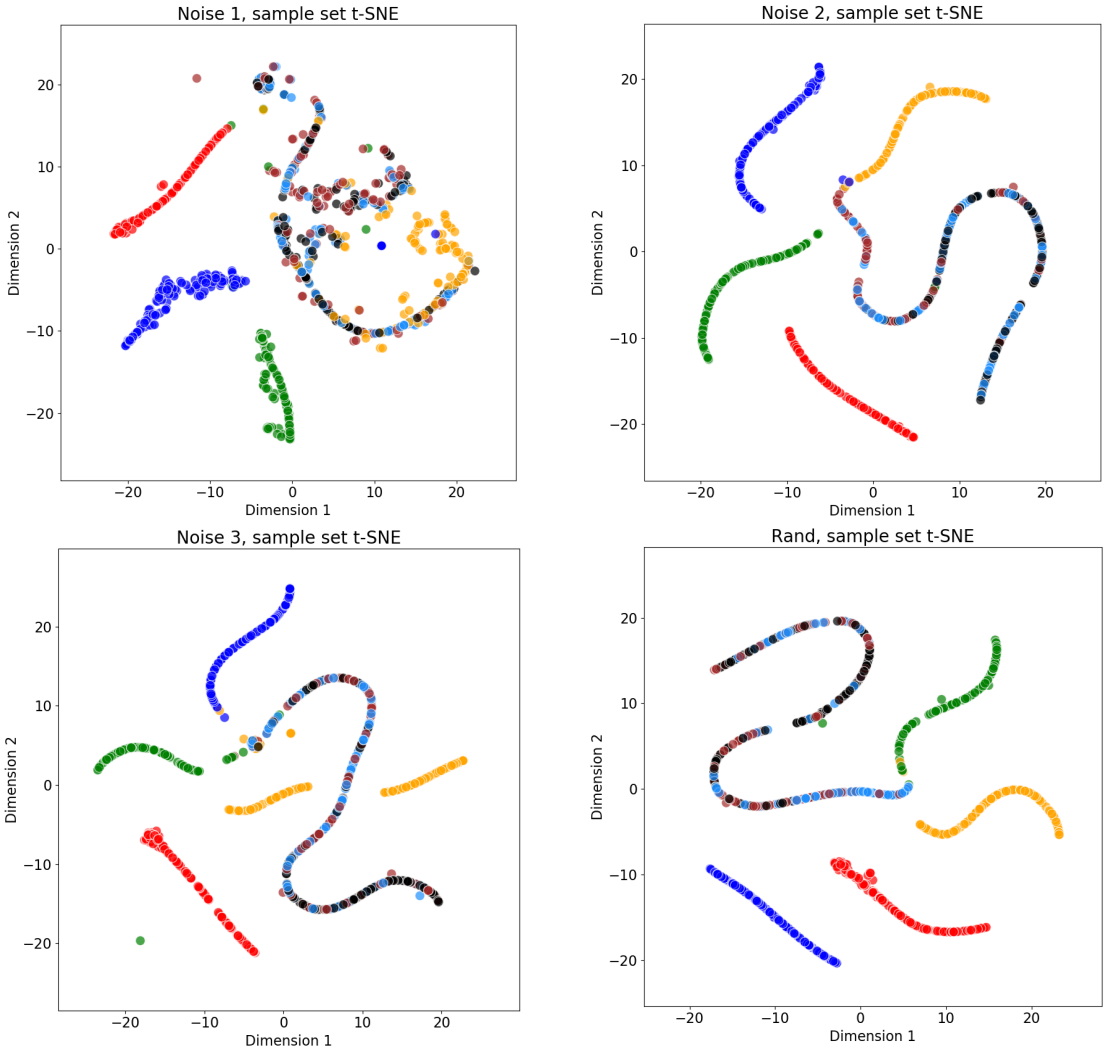


Figure 8.4: Results of t-SNE performed to 1000 model output vectors of the different dataset variants.

Same discourse patterns are also distinguishable as separate clusters in the t-

SNE models of the vectors produced using the noisy dataset variants, shown in Figure 8.4. The four patterns are most clearly clustered when using the 'Noise 2' and 'Rand' datasets. The other two examples show no clear improvement when compared with the 'Basic' model, where the data contains no added noise. Therefore the t-SNE results also support the idea that noise in the data can improve learning results, although this does not seem to happen in every case. The added noise does not, however, help the model to distinguish the three discourse patterns, 'Spike Up', Spike Down' and 'Stable', which appear mixed together in all of the t-SNE plots.

Timepoint vectors

The timepoint representations are 300 dimensional vectors that are the output of the first part of the model, which contains transformer encoder layers followed by fully connected layers. A timepoint vector requires as input the document representations belonging to a single timepoint.

At this level it is possible to investigate whether the model has encoded information to the vectors that relates to the discourse pattern of the dataset. Perhaps more interestingly, we can also examine if the timepoint representations differ based on whether the timepoint belongs to an unstable discourse pattern or not.

t-SNE is again used for obtaining a visual idea of possible groupings of the timepoint vectors based on the two classifications. In both cases, however, the plots show very few signs of meaningful groupings. As can be seen in Figure 8.5, in each of the three sample cases ('Basic' and 'Double smoothing' models using non-randomised data, and 'Basic' model with 'Rand' samples) the timepoint representations are scattered quite randomly on the two-dimensional plane, and no clear groupings based on either the pattern labels or the timepoint labels are visible. The t-SNE plots of the selected samples are representative of all of the tested model and data variations.

The information obtained from K-means clustering is slightly more varied. Figure 8.6 presents the results of binary clustering of the timepoint vectors. The 'Mean' bar shows the adjusted Rand score averaged over the sampled datasets, and 'Min' and 'Max' are the lowest and highest scores for each model and dataset variation. The differences between the models and datasets are very small, and in each case the mean score is beween 0.3 and 0.4. This means that the clusterings are not random, although they are still on average far from being a perfect match with the labels. However, large gap between the minimum and maximum values is a sign of high variation in the results for individual datasets.

Pattern based clustering, where the number of clusters corresponds with the number of discourse patterns in the data, shows much weaker results. The plots in

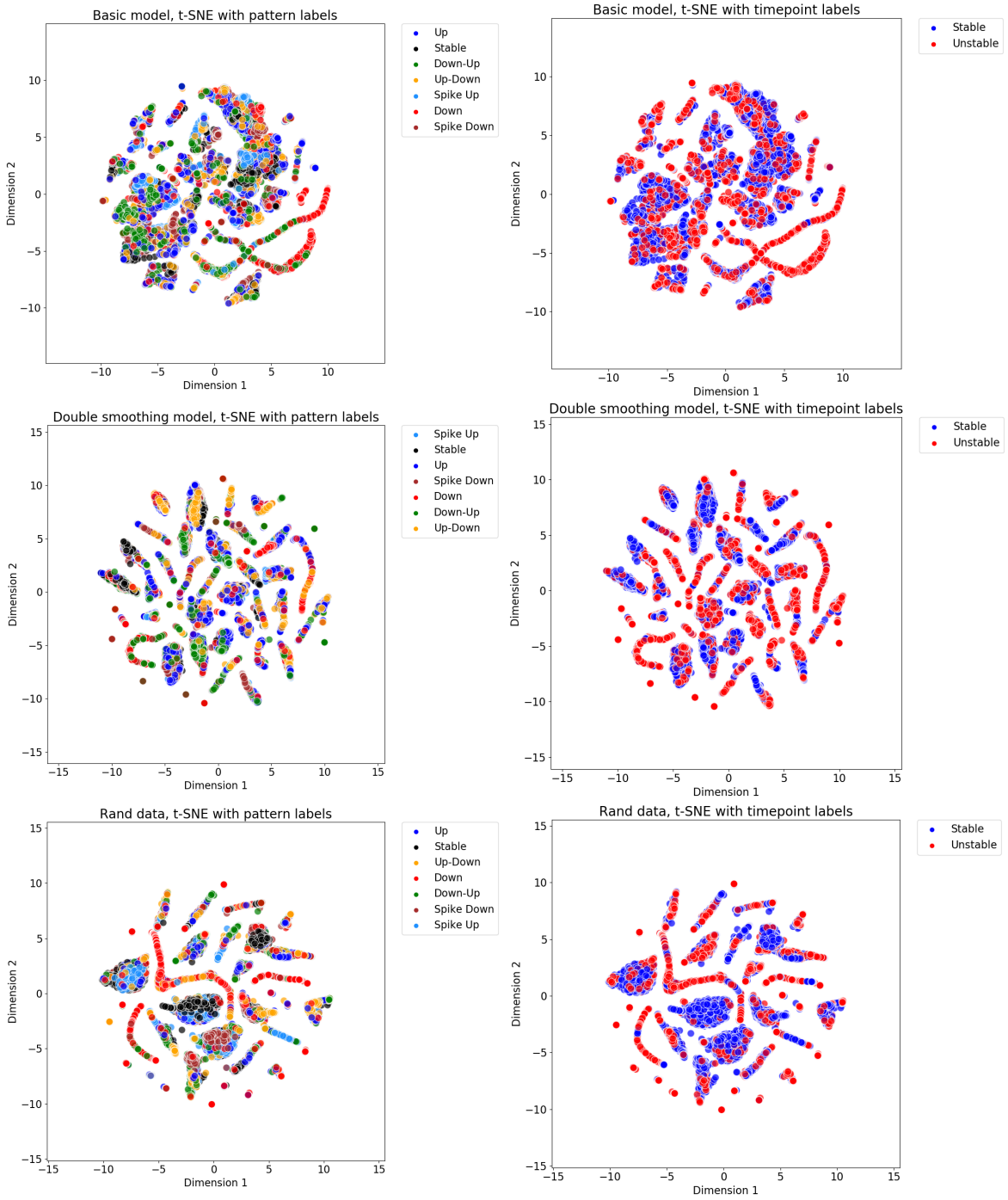


Figure 8.5: Results of t-SNE performed to 50000 timepoint vectors, using perplexity value 30. The colors used in the plots on the left are based on the discourse pattern labels, and in the plots on the right on the timepoint labels (stable / non-stable timepoint).

Figure 8.7 reveals that all of the adjusted Rand scores are very close to zero, which means that the clustering results do not match at all with the classification based on the discourse pattern labels.

The third, and final, method used for exploring the impact of contrastive pretraining on the timepoint vectors is permutation testing. Here, the distance between the

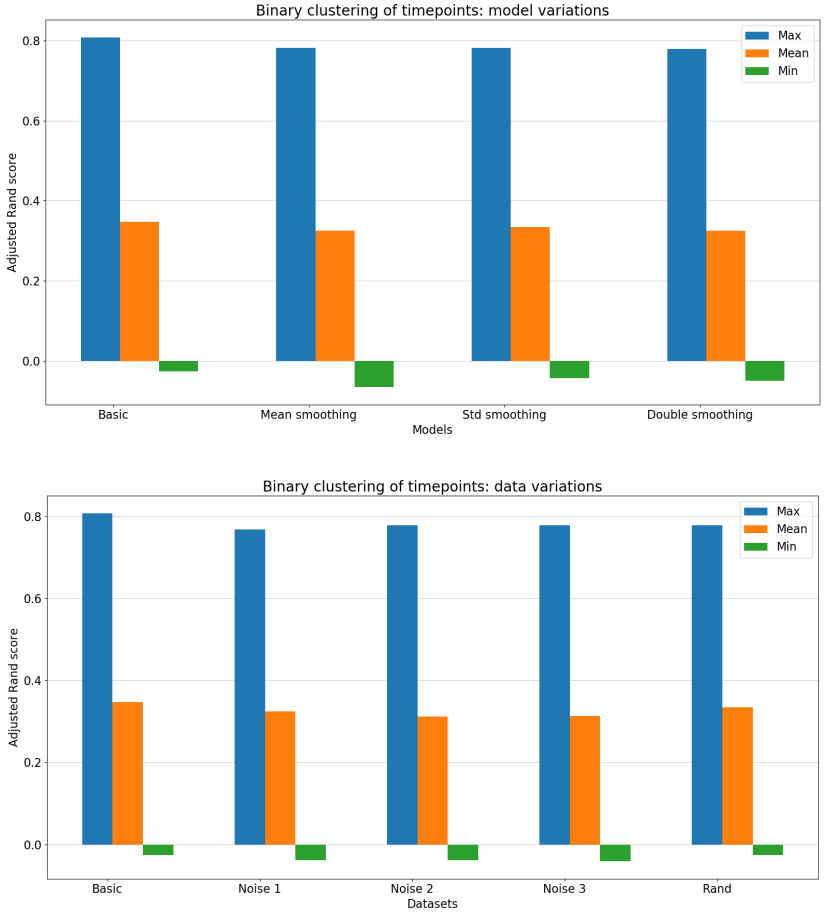


Figure 8.6: Adjusted Rand scores for binary K-means clustering performed on the timepoint vectors. Clustering was done for 500 sample datasets, and the maximum, minimum and mean Rand scores are indicated with the their own bars.

'average' vectors of the two groups (stable / unstable timepoints) is measured and compared with distances between groups formed from random permutations of the same vectors. P-value is measured for each sample set as the proportion of permutation distances that are higher than the averaged distance between the two classes.

Figure 8.8 shows a representative example of the results of permutation tests, where 1000 permutations were created using timepoint vectors from 400 sample datasets. The blue bars in the plot show the distribution of the distances between the random groupings of vectors, while the vertical red line indicates the mean of the observed distances between the label groups (stable/unstable) in the sample datasets.

The averaged p-values obtained from these tests are, as can be seen from Figure 8.9, well below the threshold value 0.05 for all model and data variations. These results indicate that the distances between the timepoint vectors are not independent of the binary class labels that define them as either 'stable' or 'unstable', depending on their location in the timeline in relation to the unstable discourse patterns.

Based on the above evaluations, it appears that contrastive pretraining does en-

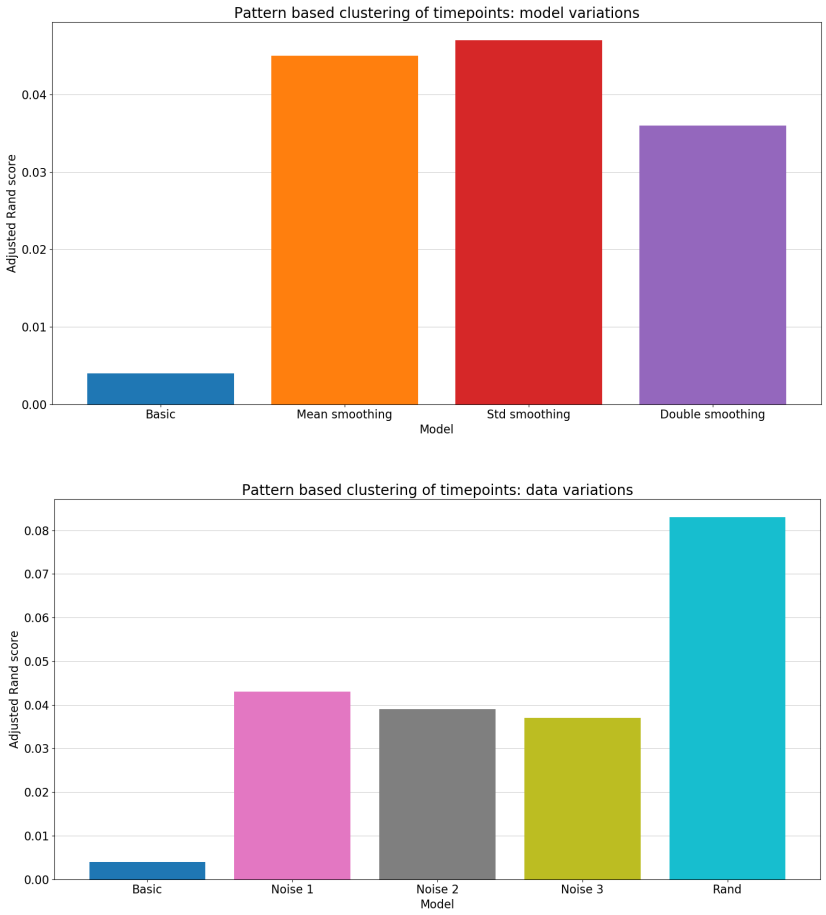


Figure 8.7: Adjusted Rand scores for pattern based K-means clustering performed on the timepoint vectors. Clustering was done together for all timepoints in the 500 sample datasets.

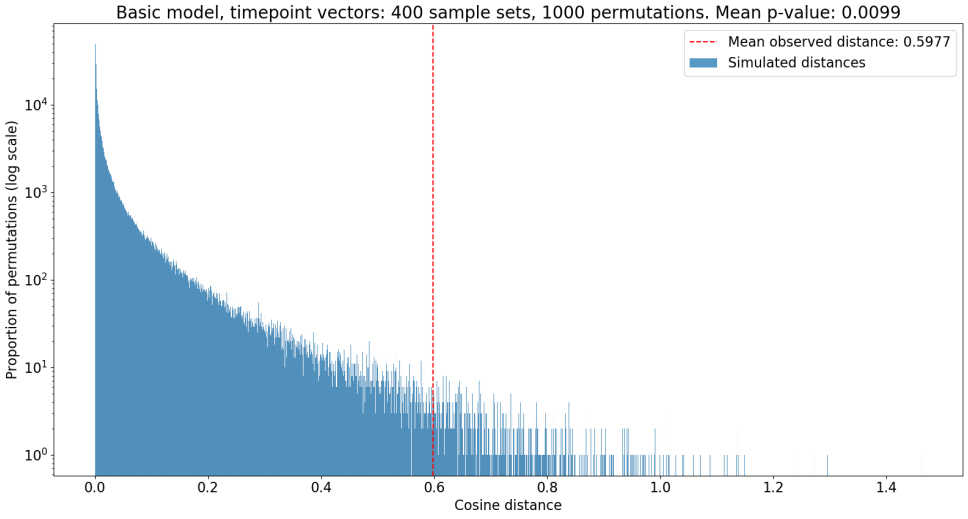


Figure 8.8: Results of permutation tests performed to the timepoint vectors from the 'Basic' contrastive model. X-axis shows the distribution of the measured cosine distances in logarithmic scale.

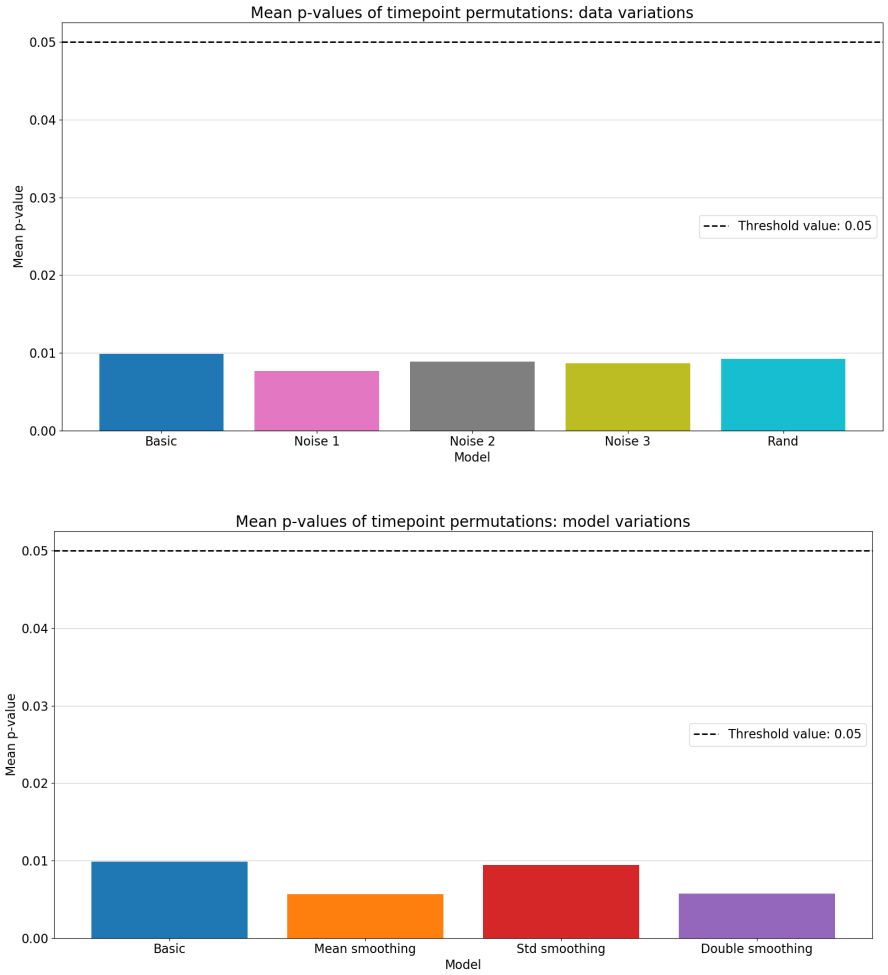


Figure 8.9: Averaged p-values from permutation tests performed to timepoint vectors using different data and model variations. The dotted vertical line marks the threshold value $\alpha = 0.05$.

code some information related to the binary labels also to the intermediate timepoint level representations. Strongest evidence of this is given by the permutation tests, while the t-SNE plots do not display a grouping of the timepoint vectors based on the timepoint labels. Results of K-means clustering are somewhere in between, indicating that there is a relatively weak differentiation between the vectors based on the binary labels. Interestingly, there is still much stronger evidence for this than for the differentiation of the timepoint vectors based on the discourse pattern labels, which were used as the basis for the contrastive loss.

To what extent the information encoded in the timepoint vectors is useful in downstream classification tasks will be discussed further in Section 8.2.

Document vectors

The documents in a synthetic sample dataset all belong to a specific timepoint, and all of them have a designated category, or topic. These, then, are features that can be explored when looking at documents belonging to a single dataset. Figure 8.10 uses again t-SNE to illustrate if, and how, these elements are visible in the output vectors of the pretrained transformer encoder layers.

The data used for the t-SNE consists of 2000 vectors sampled in each case from a single dataset, with equal number of documents being sampled from each of the timepoints. Sampling was used since the total number of documents in a single sample set is very large, on the order of 30000-40000.

The colors in the plots on the left side of the figure signify the categories which the documents belong to. In the two uppermost images, the dataset contains only two categories, which are quite well distinguishable from each other, even though they do not form clear clusters. The plot in the lower left corner uses the 'Random' data variation, where the stable discourse pattern contains documents from a variety of categories. As can be seen from the image, these are already much less distinguishable from each other.

The colors in the plots on the right side of the figure are based on the timepoint labels. Clearly the red dots, depicting the documents belonging to unstable timepoints, are in each of the three cases spread all over the plots. None of the examples show any sign of separation in the vectors based on whether the documents belong to a stable or unstable timepoint.

The implications of the t-SNE plots are supported by the results of K-mean clustering performed for the document vectors. Figure 8.11 shows the mean, maximum and minimum adjusted Rand scores for both binary (upper row) and category based (lower row) clustering, performed to documents from 500 datasets.

For binary clustering, the mean Rand scores are close to 0 for all model and data variations, and even the maximum values are very low. Both values are clearly higher for category based clustering, although the mean scores never go above 0.2. This does not give a strong indication of differentiation in the document representations based on either the timepoint or the category labels.

The permutation tests that were performed for the document vectors using the timepoint labels also give little support to the hypothesis that contrastive pretraining would differentiate the output of the transformer encoder layers based on the corresponding documents belonging to either stable or non-stable timepoints. Figure 8.12 shows an example of the results of a permutation test, performed to vectors from the 'Basic' model variant. The test used 2000 vectors sampled from each dataset, and the

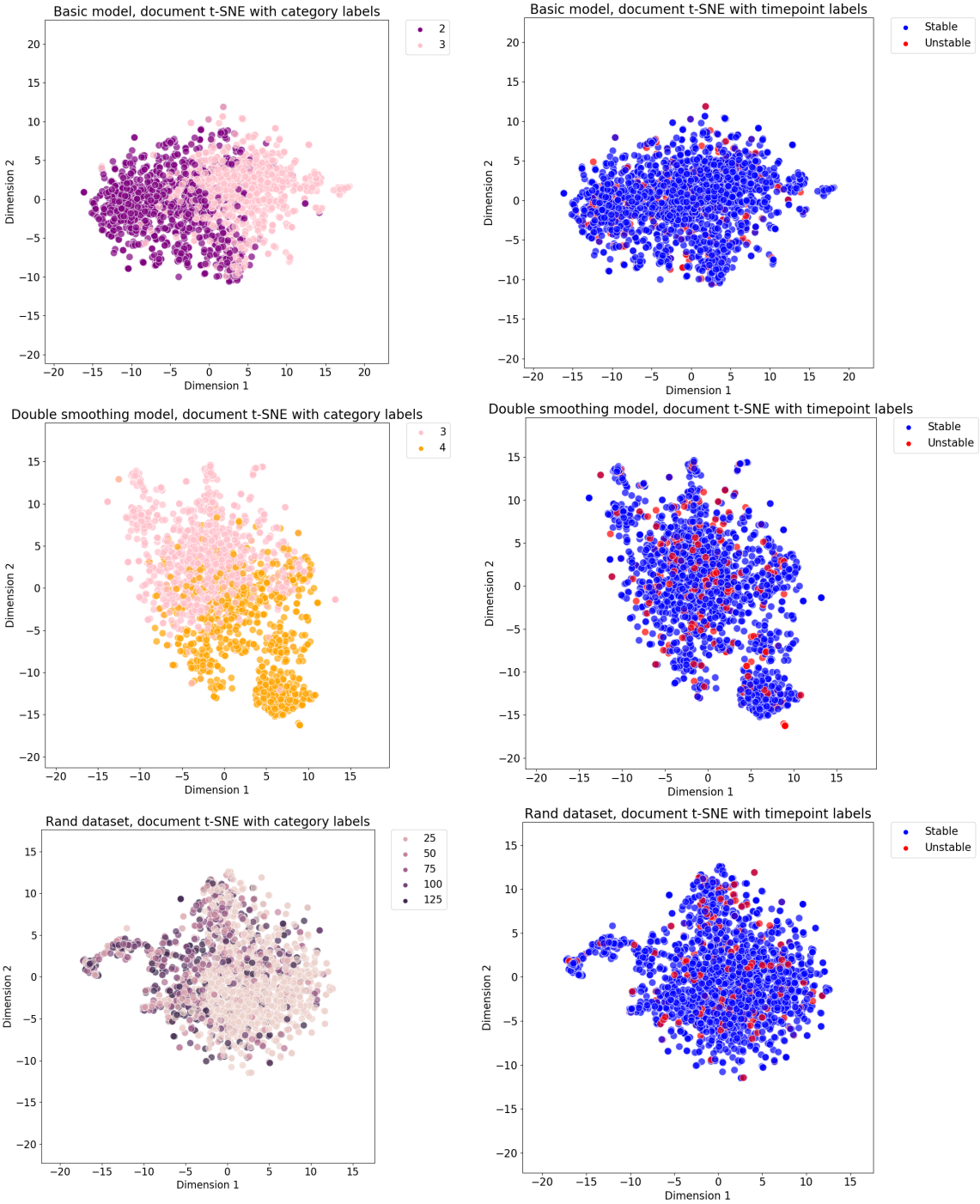


Figure 8.10: Example of results of t-SNE performed to 2000 document vectors sampled in each case from a single dataset. The colors used in the plots on the left are based on the discourse category labels, and in the plots on the right on the timepoint labels (stable / non-stable timepoint).

plot shows the results of 50 tests. The red vertical bar indicates the mean distance between the classes based on the timepoint labels.

The averaged p-value for the tests performed with the basic model variant is above the 0.05 threshold, and, as is shown in Figure 8.13, this is the case also with

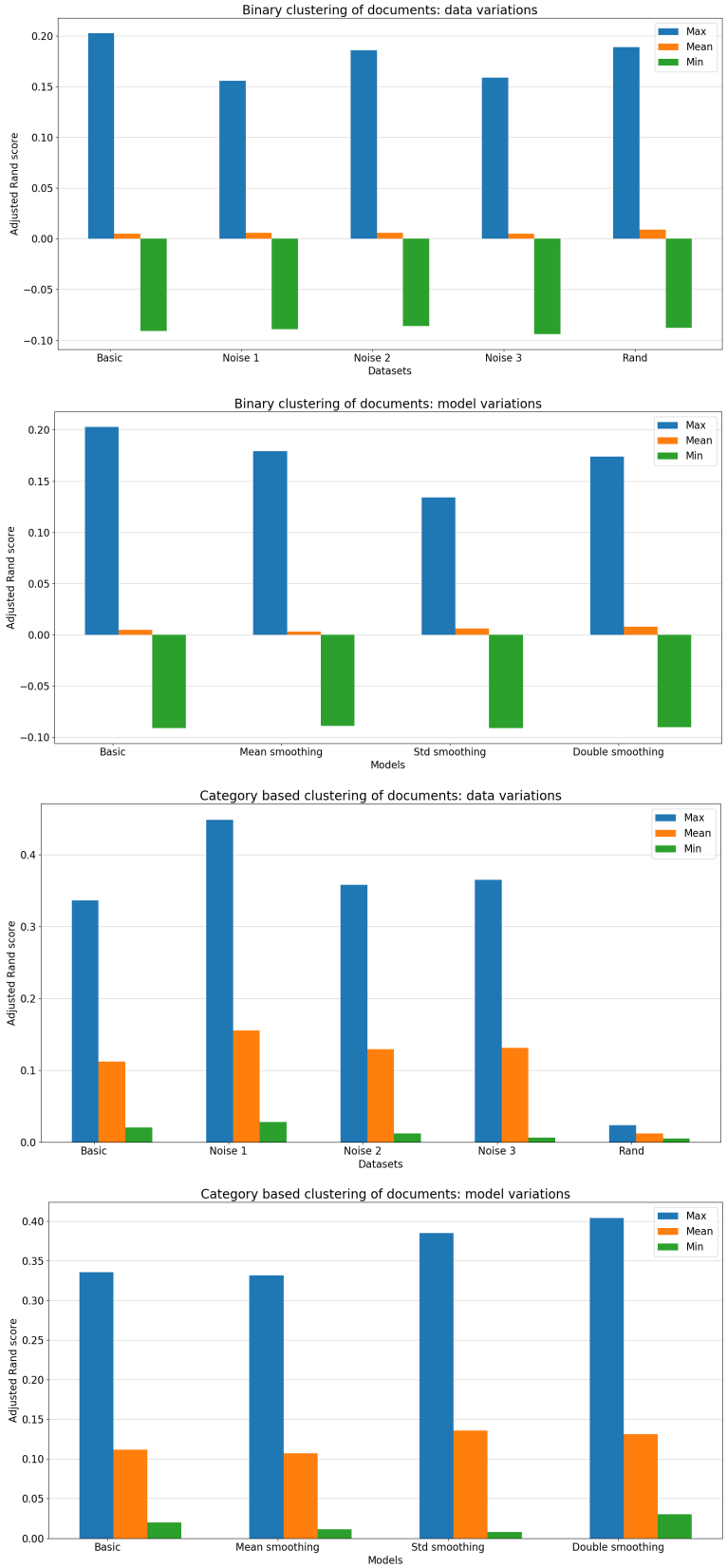


Figure 8.11: Adjusted Rand scores for binary and category based K-means clustering performed for the document vectors. The mean score is an average over the scores for 500 sample datasets.

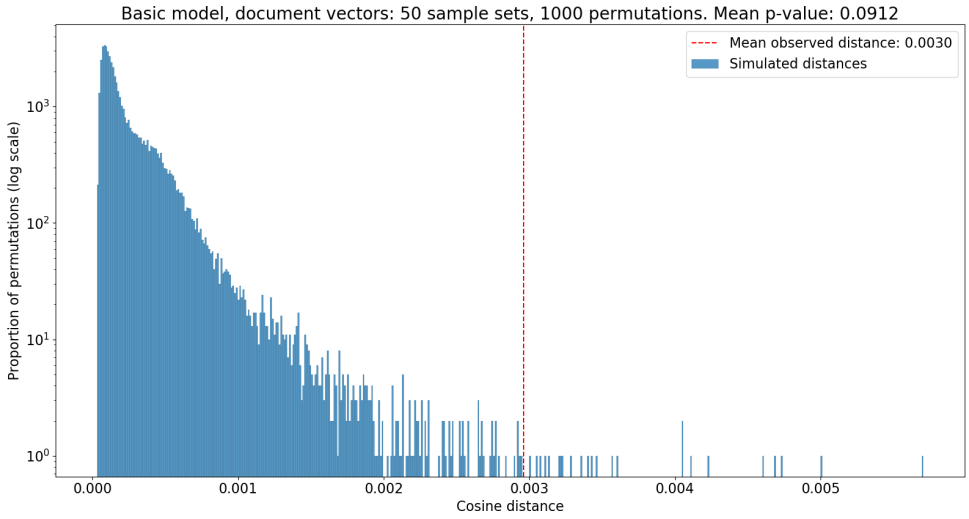


Figure 8.12: Results of permutation tests performed to the document vectors from the basic contrastive model. The distances are based on 100 permutations performed to 2000 document vectors from each of the 50 sample datasets.

most of the other model and data variants. Based on these results, the null hypothesis, which claims that there is no dependency between the timepoint labels and the distance between the document representations, can not be discarded.

8.1.3 Summary of contrastive model learning

The above exploration of three kinds of data representations provide mixed results concerning the contrastive model learning. Strongest evidence of model learning could be observed in the output vectors, which reflect most directly the learning goals set for the contrastive loss function. Based on the t-SNE visualizations, the pattern-based distancing of the output vectors succeeded in most cases to distinguish three of the seven discourse patterns used in the data. Three patterns, however, seem to have remained indistinguishable for all model and data variants, which is a probable reason for the stabilization of the model losses at the same level well before the training finished. This shows that the contrastive pretraining succeeds relatively well in its main task, defined in the loss function, although there is still room for improvement even with the simple synthetic data variations that were used in the experiments.

The learning results were significantly worse when observed from the intermediate layers of the model. Here evidence for the original hypothesis whereby the model structure would induce it to encode information related to, but not directly corresponding with, the contrastive pattern differentiation task, was at most very weak. Permutation tests and K-means clustering results showed that the model had encoded some informa-

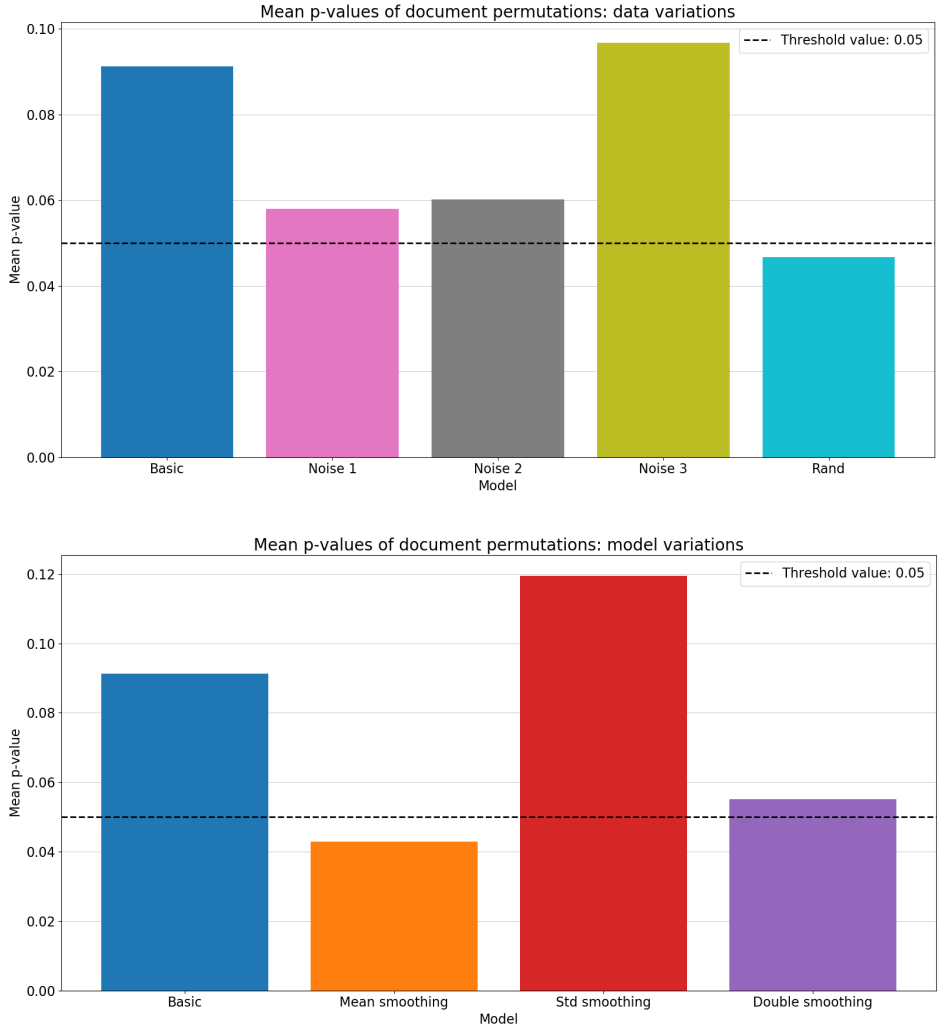


Figure 8.13: Averaged p-values from permutation tests performed to document vectors using different data and model variations. The dotted vertical line marks the threshold value $\alpha = 0.05$.

tion relevant for distinguishing the the timepoint vectors based on their binary labels into the vector values, but possibly due to the weakness of these impacts they were not visible in the t-SNE visualizations. The document representations, which in the model pipeline are closer to the input data, gave in practice no indication of differentiation on the basis of the timepoint labels, and only very weak signs of having encoded information related to the discourse categories. The use of different exploration methods enabled a diverse investigation of the learning results, and in most cases the results of the different approaches were also consistent with each other.

8.2 Timepoint classification

As discussed in Chapter 6 above, the model that has been trained with labeled data and contrastive loss function is used for classifying timepoints based on whether they belong

to an unstable discourse pattern or not. The following sections discuss the results of timepoint classification and pivot point detection in both supervised and unsupervised settings. In the supervised case, also the results of pattern based classification are presented.

8.2.1 Unsupervised timepoint classification

Binary classification of timepoints

The unsupervised classification methods, which are described in detail in Section 6.2, take advantage of the data encoded in the timepoint vectors by the trained contrastive model.

Figure 8.14 shows the accuracy and F-scores for unsupervised binary classification of timepoints based on whether they belong to an unstable discourse pattern. The differences between the model and data variations are small, and in each case the F-score is significantly lower than the accuracy score. This reflects the typical problem, discussed in Section 7.2, with using accuracy to evaluate classification results when the classes are unevenly distributed in the data.

The F-score, on the other hand, is the harmonic mean of precision and recall scores, which are measured here using the 'unstable' class as the positive category. The results can be concretized by using again the example from Section 7.2, where the sample timeline has 10 unstable and 90 stable timepoints. An F-score between 0.3 and 0.4 can be obtained for instance by a classifier that predicts all of the stable timepoints and 3 out of 10 unstable timepoints correctly, or, alternatively, by one that predicts all of the unstable timepoints correctly but in addition classifies 40 out of the 90 unstable timepoints as stable.

Pivot point detection

Figure 8.15 shows that the F-scores for unsupervised pivot point detection are very similar to those observed in timepoint classification. As described in Section 7.2.1, a margin of error $M = 5$ is used when calculating the precision and recall scores, so that a pivot point prediction that is less than 5 timepoints away from a true pivot point is counted as a true positive.

The Rand scores, which measure the similarity of the allocations of pairs of timepoints into stable and unstable segments in the true and predicted timelines, reach much higher level than the F-scores. When using a timeline with 100 timepoints, however, a good Rand score can be obtained even when the predicted pivot points are further than 5 timepoints away from the true ones. For a Rand score close to 0.7,

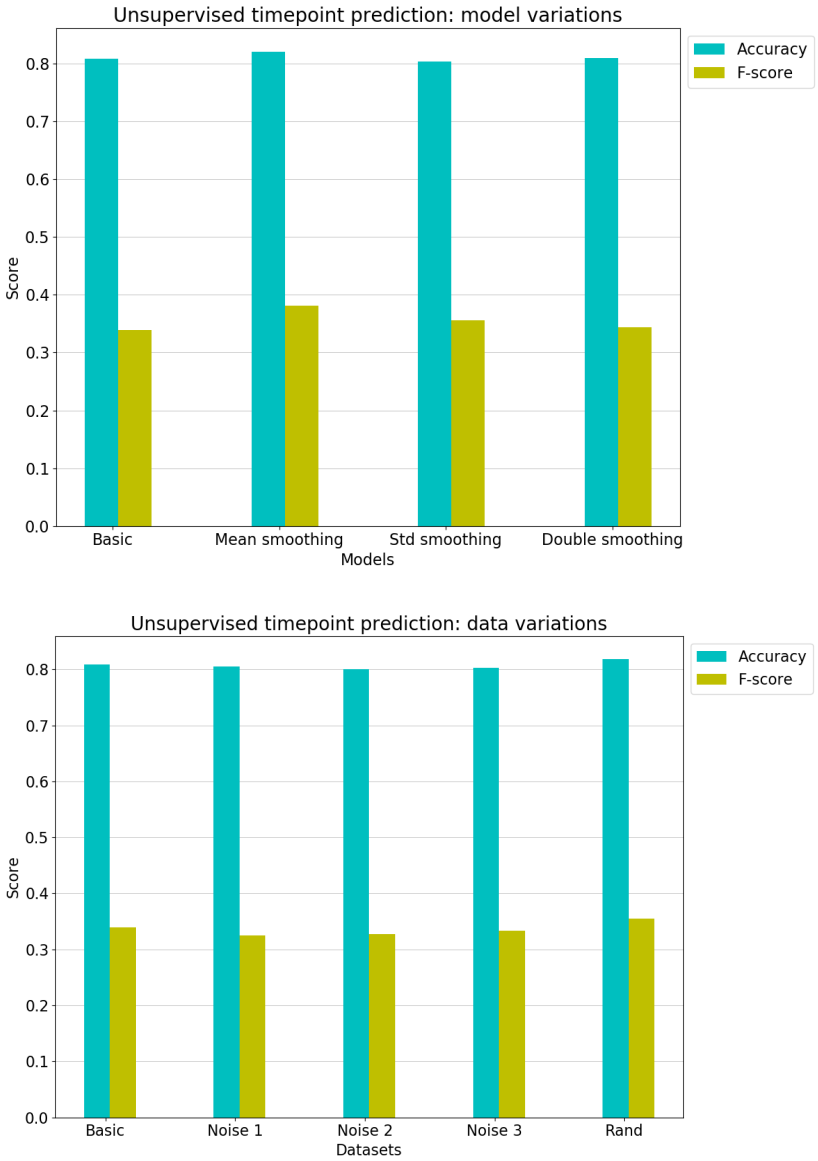


Figure 8.14: Classification accuracy and F-score for unsupervised timepoint classification with different model and data variations.

the numbers of true and predicted pivot points also do not have to match perfectly. These two factors are the most probable explanations of the difference between the two scores.

The results of unsupervised classification presented above indicate that contrastive pretraining has managed to encode some information to the timepoint vectors that can be used in downstream classification tasks. There is, however, much to be improved to reach a level where the classification results could be of practical use. Interestingly, the differences between the model and data variations are quite small. The conclusion that can be drawn from this is that the problem lies at a more fundamental level, in the ability of the contrastive pretraining model with the proposed structure

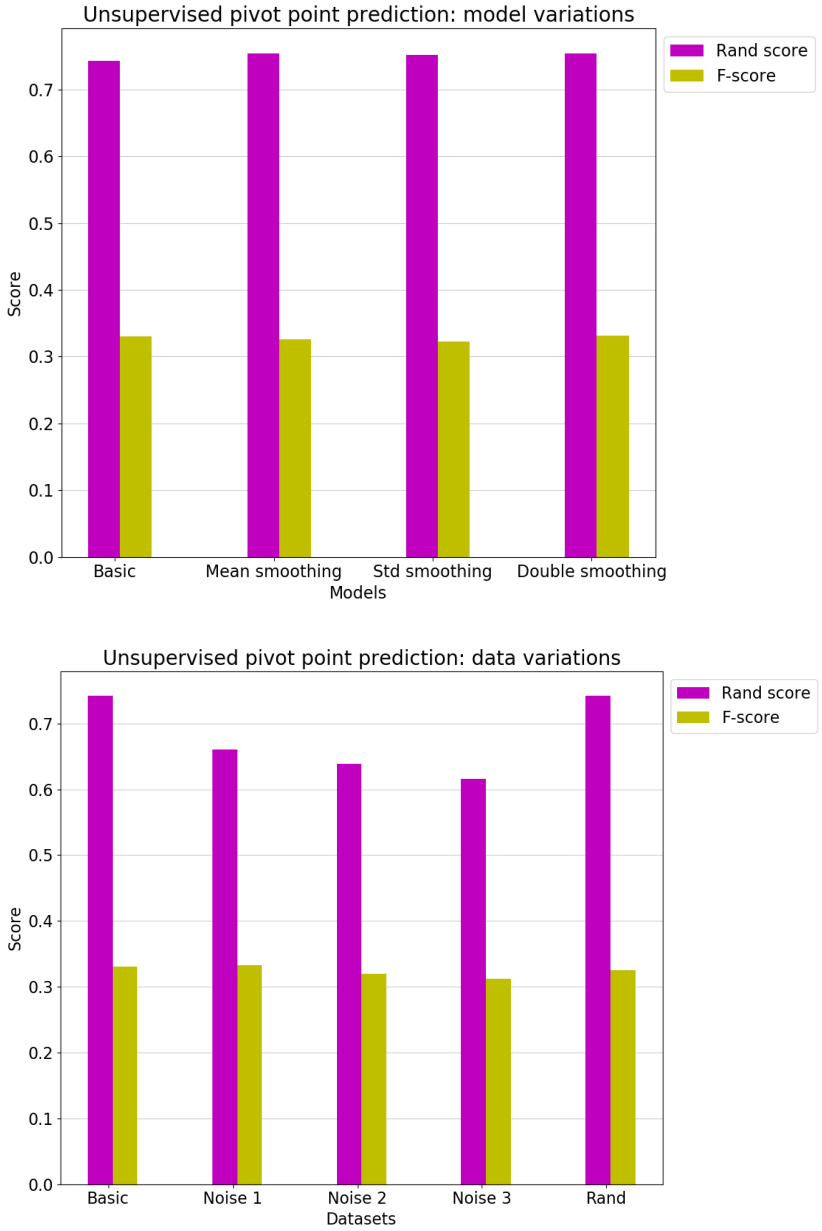


Figure 8.15: Adjusted Rand score and F-score for unsupervised pivot point detection with different model and data variations.

and training to encode enough relevant information into the timepoint vectors.

8.2.2 Supervised timepoint classification

Binary classification of timepoints

As was discussed earlier in Section 6.1, supervised classification is based on training an additional classifier element on top of the pretrained contrastive model, using either the binary timepoint labels or the pattern labels to measure the classification loss.

Supervised models were trained separately for all of the model and data variations,

both for the binary and for the multiclass classification setting. Figure 8.16 shows an example of the training, validation and test losses for the different model variations used for the binary classification task.

All of the models were trained for 20 epochs, and for all of the model and data variations the training losses are slowly but steadily declining. Validation losses display a bit more fluctuation, but the general downward trend is also visible here. The differences between the models are small, especially when comparing the test losses.

The timepoint classification and pivot point prediction results are evaluated using the same measures as in the unsupervised case. Figure 8.17 indicates that there are observable differences between the supervised and unsupervised timepoint classification results. In the unsupervised case (see Figure 8.14), classification accuracy is significantly higher for all model and data variations, while the F-score is at a slightly lower level. This can mean that the unsupervised classification model overestimates the number of stable timepoints, which results into good accuracy score, while failing to predict correctly unstable timepoints more often than the supervised model. When considering the fact that the supervised model was trained using the timepoint labels, unlike the unsupervised model, the difference between the two is surprisingly small. Neither model, however, reaches especially good results, as the F-score remains in all instances below 0.5.

Pivot point prediction

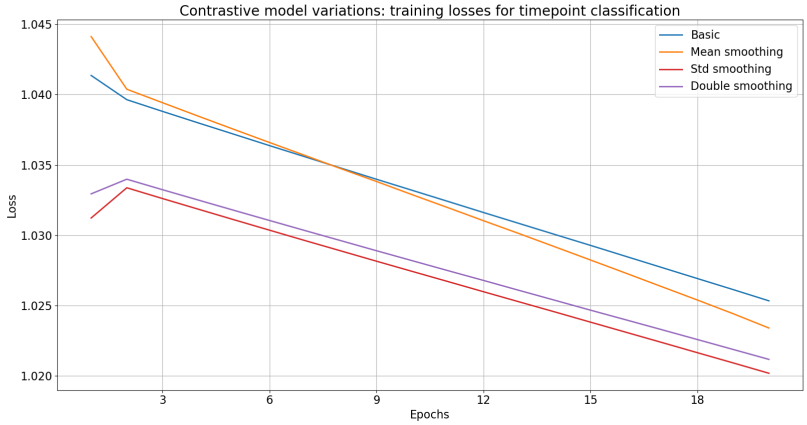
Pivot point prediction was also performed using the timepoint labels that are the output of the supervised model. The results, shown in Figure 8.18, are generally worse than the ones achieved when using unsupervised pivot point detection. The F-score is lower in all example cases, and the same applies also in most instances to the Rand score. The differences are large enough to support the conclusion that out of these two methods, the unsupervised model that uses a timeline of the averaged distances between the timepoint vectors is able to achieve better results in the pivot point detection task.

Discourse pattern prediction

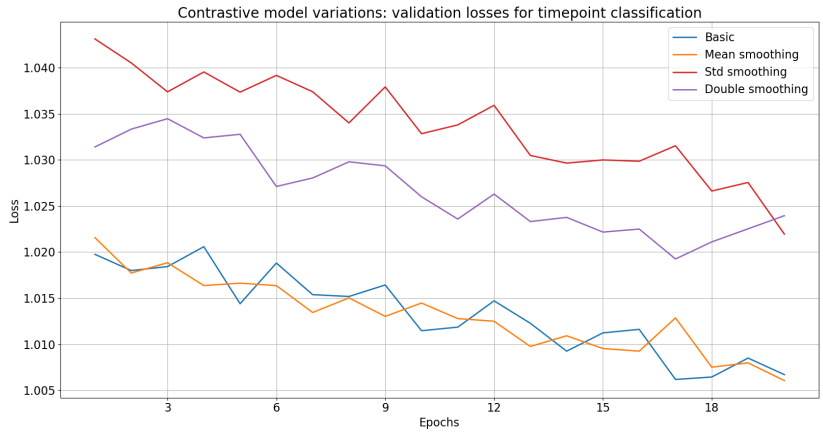
In supervised pattern-based classification, the classification layer of the model makes a prediction for the (unstable) discourse pattern of the input dataset, and the loss is calculated based on the pattern labels. As the classification task is closely aligned to the contrastive prelearing setting, the assumption is that the classification results should be better than in the timepoint classification case.

Figure 8.19 visualizes the training, validation and test losses for the different model variants. The differences between the models are again relatively small, with

(a) Training losses.



(b) Validation losses.



(c) Test losses.

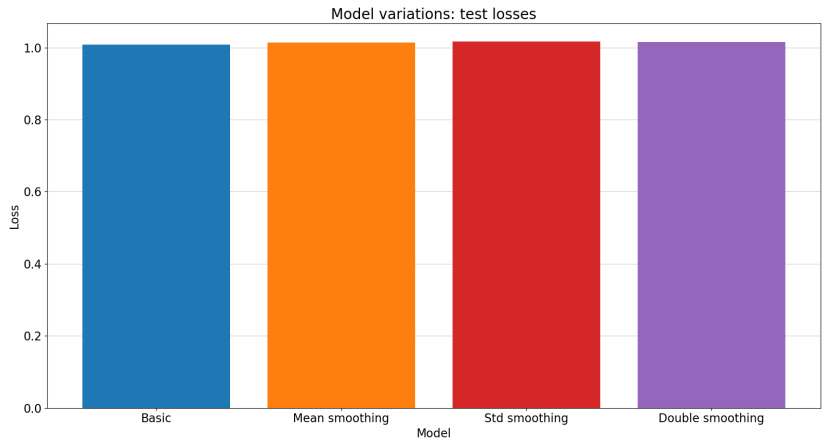


Figure 8.16: Training, validation and test losses for supervised timepoint classifier with different model variations.

the exception of the 'Mean smoothing' model, which performs significantly worse than the other models. The same difference can also be seen in the t-SNE illustrations of the contrastive model outputs, shown in 8.3, where the output vectors of the mean

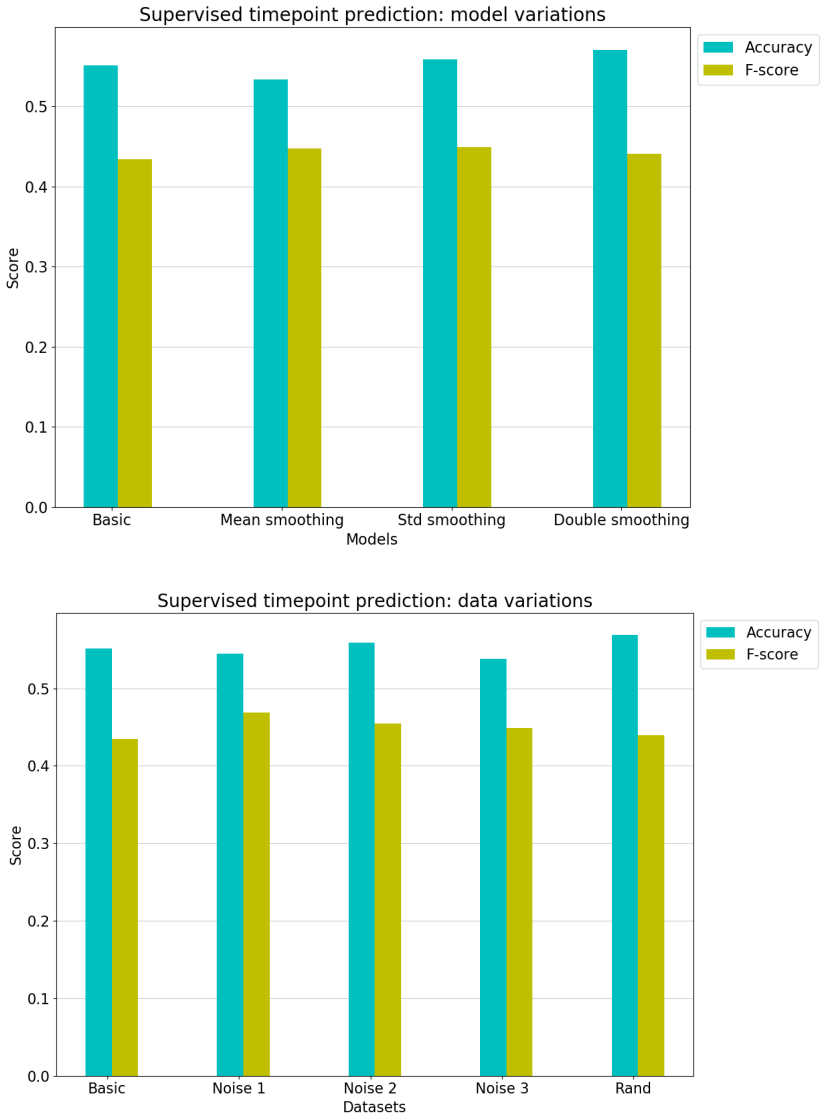


Figure 8.17: Classification accuracy and F-score for supervised timepoint classification with different model and data variations.

smoothing models are much less clearly clustered based on the pattern labels than is the case with any other model or data variant.

The poor performance of the 'Mean smoothing' model also stands out when comparing the test accuracies and F-scores of the model and data variants, shown in Figure 8.20. In other respects the results also align with the obervations made regarding the training results of the contrastive model. 'Double smoothing' model achieves best results of the model variants, and the data variants with the highest level of noise also get good scores. Noise, it appears, helps the model to generalise better to unseen data.

Overall the accuracy and F-scores for most of the model and data variants are higher than was the case with supervised timepoint classification. This was to be expected, as the contrastive pretraining was specifically used for differentiating the

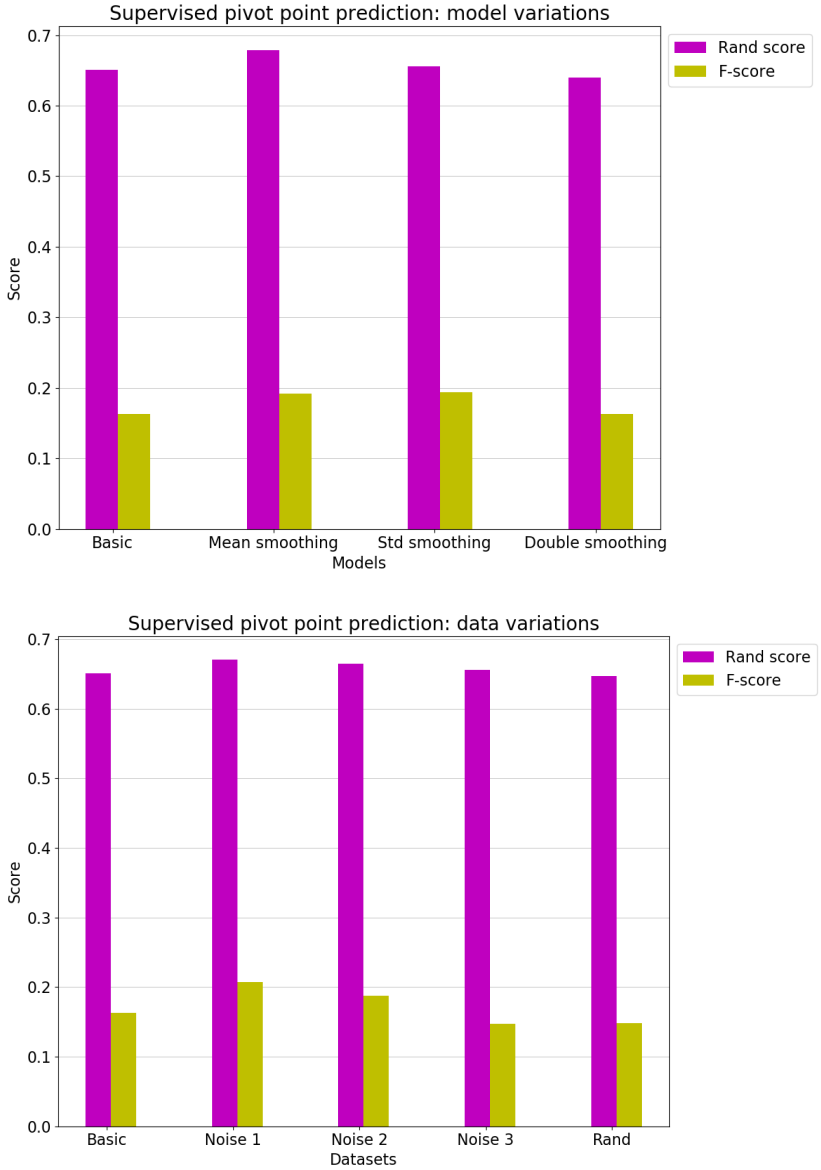
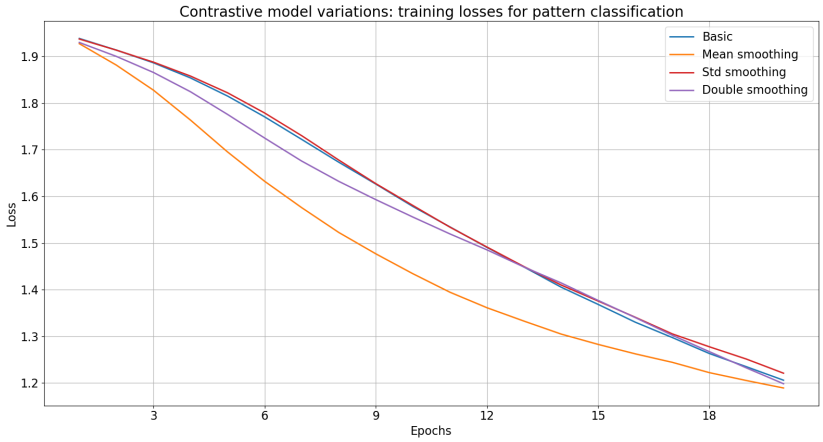


Figure 8.18: Adjusted Rand score and F-score for supervised pivot point detection with different model and data variations.

data representations based on the pattern labels. On the other hand, the differences in the results of the two downstream classification tasks are still relatively small, when considering the contrastive training objective. This is an indication of the ability of contrastive training to produce data representations that contain information relevant also for solving downstream tasks that are not directly derived from the contrastive learning objective. Although there is still plenty of room for improvement in the classification results, they do show that contrastive pretraining has potential to be used as part of a deep learning model for discourse change detection.

(a) Training losses.



(b) Validation losses.

(c) Test losses.

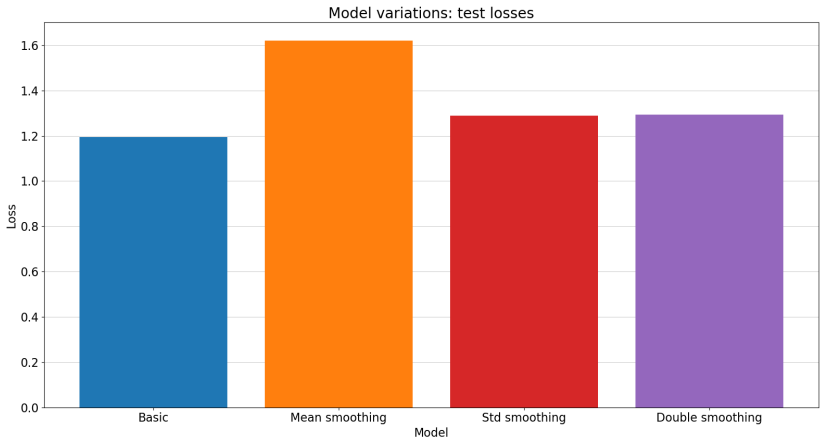


Figure 8.19: Training, validation and test losses for supervised pattern classifier with different model variations.

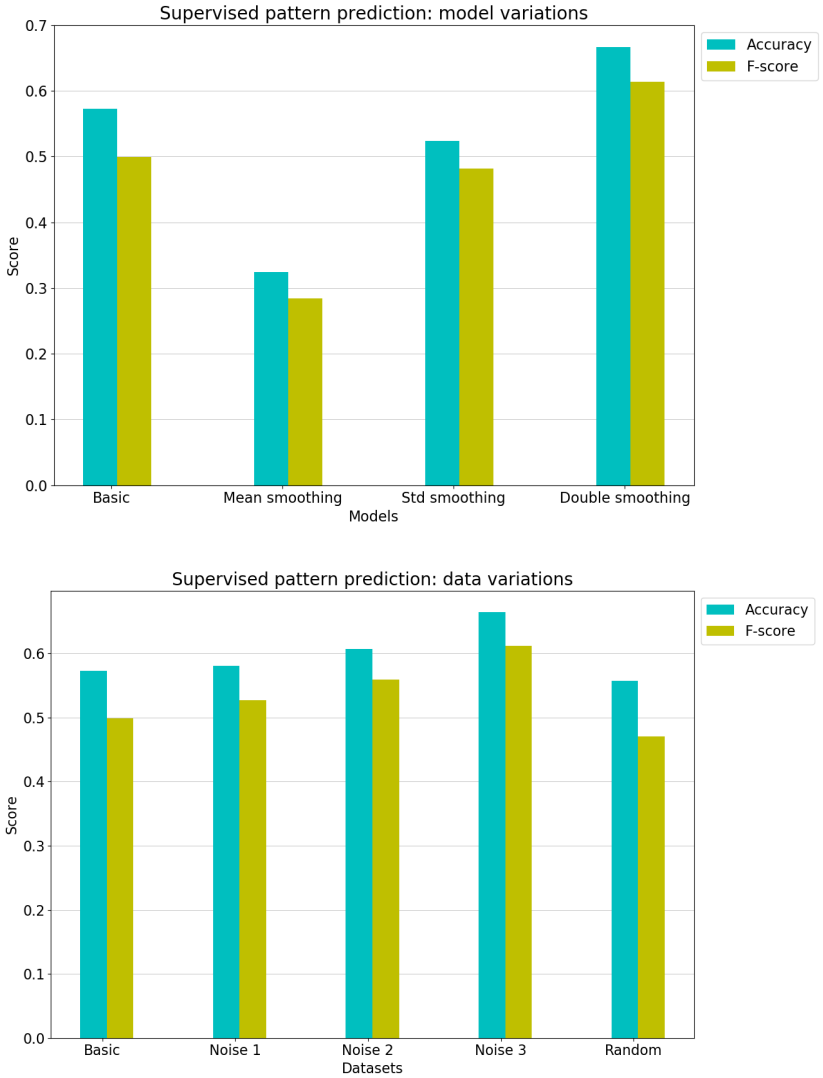


Figure 8.20: Classification accuracy and F-score for supervised pattern classification with different model and data variations.

9. Conclusions

Detecting discourses from large scale diachronic text corpora is a challenging task, and existing methods, such as topic models, used for the purpose are in many ways deficient. One alternative is to explore the potential of different deep learning techniques to extract information from texts that can then be used for locating changes in discourse patterns. This has been the approach taken in this thesis, which has introduced a deep neural network model that uses contrastive pretraining to learn patterns from the data that are then applied for discourse change detection.

The datasets are synthetically generated timelines of news articles, where the news categories are used as proxies for discourses, and the generated unstable discourse patterns are based on variations of the sigmoid and Gaussian functions. The use of synthetic data and the way that the model was built made it possible to formulate the discourse change detection objective as a classification task.

Variations of the synthetic datasets were created by injecting noise to the document frequencies as well as the categories used in data generation. Model variations applied different smoothing strategies to enrich the information passed to the LSTM layers of the contrastive neural network model. Contrastive models were trained separately for all data and model variations, and their learning results were evaluated by analysing both the model outputs as well as the performance of the models in the downstream classification tasks.

The training performance of the contrastive models and the analysis of the model output reveals that the contrastive learning of the discourse patterns is working, but only up to a certain point. The training and validation losses stop decreasing when the loss level is still relatively high, and while the best model and data variations can clearly separate four of the seven discourse patterns, the three remaining patterns appear to be indistinguishable to the models.

Analysis of the intermediate products of the contrastive model with the help of t-SNE visualisations, K-means clustering, and permutation tests showed that the model is able to produce timepoint level representations of the data that reflect to a limited extent the classification of timepoints based on whether they are part of an unstable discourse pattern.

On the other hand, the document level representations could not be distinguished based on their timepoint affiliation, and there were at most very weak indication of the results being better concering information related to the discourse categories.

The performance of the different model variations in the supervised and unsupervised classification and pivot point prediction tasks was likewise mixed. The results of timepoint classification showed signs that the contrastive pretraining had managed to encode information to the model parameters, and the intermediate timepoint vectors, that was useful for the classification task. Still, the overall classification performance was in both cases relatively poor. This is probably partly due to the above mentioned limitations of the contrastive learning model, and partly also to the limited generalizability of the pattern-based contrastive learning to the timepoint level classification using intermediate products of the model.

One interesting result from the experiments was obtained from the comparison between the supervised and unsupervised classification results. The performances of the two approaches were very close to each other, and the unsupervised model can even be said to have been superior to the supervised one in pivot point prediction. This indicates that there is relevant information encoded in the timepoint level representations, and it is also a positive sign when considering the possible uses of this approach to discourse change detection with real world data.

Considering that the models presented in this thesis were trained and tested using synthetic datasets that are much simpler than most 'real world' diachronic text corpora, there is still much to be done to reach a level of performance that would make the model competitive with existing approaches, like the one presented in Section 2.3. Another challenge is posed by the use of supervised learning in the contrastive pretraining stage. The results of the downstream tasks using synthetic datasets indicate that contrastive pretraining would need to significantly improve its performance in order for this approach to generalize into non-synthetic data.

An alternative option would be to develop a self-supervised method for the contrastive training stage. This is, however, not an easy task in the NLP domain, in particular when dealing with basic units that are not words or sentences, but documents. Whether pretext tasks or proxy labels, both familiar from the computer vision context, could be used to achieve this, remains an open question. In any case, knowing some of the possibilities and challenges related to the contrastive learning approach as presented above in the supervised setting can be helpful when designing alternative methods for discourse change detection.

Bibliography

[1] F. Bianchi, S. Terragni, and D. Hovy. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 759–766, Online, Aug. 2021. Association for Computational Linguistics.

[2] D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022, 2003.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[5] A. B. Dieng, F. J. Ruiz, and D. M. Blei. The dynamic embedded topic model. *arXiv preprint arXiv:1907.05545*, 2019.

[6] R. Dror, L. Peled-Cohen, S. Shlomov, and R. Reichart. Statistical significance testing for natural language processing. *Synthesis Lectures on Human Language Technologies*, 13(2):1–116, 2020.

[7] Q. Duong, L. Pivovarova, and E. Zosa. Benchmarks for unsupervised discourse change detection. In *HistoInformatics 2021–6th International Workshop on Computational History*, Online, 2021. CEUR Workshop Proceedings.

[8] A. Galassi, M. Lippi, and P. Torroni. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4291–4308, 2020.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

[11] A. Jaiswal, A. Ramesh Babu, M. Zaki Zadeh, D. Banerjee, and F. Makedon. A survey on contrastive self-supervised learning. *Machine Learning*, 12:4182–4192, 2020.

[12] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[13] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[14] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, (01), 2021.

[15] Y. Liu, A. Medlar, and D. Glowacka. Statistically significant detection of semantic shifts using contextual word embeddings. In *Proceedings of the 2nd Workshop on Evaluation and Comparison of NLP Systems*, pages 104–113, 2021.

[16] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[17] N. Rethmeier and I. Augenstein. A primer on contrastive pretraining in language processing: Methods, lessons learned and perspectives. *arXiv preprint arXiv:2102.12982*, 2021.

[18] J. M. Santos and M. Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, pages 175–184. Springer, 2009.

[19] H. Trappes-Lomax. Discourse analysis. In A. Davies and C. Elder, editors, *The handbook of applied linguistics*, chapter 5, pages 133–164. Blackwell Publishing Malden, MA, 2004.

[20] C. Truong, L. Oudre, and N. Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

[21] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[23] I. Vayansky and S. A. Kumar. A review of topic modeling methods. *Information Systems*, 94:101582, 2020.

[24] Y. Yu, X. Si, C. Hu, and J. Zhang. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, 31(7):1235–1270, 2019.