Master's thesis

Master's Programme in Computer Science

Discourse change detection in diachronic text collections with synthetic datasets and neural networks

Quan Duong

June 14, 2021

Faculty of Science

University of Helsinki

Supervisor(s)

Dr. Lidia Pivovarova

Examiner(s)

Dr. Lidia Pivovarova, Prof. Hannu Toivonen

Contact information

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

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Master's Programme in Computer Science

Tekijä — Författare — Author

Quan Duong

Työn nimi — Arbetets titel — Title

Discourse change detection in diachronic text collections with synthetic datasets and neural networks

Ohjaajat — Handledare — Supervisors

Dr. Lidia Pivovarova

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	June 14, 2021	48 pages

Tiivistelmä — Referat — Abstract

Discourse dynamics is one of the important fields in digital humanities research. Over time, the perspectives and concerns of society on particular topics or events might change. Based on the changing in popularity of a certain theme different patterns are formed, increasing or decreasing the prominence of the theme in news. Tracking these changes is a challenging task. In a large text collection discourse themes are intertwined and uncategorized, which makes it hard to analyse them manually.

The thesis tackles a novel task of automatic extraction of discourse trends from large text corpora. The main motivation for this work lies in the need in digital humanities to track discourse dynamics in diachronic corpora. Machine learning is a potential method to automate this task by learning patterns from the data. However, in many real use-cases ground truth is not available and annotating discourses on a corpus-level is incredibly difficult and time-consuming. This study proposes a novel procedure to generate synthetic datasets for this task, a quantitative evaluation method and a set of benchmarking models. Large-scale experiments are run using these synthetic datasets. The thesis demonstrates that a neural network model trained on such datasets can obtain meaningful results when applied to a real dataset, without any adjustments of the model.

ACM Computing Classification System (CCS)

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

Computing methodologies → Artificial intelligence → Natural language processing → Information extraction

Information systems → Information retrieval → Evaluation of retrieval results → Presentation of retrieval results

Avainsanat — Nyckelord — Keywords

discourse dynamics, trends detection, neural networks, quantitative evaluation, synthetic data

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Algorithms, Data Analytics and Machine Learning subprogramme

Contents

1 Introduction

Large collections of text, such as news archives, reflect valuable information on *discourse dynamics* – the change in prevalence of certain topics, opinions, and attitudes over a period of time. This is a valuable source of information in digital humanities and computational social sciences. Various natural language processing (NLP) methods, from keyword extraction to topic modelling, have been established to facilitate discourse analysis. However, studying discourse dynamics is a challenging research area that still needs to be developed.

This study tackles a problem of automatic detection of discourse change in diachronic textual data. Input data for this study is a collection of texts, split into multiple time periods. The main focus for this study is providing a method for automatic discourse change detection rather than investigating any particular use case. In this study scope, the discourse can be defined as a theme extracted from articles distribution across a bounded time range. For example, the articles about the theme World War II were very popular during the conflict time and become less popular after a few decades.

In digital humanities, research questions are generally complex and involve a lot of uncertainty, thus the ground truth needed for quantitative evaluation is usually unavailable. Therefore, evaluation is the primary concern in this research. Moreover, quite often digital humanities research deals with a specific use case, applying a method designed from one annotated dataset might not fit to another non-annotated dataset.

To overcome this difficulty, this thesis proposes a framework to generate multiple synthetic datasets and to provide a qualitative evaluation approach. In this framework, various computational models are introduced as methods to detect the changing patterns of the discourse as well as the pivot points of the changing periods.

1.1 Motivation

In digital humanities research, detecting interesting discourses which contain changing patterns in a huge text corpus could bring a lot of benefits. With a large amount of data from different sources like historical newspapers or public speeches, it would take a lot of work to read and link the important information to illustrate the big picture of an event.

For example, if we want to find out how a discourse theme was changing during the 70s, we need to search and analyse the related documents during that time. It is even more intricate if the question is "when did people's interest in discourses about religions like Catholicism or Buddhism start increasing then decreasing?". A lot of valuable information might be omitted if we only use subjective keywords to collect the data. Such as when searching about war topics with the keywords like 'weapon' and 'casualty', we might forget to include the keywords related to food, which is a momentous topic in the crisis reported outside the war region. Moreover, with the growing data like news streams, social media, manual analysis is not realistic for humans. It is more efficient if the information is purposefully filtered before being analyzed by experts for specific intents later. However, automating this process is not always an easy task. Firstly, the textual data contain a lot of noise and ambiguous information. Secondly, applying unsupervised methods, it is hard to have an objective evaluation process. Though, using supervised methods requires annotated dataset which is lacking for this task at the writing time. Manually annotating the data is extremely time consuming and requires domain expertise. Finally, the big volume of data is also a challenge with computing resources.

Automatic discourse change detection is motivated by the needs of humanities scholars but the point of view is methodological—this study proposes a framework includes three parts: a synthetic data simulator for generating training and test data with annotated temporal changes in their themes, computational models for supervised learning to detect temporal changes in themes, and an evaluation method to benchmark the quality of the models.

The main contributions of this study are the following:

- Draw attention to a discourse dynamic detection task that is relevant for humanities and computational social sciences but has been less studied within the NLP field.
- Establish a novel framework for discourse change detection and perform a large-scale experiment on a set of thousand synthetic datasets created to emulate six different patterns of discourse change.
- Propose several benchmark methods to tackle the problem. The best-performing method yields 78% accuracy.
- Finally, perform a qualitative evaluation on a separate (unannotated) news corpus and demonstrate that a proposed method is able to find discourse change in a large news stream.

1.2 Related Works

Discourse dynamics has been a topic of several multidisciplinary studies that apply NLP to historical or social science research questions. Quite often these studies lean on topic modelling [37, 18, 40, 23], though others use techniques, such as language models and clustering [11, 14]. Each of these studies deal with a complex research question, such as "immigration discourse" or "nation building", and the suitability of the applied methods is assessed only qualitatively, using close reading or background knowledge of the field.

There have been several attempts within the NLP field to model discourse change, by the means of unsupervised topic models, such as dynamic topic models [1] or Topics over Time [38]. More recent models make use of word embeddings and neural inference networks to learn topics from data streams [5, 17]. However, even papers proposing these models often rely on use cases rather than numerical evaluations. As a result, the general validity of the models remains unclear especially for research questions that go beyond localizing well-known historical events in time. Any model has certain limitations, that are rarely articulated [20]; quite often a basic Latent Dirichlet Allocation model is preferred to more sophisticated models [9].

Another task relevant to diachronic change is lexical semantic change detection, which recently got a boost by leveraging word embeddings [15, 34]. In this task, manual data annotation is extremely challenging [28], and the datasets are rather small and scarce. Thus, synthetic datasets commonly used [36, 29, 32, 27]. Currently the usual approach is to merge two words with different meanings into one pseudo-word and then sample from their contexts according to some predefined distribution. A model is then evaluated by its ability to recognise the distribution.

This study is positioned in between the aforementioned fields. A novel framework for automatic discourse change detection is proposed as a reference method for creating synthetic data, training models and quantitative evaluation. The evaluation procedure is based on extensive experiments on multiple synthetic datasets, an approach adopted from the closely related task of lexical semantic shift detection.

1.3 Problem Statement

The term "discourse" has many definitions across humanities and social disciplines; it could be understood either as a property of a corpus as a whole or a property of a single text and its structure [25]. In this study, the discourse is considered as a body of text meant to communicate specific data, information, and knowledge about a subject or a theme which is existing implicitly in the corpus. A fine-grained structure of particular documents is irrelevant for our research question and ignored in the experiments. The discourse change could only be found in *diachronic corpus*, i.e. corpus that contains data from several consecutive time periods.

Thus input for our methods is a collection of texts, split into multiple time periods. The task breaks up into these following **sub-tasks**:

- To define the changing patterns by computational formulas for synthetic data generation.
- To sample suitably the synthetic datasets from existing dataset for both training and evaluation steps.
- To build computational models for:
  - finding *a subset of documents* that belong to a theme to build time-series.
  - detecting whether a certain discourse (theme) is *non-stable*, e.g. increasing or decreasing pattern.
  - finding *pivot points* in the time-series, i.e. time points where non-stable behaviour starts and ends.
- To define evaluation metrics which reflect the practical meaning for the outcome.

As stated in the motivation section 1.1, finding training and evaluation data for this task is currently not possible because, according to subjective knowledge, there does not yet exist diachronic corpora annotated with discourses. Moreover, it is difficult to produce an annotated corpus for several reasons. First, annotation on a discourse level requires a lot of effort and can only be done by someone with a thorough knowledge of the corpus. Second, it is difficult for a human to distinguish between an actual change in the data and noise, and especially difficult to find a concrete pivot point, apart from some obvious cases. Third, since the task is defined on a corpus level, supervised learning would require

annotation of multiple corpora, which is not feasible. Thus, all our models are trained and tested on synthetic data, while their applicability to real-world use cases is demonstrated qualitatively.

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

1. Chapter 2 introduces the theoretical background of natural language processing, clustering, topic modeling and deep learning algorithms.
2. Chapter 3 describes the text corpus used in the study and the construction of the synthetic data.
3. Chapter 4 represents the methods to detect the changing patterns and pivot points on synthetic data.
4. Chapter 5 provides various evaluation metrics to compare the performance of models from the previous chapter. The results of all models are also analyzed in this chapter.
5. Chapter 6 experiments the trained models on realistic data. The first dataset is in Finnish and the second one is in English. The results are manually analyzed to show interesting finds.
6. Chapter 7 summarizes this thesis and discusses future work.

2 Background Theories

In this chapter, we will review the theoretical background necessary for understanding the methods discussed in the following chapters. First, we discuss in detail unsupervised learning methods which include Doc2Vec, K-means clustering and Latent Dirichlet Allocation (LDA). Next, we describe the neural networks algorithms consisting of recurrent neural networks (RNN) and convolutional neural networks (CNN).

2.1 Doc2Vec

Document originally stored in text format is convenient for humans to read, but might be a challenge for computing tasks. Transforming from character to fixed length numeric representation is helpful for many purposes, for example: document retrieval, semantic comparison. One vector representation of text has been introduced by Harris [10] as Bag Of Words method. Even though this method is easy to compute and shows the efficiency in many cases, there are still some weaknesses such as lacking the word order and suffers from data sparsity and high dimensionality. It is also missing the semantic meaning between the words, for example, the words "dog" and "cat" are more similar than "dog" and "car" but they are treated equally in the Bag of Words method.

Doc2vec is a document embedding algorithm that comes to solve the issue from Bag of Words. This section will review the work from Q. Le and Mikolov [16] to explain Doc2Vec in detail.

To understand how Doc2Vec works it would be easier to explain the concept of Word2Vec algorithm first. The algorithm was introduced by Mikolov et al. [21] with the idea of representing a word as a vector of real numbers in a vector space. In this space, the embedded vectors with their distances reflect the semantic relationship. Back to the previous example, the word "dog" and "cat" are in closer position in the vector space compare to the word "car". There are ways of training the Word2Vec model: Continuous Bag-of-Words (CBOW) and the Skip-Gram.

In the CBOW model, a sliding window is created for a sentence to predict a next word from its surrounding words (context). Contrarily, the Skip-Gram idea is to use the current

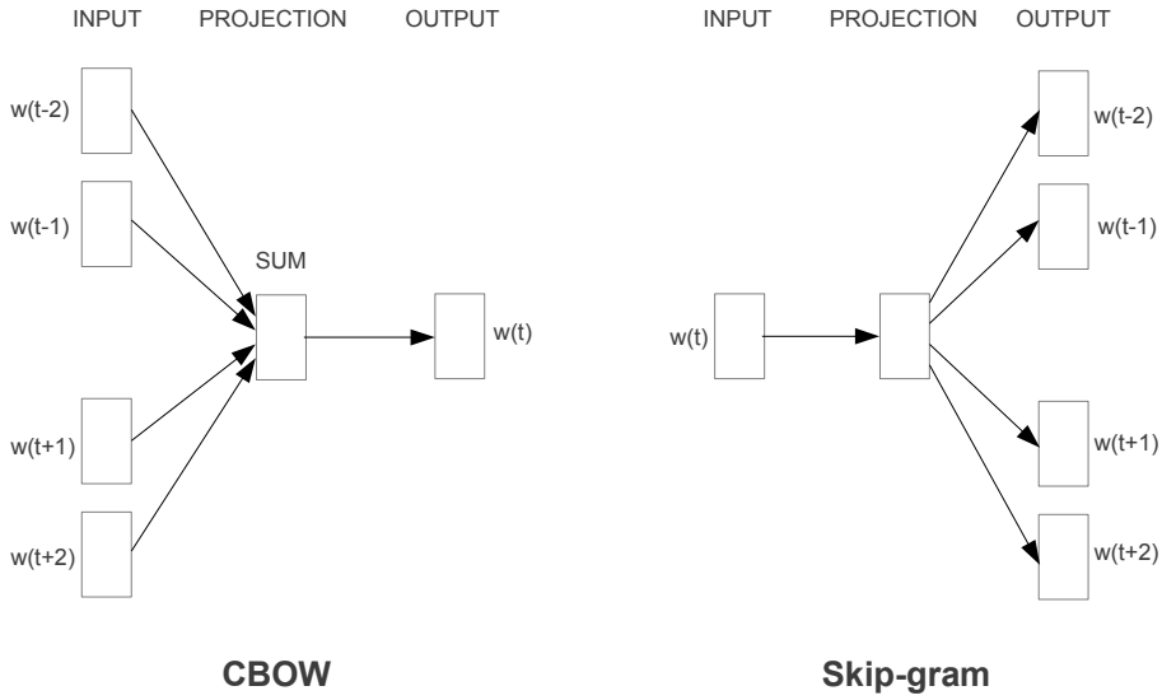

Figure 2.1: Word2Vec models: CBOW predicts next word using context words and Skip-Gram using current word to predict the surrounding words with a small window size. Source: [21].

word to predict the context words. This makes Skip-Gram slower to train but give better performance on infrequent words. The figures 2.1 adapted from Mikolov et al. [21] study shows the CBOW and Skip-Gram models architecture.

The model is trained to predict the words in the sliding window and thus can capture the semantic relationship of the words. In this sliding window, the words appearing in the same context more frequently have closer semantic meaning and opposite for the words are rarely in the similar context window. The output for each word in embedding layer is a dense vector which can be later used for downstream tasks.

The concept of Doc2Vec is very similar to Word2Vec with a little adjustment. Instead of representing a word as a vector, it vectorizes a whole text document regardless its length in words. The idea of training a Doc2Vec model is inherited from Word2Vec. Each document or paragraph is assigned an unique ID. In the first model, Q. Le and Mikolov [16] combine a word's context with the paragraph ID to predict the current word. This model learned by the standard paragraph vector with distributed memory (PV-DM). We can see this is an extension of CBOW model in Word2Vec with additional paragraph ID as an extra context vector. In the second model – PV-DBOW – the same idea is applied

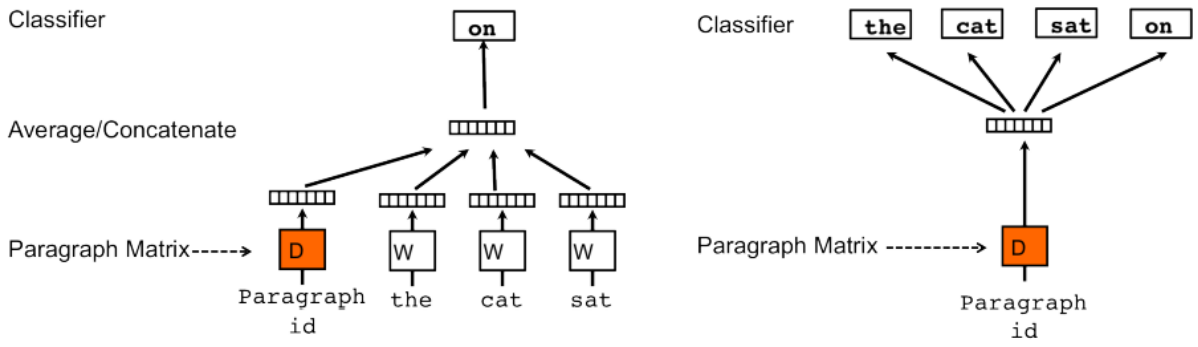

Figure 2.2: Left side: PV-DM architecture predicts next word using context words and paragraph ID. Right side: PV-DBOW predicts the surrounding words by only paragraph ID. Source: [16].

for extension of Skip-Gram where the paragraph ID is used to predict the context words. Both architectures are visualized in figure 2.2.

In practice, the Doc2Vec model can be trained to output both word vectors and document vectors. A softmax layer with trained weights is used for an unseen document's inference.

2.2 K-means

The K-means is a clustering algorithm which is used to partition data into groups. These groups or clusters objective is to contain data points that are more similar to other points in their cluster than in other clusters. This study will explain K-means based on the work from Lloyd [19] as reference. K-means algorithm uses Euclidean distance metric to measure how relevant an entity to its cluster center point, which is called centroid. K-means seeks to minimize the distance between centroids and the members associated with those centroids. That way it attempts to find the hidden structure in a dataset. In reality, the number of clusters k is unknown. This number need to be set manually for algorithm to operate. Thus, K-means will always give an output, however, this output might not always reflect the real structure of the dataset. The reason for that is the way K-means defines the similarity might not the same as we define the similarity in a particular problem.

The benefit of using K-means is due to its simplicity and flexibly for applying to various data types. The K-means algorithm is a clustering method defined by its partitioning based or non-hierarchical methodology. Given a set of objects X, K-means divides up every data point x_i into a cluster c_j (where $j \leq k$ and k is a parameter that defines a

number of clusters) that minimize the contained group sum of squared errors. Clustering is done to minimize the objective function where μ_j is the centroid of cluster c_j:

$$\sum_{j=1}^{k} \sum_{x_i \in c_j} \| x_i - \mu_j \|^2$$

To demonstrate in detail, given a data set of items with certain features values, the task is to categorize those items into k groups. The clustering steps work as follow. Firstly, we initialize randomly k points as the cluster centroids. Next, we calculate the Euclidean distance from each item to all centroids and assign it to the closest one. After that, the centroid of each cluster is updated by averaging the position of the items in each cluster. The memberships of items are again re-assigned to the closest new centroid position. The process of assigning and re-assigning membership and updating the centroids is repeated until there is no more change in all cluster centers coordinate, which is also known as convergence.

2.3 Latent Dirichlet Allocation

Topic modeling is the process of discovering topics in a set of documents. The "topic" in topic modeling is an abstract semantic structure which are represented by a collection of words with similar expressions that best characterize a set of documents. There are many applications like recommender systems or search engine where knowing the topics of documents is important. One popular algorithm for topic modeling is Latent Dirichlet Allocation (LDA). This section will discuss about LDA based on the work from Blei, Ng, and Jordan [2].

LDA is unsupervised probabilistic algorithm. The intuitive idea of LDA is that it sees documents as bags of words with an assumption: document was formed by a set of topics where in each topic there is a set of relevant words. In order to find the hidden topics, LDA counts co-occurrences of words in documents, which groups similar word patterns to infer topics within unstructured data.

More concretely, LDA builds a probabilistic model for observed documents so that each document is associated with a probability distribution over a set of topics, and topics are associated with a probability distribution over the set of words. This means if two documents sharing the same topic, they also share a set of similar words. The way LDA works is explained as below:

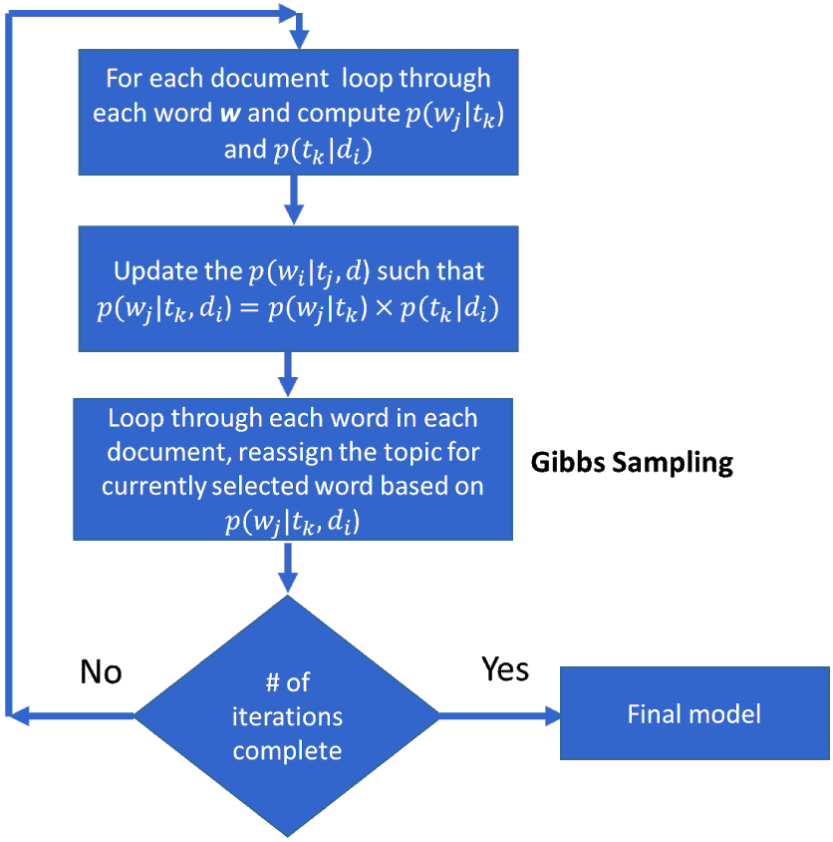

Figure 2.3: The steps to update LDA model. Where $p(w_j|t_k)$ is proportion of all documents assigned to a topic t_k for a given word w_j and $p(t_k|d_i)$ is proportion of words in document d_i that are assigned to topic t_k. Source: https://www.mygreatlearning.com/blog/understanding-latent-dirichlet-allocation/

Given a predefined K topics, we go through each document and assign each word in the document to one of k topics randomly. Next, we compute the probability of a word for a given topic as illustrated in figure 2.3. Note that LDA begins with random assignment of topics to each word and gradually improves the assignment of topics to words through Gibbs sampling.

2.4 Recurrent Neural Network

A recurrent neural network (RNN) is a type of deep neural network which handles sequential data. This deep learning algorithm and its variants are commonly used for ordinal or temporal problems, such as machine translation, stock market prediction, and image captioning [33, 26, 39]. As a supervised learning algorithm, RNN utilizes training data to learn. In a vanilla neural network (feed forward) architecture, the model takes a fixed

size vector as an input, which limits its usage in situations that involve a series type input with no predetermined size. Even in combining multiple feed forward networks to handle the serial data, the relationships of time steps are not considered. The RNN architectures came to solve this issue by providing a "memory" mechanism for managing the temporal relationship between input time steps. The general idea is that information from prior inputs is brought to influence the current input and output.

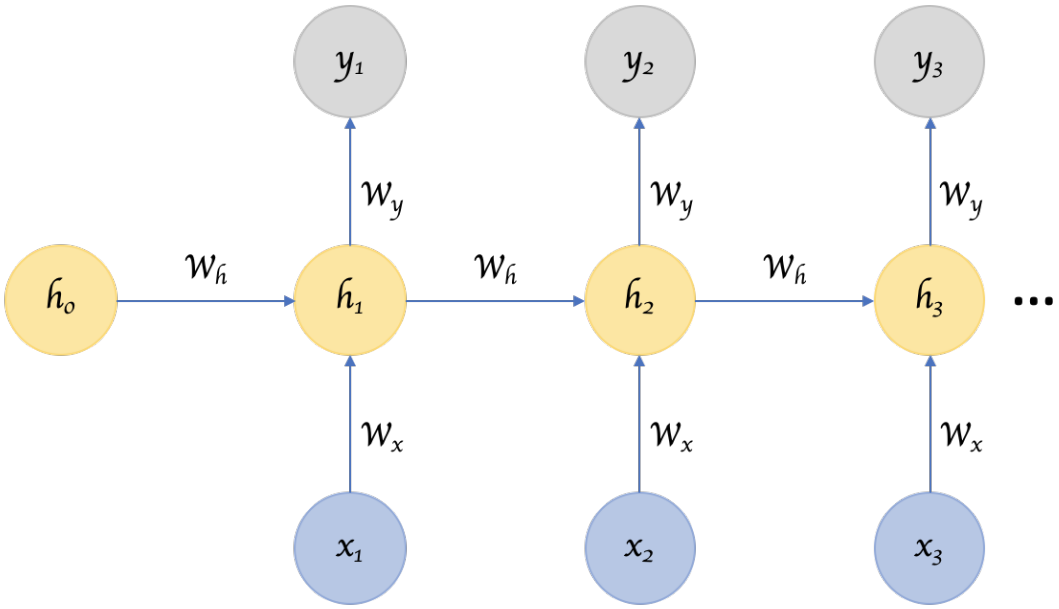

Figure 2.4: Recurrent Neural Network takes a series input and produces an output for each time steps. During the training, the previous time step information is stored in memory cells or hidden units. Image source: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

The figure 2.4 shows the unfolded RNN architecture in K time steps which carries out an input sequence of vectors $[x_1, x_2, ..., x_k]$. The hidden layer is shared among the network and maintains a hidden state h. For each time step t, the input x_t is combined with information in the hidden layer at the state $h_{(t-1)}$ to update the h_t and output the y_t. By doing that, the current time step has dependency information from the previous time step, or it is able to "remember" the pass.

The advantages of RNN model are possibility of processing input of any length and the size of model not increasing with size of input. However, training RNN models might take a longer time because the time steps are processed sequentially. Another drawback of RNN is that, in practice, it is only able to remember the very recent information and forgets the information from earlier time steps. It is due to vanishing or exploding gradients problem in the back-propagation process, when multiplying gradients that are smaller or

larger than 1 can be exponentially decreasing/increasing with respect to the number of time steps. For example, to guess the next word in the sentences: "If it is rainy, please don't go out without bringing an...". It is obvious to select the *umbrella* as a suitable word if the model is aware of the word *rainy* word. However, due the far distance in the long sequence, the information of the very first words tends to be omitted.

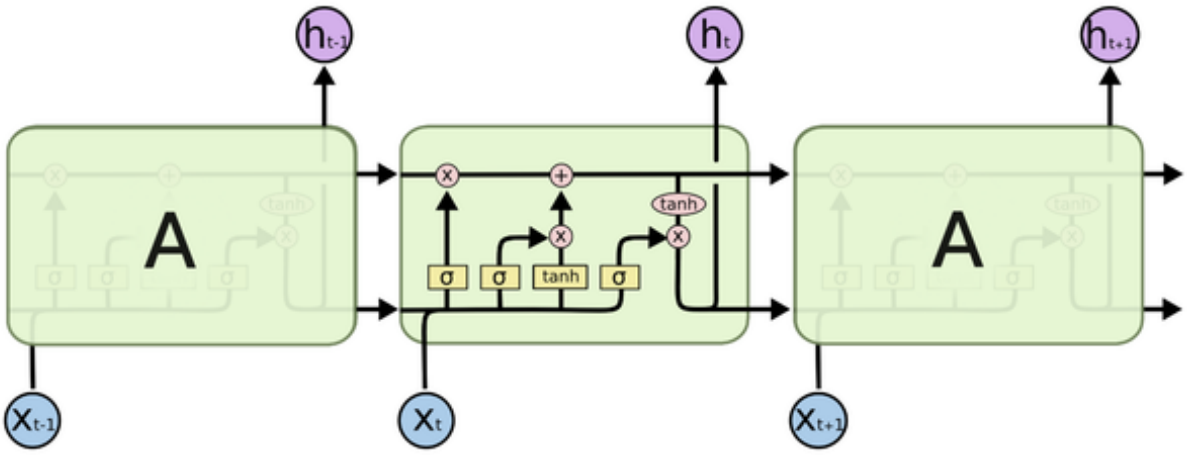

Figure 2.5: Long short-term memory cell architecture. Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Fortunately, there are variants of RNN designed to overcome this issue such as GRU (gated recurrent units) or LSTM (long short-term memory). The LSTM neural network was introduced by Hochreiter and Schmidhuber [12], which was designed to avoid the long-term dependency problem. This thesis will not dive deep into how LSTM works, but the core idea is the cell state memory management in the model. As shown in figure 2.5, in a LSTM cell, the information is passed through different gates where relevant information is kept and unimportant one is removed. This helps LSTM to maintain a longer dependency compared to RNN.

While future events would also be helpful in determining the output of a given sequence, unidirectional RNN cannot account for these events in their predictions. Schuster and Paliwal [31] propose bidirectional RNN (Bi-RNN) architecture to take the future events in advance. Bi-RNN is basically a stack of two RNN layers but with one layer taking the input in reserved a way. This can be demonstrated as reading the sequence from two different directions from left to right and right to left to have a better view. The idea is also applicable to RNN families like LSTM. This study will focus on using Bidirectional LSTM (Bi-LSTM) architecture as the delegate for the RNN variants.

2.5 Convolutional Neural Network

A convolutional neural network (CNN) is another type of artificial neural network. While RNN is good at handling temporal data, CNN is designed to detect the visual patterns in spatial data such as images. CNN has many applications in the computer vision like object recognition, segmentation and image reconstruction. Even dominating in computer vision field, the CNN is also a powerful algorithm in solving NLP or signal processing problems. For instance, Gao, T. Li, and Huang [8] applied CNN for text classification while Mustaqeem and Kwon [22] found it is successful for processing audio signals. This section will discuss CNN in detail based on O'Shea and Nash [24] study.

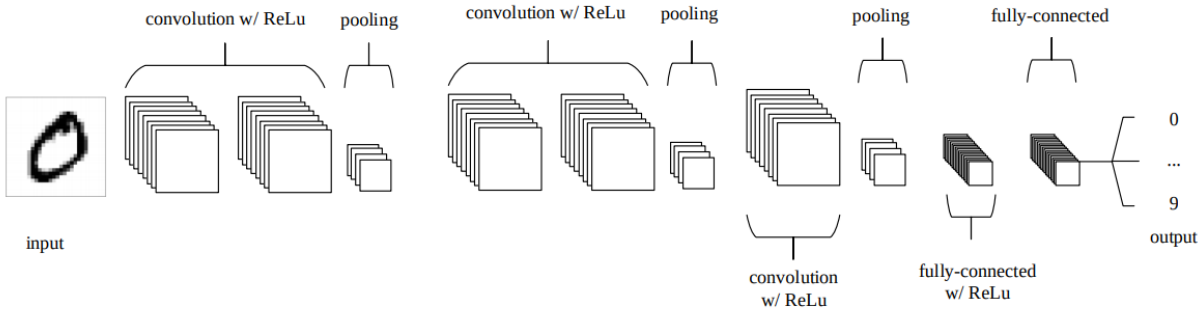

Figure 2.6: Convolutional Neural Network architecture. Image Source: [24]

The image of an object is complex and inconsistent due to the viewing context which is impossible to define as a whole without dividing into smaller patterns to recognize. Human brain is able to spot and recognize these patterns without having to re-learn the concept and identify objects no matter the angle we look at it. This behaviour encourages the design of convolutional neural networks, which is simulated to the connection pattern of neurons in the human brain. The key idea is inspired by the biological processes [13] of the visual cortex in animals. Intuitively, when you are looking at an image, individual neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. There are different receptive fields that respond to different patterns which overlap each other to cover the entire visual area.

The CNN architecture inherits this neurology idea to create different filters called kernels to capture the low level features such as edges, colors or gradient orientation of an image. The figure 2.6 demonstrates the process on how CNN learns these features from a 2D input such as an image. In a typical 2D CNN architecture, the input is a tensor with shape: N x H x W x C where N is the number of inputs, H is the height, W is the width and C is

the channels (colors). This input is then passed through the convolution layer which has convolutional filters or kernels used to map the abstract features of images. The feature map tensor has the shape: H x W x C where H is the kernel height, W is the kernel weight and C is the number of the new output channels or the extracted features from the input.

Multiple convolution layers could be stacked. The first layer learns low-level features. Then the next layer uses that learned information to extract the higher level features. For example, to classify a dog photo the first convolution layer can detect the edge of an object, the filters in the next layer are able to detect circle path or square path and then the final layer can recognize the more complex patterns like furs or eyes. This feature mapping process performs an efficient fitting to the image data due to the reduction in the number of parameters and re-usability of weights. The convolution layers also run in parallel, which is a huge advantage in computational time. To improve the processing time even more, the architecture introduced the Pooling layer which is responsible for reducing the spatial size of output features. By decreasing the input size for the next convolution layer, it helps in optimizing the computational power without losing the important features.

There are two types of pooling: max pooling and average pooling. The max pooling layer is used to extract the dominant features from previous layer output. The idea of max pooling is that it uses a kernel to cover an area of the image and return the maximum value from that portion. On the other hand, the average pooling layer returns the mean value of the current area. The output features from convolution layers and pooling layers are then passed to, for example, fully connected layers to have the final result, which depends on the specific problem.

In addition to a 2-dimensional CNN model, there are 3D or 1D CNN architecture with the same working mechanism but adapting to different input dimensions. This study will apply 1D CNN architecture with the details in the following chapters.

3 Synthetic Datasets Generation

3.1 Yle News Corpus

The synthetic datasets are created from a corpus of news articles published from 2011 to 2018 by the national Finnish broadcasting company Yle. The corpus is distributed through Finnish Language Bank (Kielipankki)[†] and is freely available for research use.

The Yle corpus contains more than 700,000 articles written in Finnish and published from 2011 to 2018. Each article belongs to one major category and one or more sub-categories. To create the synthetic datasets, only the articles that belong to well-separated major categories are taken, which is important for the quality of the data. We assume that a category in this corpus is presented as a discourse. There are 12 categories in the corpus that are suitable for this purpose: *autot* (cars), *musiikki* (music), *luonto* (nature), *vaalit* (elections), *taudit* (diseases), *työllisyys* (employment), *jääkiekko* (hockey), *kulttuuri* (culture), *rikokset* (crimes), *koulut* (schools), *tulipalot* (fires) and *ruoat* (food). These categories have a relatively balanced number of articles and cover distinct subjects, which is appropriate for creating a clean dataset for evaluation. However, a single article may have overlapping themes–for instance, an article about elections can have information about employment issues. Naturally, this introduces additional noise in the synthetic datasets and thus a desirable property.

After limiting the data to these 12 categories, a reduced corpus contains 207,881 articles in total. This is then used for generating the synthetic datasets described in the following section.

3.2 Discourse Change Patterns

For the experiments, the corpus is sampled to create datasets which simulate pre-defined patterns of discourse change. Each dataset consists of 100 artificial time points. For each time point, documents from several categories are randomly sampled in such a way that one category follows a non-stable pattern—for example, increases over time—while all

[†] http://urn.fi/urn:nbn:fi:lb-2017070501

others remain stable, i.e. randomly oscillating.

This study proposes six possible patterns of discourse behaviour across time, which are illustrated in Figure 3.2:

- **Up**: The number of articles belonging to a discourse starts increasing at certain time point, and grows until some later point, when it becomes stable.
- **Down**: The number of articles decreases between two time points, then becomes stable.
- **Up - Down**: The number of articles increases, then decreases, then becomes stable.
- **Down - Up**: The number of articles decreases, then increases, then becomes stable.
- **Spike Up**: The trend behaves similar to the Up-Down pattern but spikes are more steep and could appear several times
- **Spike Down**: The trend behaves similar to the previous one but in reversed way.

In addition, the study uses a **Stable** pattern, where there is no significant change in discourse prevalence over time. The precise formulation for the patterns are presented in Section 3.3

For the experiments, 100 time points are used, but this number can be changed if needed. Out of the 12 categories, one is randomly selected as the target category (changing discourse) and then for this category one of the six non-stable patterns is randomly selected.

For the target category, in each time point t, n articles are sampled so that the timeline follows a selected non-stable pattern. While generating these sequences, the pivot points are also randomly assigned when the non-stable pattern starts and ends, which is used as the label for training pivots prediction task later. The remaining 11 categories are sampled following a stable pattern. Thus, the first task for the model could be reformulated as finding documents that belong to a non-stable category among all documents in a given dataset.

An example dataset is presented in Figure 3.1. In this example the Up pattern is used. It can be seen in the figure that random noise is added to all categories, so small spikes are visible for all categories, including stable ones. Note that the input to the trend detection models, described in Section 4.2 are raw texts, while categories are hidden. In this way, we try to emulate a realistic situation where many themes are oscillating in the news at the same time and only a few of them display a certain increasing or decreasing trend.

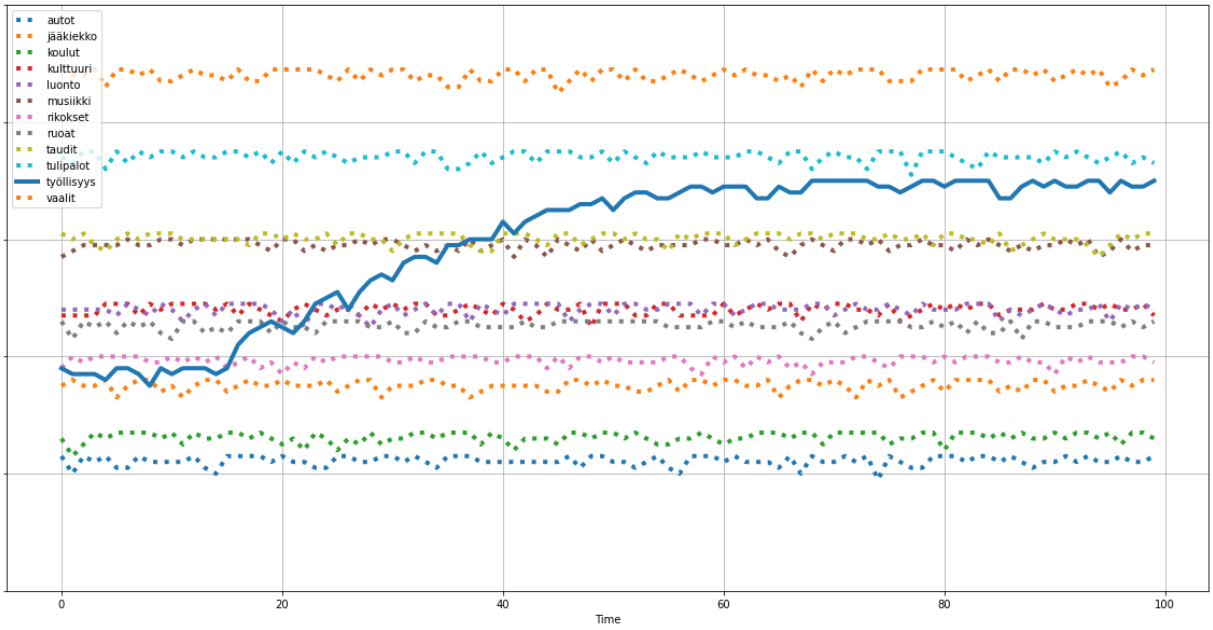

Figure 3.1: A sample experiment with 1 increasing category (Up) and 11 stable categories.

3.3 Pattern Definitions

We now present formal definitions for the patterns. Two functions are used as fundamental components for discourse change: *Sigmoid* or *Gaussian*. Each function with its adjustable parameters can create a typical shape, which will be discussed in more details.

The Sigmoid function is used to sample the **Up** and **Down** patterns. It is assumed that a novel discourse slightly increases or decreases at the beginning, then speeds up in the middle and then gradually slows up before becoming stable again, which is exactly how the Sigmoid function behaves. Thus, the discourse change forms an S-curve, which is a natural shape in many language-change processes [3].

More concretely, a number of articles for each time point in **Up** and **Down** patterns follows this formula:

$$X_i = N + \frac{1}{1 + e^{-k \times (T_i - (T_{end} - T_{start})/2)}} \times N \times R$$

where T_{start} and T_{end} are the time points when the pattern starts and ends, respectively; X_i is the number of articles at time point $T_i \in [T_{start}, T_{end}]$; N is the number of articles before the starting point, R is the change rate for the pattern, and k is the parameter that defines how the change is distributed along the time. With a large k the S-curve is steep,

with a slow change at two ends of the range, and a rapid change in the middle. It was set $k = 0.1$ to form a gradual change from the start to the end. The formula can be used to sample both **Up** and **Down** patterns by reversing the change rate sign. While the **Up** pattern is sampled by using a positive change rate, the negative one is used for the **Down** pattern.

In the same way, the Gaussian function is suitable for the **Up - Down** and **Down - Up** patterns which have a bell shape. By modifying the mean and standard deviation of the Gaussian, it can produce different forms of the bell shape, depending on the amount of data and the number of time points. The bell pattern is sampled using the following formulas:

$$X = Random\_Sample(PDF(\mathcal{N}(\mu, \sigma^2))$$

$$\mu = (T_{end} - T_{start})/2, \sigma = (T_{end} - T_{start})/k$$

$$X_i = N + \frac{X_i - \min(X)}{\max(X) - \min(X)} \times N \times R$$

The X is collection of S samples randomly drawn from the Gaussian probability density function (PDF), where S is the number of points from T_{start} to T_{end}, μ is the mean or middle point in time range, and σ is the variance of distribution. The σ is adjusted by a parameter k in the equation to control the spread of the bell shape. A large k causes a smaller σ, which creates a shape with a narrow spread and sharp peak in the middle and vice versa. From the experiments, it was found that $k = 5$ gives a suitable smooth changing pattern.

After having X sampled in the bell shape, the number of articles is calculated for each time point, however, X needs to be rescaled using min-max scaling as in the last equation. The reason for that because the values of X is a set of probabilities sum up to 1 but we want to have a list of number in range [0, 1] to align with the Sigmoid function output. Similar to the Sigmoid, the **Up - Down** and **Down - Up** patterns are also sampled by reversing the changing rate sign, where the positive value is for the former and negative value is for the later.

Another pattern that uses the Gaussian distribution is **Spike Up** or **Spike Down**. This pattern will have a very short range of beginning and ending time points which is similar to a pine shape. The way to sample these patterns is the same as **Up - Down** and **Down - Up** patterns.

All the non-stable patterns are sampled with the change rates R sampled from uniform distribution in range $[0.3, 0.8]$ to have more variants of shapes.

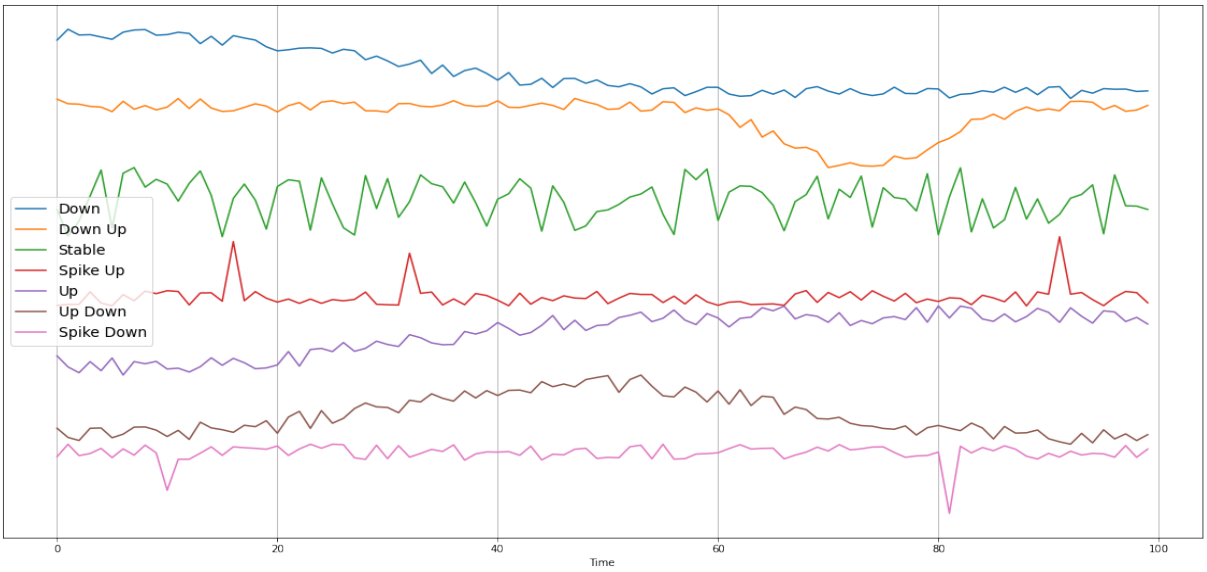

Figure 3.2: Seven patterns used to emulate discourse dynamics in the synthetic datasets.

The stable pattern is a constant plus randomly sampled noise. The same noise is added to all other patterns to simulate a natural distribution of documents in the collection. The experiment designs approximately 100-200 documents from each category at each time point, though obviously this number depends on the pattern.

3.4 Data Sampling

For each synthetic dataset only one category follows a non-stable pattern, while 11 additional categories follow a stable pattern. The role of stable patterns is to add noise for the next training steps, which helps create more generalised models. The way of constructing data points for each pattern is as follows:

For the up, down, up-down and down-up patterns, we randomly assign two points in the timeline as the *pivot points*, denoted by T_{start} and T_{end} in the formulas above. Before and after the pivot points, the data is sampled according to the stable pattern. The data between the two pivots are generated by either the sigmoid or Gaussian functions. Note that each part will receive the number of articles $X_{start-1}$ of the last time point before the pivot point, so it can continue to generate $X_{start}....X_{end}$ for the new time range. Then a stable pattern is formed using X_{end} as a starting value.

Similarly, the spiking (Up or Down) patterns share the same idea of generation, with the difference that there are more than two pivot points in the timeline. For the spiking

patterns, p intervals are randomly generated in the timeline. The time points when the spiking pattern starts and ends are considered as pivots. Using the Gaussian function, we sample the data points around these pivots. Lastly, the stable pattern has no pivots and can be easily sampled by assigning the number of articles to all time points equally.

As will be discussed in the next section, the approach used to detect changing discourse consists of two steps: unsupervised clustering of the documents content for building time-series data, followed by supervised models to predict if a cluster has non-stable pattern and also pivot points. Since only the second step requires training data and each time-series is processed independently of the others, the *training* step uses the synthetic time-series generated according to the formulas described above. In other words, for training it is possible to directly generate a number of documents from a given category at each time step without bothering to sample real documents for this category because they are not used as an input for the model. Because the training data does not require to be clustered in step 1, the artificial noise is added to this data to mimic the noise from the clustering process.

In the experiments, the noise is introduced by adding a small value to each time point. This noise value is a product $n \times r$, where n is uniformly sampled from [0,1] range and r is a noise ratio sampled from 0.0001, 0.001, 0.002 with probabilities 0.1, 0.3, 0.6 respectively. Note that, r is sampled once for each time-series per dataset, and n is for each datapoint. To add even more variation, the pattern change rate R is also sampled uniformly in range from 0.5 to 0.8.

The purpose of this strategy is to provide a large number of samples for the training task and ensure the quality of the labeled pivot points which is hard to control if a clustering step is involved.

Nevertheless, the research purpose is to test the model performance in a realistic setting. Thus, *for testing* it is necessary to generate an actual corpus by sampling from several categories according to a randomly selected pattern, then run this dataset through the clustering step and finally apply the trained model. Thus, data used for testing are noisy. Again, the study deems that to be a desirable property since our goal is not to build a dataset that would be easy to classify. On the contrary, the task needs to be sufficiently complex, allowing us to discriminate between various methods.

4 Methodology

4.1 Method Overview

This thesis proposes different machine learning models to solve the tasks of detecting a changing theme and its pivots. These models can be seen as the references for detangling the problems as well as for bench-marking on the simulated data. The method idea can be summarised as the following. Given a dataset, articles can be grouped into different clusters in which they share the same theme. An important notice that the clustering method only provides approximated themes for articles. Next, a time-series is built by taking the normalized number of documents in a cluster for each time period. Thus, in one dataset, if we have 10 clusters and 100 time ranges, there will be 10 time-series with 100 time steps.

In this study, all of the experiments are gone through two major steps:

- building a time-series from textual data using clustering;
- analysing the time-series to classify them as either stable or unstable and finding pivot points.

The study plans to use supervised deep learning models with sequence-to-sequence mixed classification neural network architecture that gets a time-series as input and outputs whether it has non-stable pattern and sequence of pivots. The neural network has shown its advantages on time-series problems [7], so it can help to detect the patterns even with noisy input.

As already mentioned, while the training process can directly generate time-series input from the sampling method, the validation needs to have clustering step to build this input. For each dataset, the document collection is split into clusters using either K-means or LDA and then a separate time-series is built for each cluster. The clustering process is done separately before calling deep learning models. By doing this, we can also benchmark the performance of clustering methods in addition.

Additionally, the study will set up a baseline based on a regression model to compare the results. Since regression requires an additional step for sequence segmentation, we utilize

the sliding window approach for this purpose. The synthetic datasets are used for both training and validation. Finally, the trained models are applied to real-world data without any modification.

4.2 Building Time-series

In this section, we will discuss about how to build time-series inputs from synthetic datasets. These inputs are necessary for making models to detect changing patterns and pivot points. There are two method used are K-means and LDA which will be discussed now in following sub-sections.

4.2.1 K-means

As mentioned in the Background Theories chapter, K-means is an algorithm for clustering data in groups. By design, K-means is not able to take the text as the features without converting to numeric representation. To compute the distances between articles, K-means clustering requires a dense document representation. This type of representation can be provided by applying Doc2Vec model [16], implemented in the Gensim library *. Before training, the text is tokenized and all stopwords are removed. The setup on Doc2Vec model is kept default with a dimensionality to 128 for vector size, negative sampling of 5 words and train for 30 epochs. The inferred document vectors are then used as the input for K-means clustering. The K-means model is implemented by the Scikit-learn library † with default parameters.

In the experiment, K-means clustering is run independently for each of the 1000 datasets. Thus, each dataset simulates a single independent use case. The number of clusters was set to 20 for all datasets. Thus, the prior knowledge about the number of categories is not used for the training. Moreover, perfect clustering is not possible with this setting since the number of clusters is bigger than the number of categories used to generate a dataset. The rationale behind this is that when working with real data we would not know the number of discourses in the collection. The proposed method does not aim at perfect clustering, only on detection of non-stable trends. The number of non-stable trends found by the model does not matter; it does not affect the measures of system performance,

*https://radimrehurek.com/gensim/models/doc2vec.html
†https://scikit-learn.org/stable/

which will be described in Section 5.

Clustering is done jointly for all time points in the dataset. The timeline for each cluster is built by counting the number of documents from each cluster at each time point. Timelines are scaled to $[0, 1]$ intervals so that the biggest value for each timeline is always 1. This is needed to standardise the input for the next step of training models.

4.2.2 LDA

Topic modelling can be used as an alternative to clustering. The implementation uses the Gensim library with built in LDA * model with asymmetric priors learned from the data. Similar to K-means, one topic model is trained for each synthetic dataset, thus we have 1000 models in total. Topic model training was done in parallel †. Unlike K-means, the LDA model receives text directly as the input. The number of topics is set to 20 to align with K-means.

The timeline on top of LDA is built using soft grouping instead of hard grouping, since an article can have more than one topic. The LDA output is usually unbalanced with a few large topics containing most of the articles, and many much smaller ones. To count the number of documents that belong to a certain topic, a threshold is applied to accept the documents where the topic probability is higher than 0.25. If no topic has a probability above the threshold, the document is assigned to the topic with the highest probability. Thus each document contributes to at least one topic timeline. Similar to K-means, topic timelines are scaled to $[0, 1]$ range.

4.3 Baseline

4.3.1 Linear Regression

The baseline approach is based on linear regression. Dissimilar to the neural model, it is not independent for each cluster within the dataset. The idea is to compare the slopes of all clusters to determine which one has a significant difference.

A linear regression model is fitted to each of the 20 clusters obtained for the dataset. The absolute value of slope from linear function is normalized to a $[0, 1]$ scale, so that the

*https://radimrehurek.com/gensim/models/ldamodel.html

†It was done by my colleague Elaine Zosa (elaine.zosa@helsinki.fi).

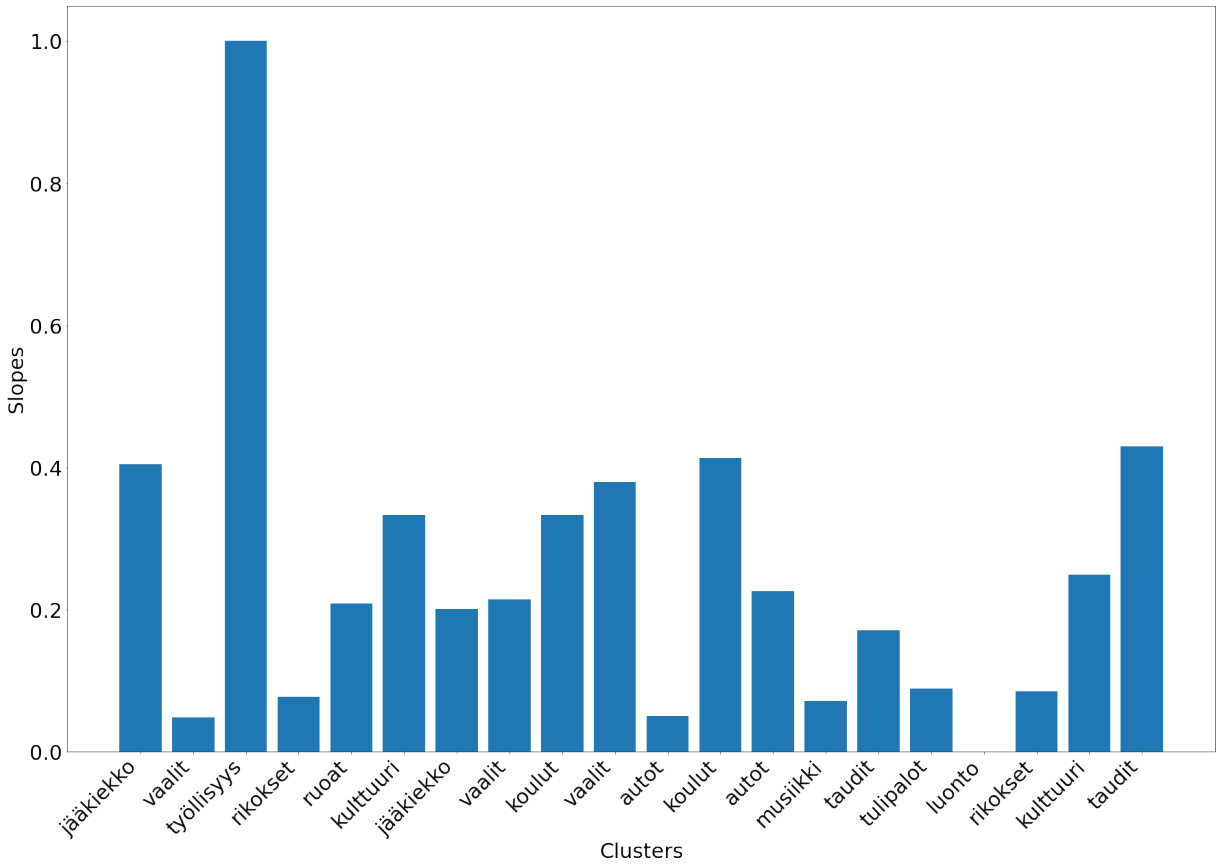

Figure 4.1: An example dataset, where each cluster, obtained from K-means, is fitted with linear regressions. The normalized slopes are shown in the histogram, with one pattern having significantly higher slope than the others, which indicates non-stable discourse dynamic. Bars are labelled with the major category of the articles within the cluster.

largest normalized slope is equal to 1. A time-series with a slope above a certain threshold is then classified as non-stable. Depending on the threshold and slope value, there could be more than one non-stable clusters. After preliminary experiments we set this threshold to 0.8 for all datasets. This also means there always will be at least one non-stable cluster detected as the max slope value is 1. However, it has no warranty the detected cluster is the correct one.

For instance, the dataset presented in Figure 4.1, after clustering and fitting to linear models, has the output slopes shown in Figure 3.1. In the histogram each bar is a cluster labeled with its major category, i.e. the most frequent category for the clustered articles. The y-axis is the normalized slope value. We can see that the category for the biggest bar—*työllisyys*, employment—is the same as one used to build the increasing pattern in Figure 3.1.

4.3.2 Sliding Window Segmentation

Time-series identified as non-stable in the previous step are processed using the sliding-window segmentation method to identify pivot points. Rupture library [35] provides an implementation which can be used directly.

The algorithm uses two windows, which slide along the timeline. These windows are used to measure the discrepancy between the left and right context at a given time point. The idea is that if both sliding windows fall into the same segment, the discrepancy will be lower. If the discrepancy is significantly higher, it can be considered as a pivot point. 4.2

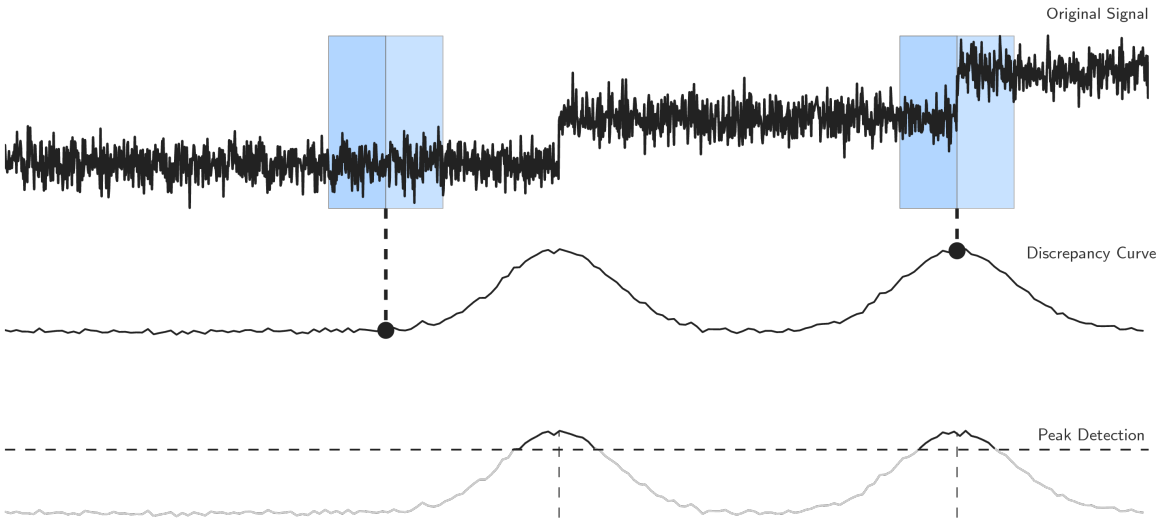

Figure 4.2: Schematic view of the window sliding segmentation algorithm. The window on the left (blue) and one on the right (light blue) are sledded over the signal. The discrepancy formed peaks on the pivots. Image source: [35]

The formula for calculating the discrepancy is taken from the Rupture documentation*:

$$d(y_{u..v}, y_{v..w}) = c(y_{u..w}) - c(y_{u..v}) - c(y_{v..w})$$

where c is the cost function to calculate the differences between two values, for example L1 or L2, y_u, y_v, y_w are the input value at timepoints of sliding windows $u..v$ and $v..w$. Following the documentation of Rupture, we set the window size 5 units, $jump = 5$, use L1 as cost function and a penalty of 0.5 to prevent overfitting. The algorithm takes the

*https://centre-borelli.github.io/ruptures-docs/user-guide/detection/window/

normalized time-series as the input. The Rupture function then gives the output as a list of points which separate the segments, or known as the pivot points.

4.4 Detecting Non-stable Patterns And Pivots

As stated in the Motivation section, the study proposes a novel neural network-based architecture that is trained jointly to solve two tasks: (1) to detect whether a time-series has a non-stable pattern; and (2) to detect the pivot points in the non-stable time-series. Someone might argue that we can look at the visualization of time-series data to detect if it has a non-stable pattern or not; also the pivot points can be easily spotted by human eyes. However, imagine if we have to extract the interesting documents belonging to certain periods of an unknown theme in a very large corpus, the task becomes not easy, as we may have to analyse thousands of visualizations. On the other hand, not all visualization is easy to detect the changing period, due to the noise in the time-series data. Applying machine learning methods for automating these tasks brings many benefits. Beside, there is assumption that machine learning models are trained on various simulated patterns with noises could generalise well to new datasets.

The model is a combination of Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). In addition to the combined model, the study also conducted experiments on CNN and RNN separately. The following sections will first present each model individually and then describe how they are combined.

4.4.1 Data Format

The cluster-based time-series, described in the previous section, are used only to construct the validation set. As has been mentioned before, real articles are not required for the training process in supervised methods. Instead, we can directly sample the patterns with noise to mimic the sequence of frequency in the clustered set.

There is no guarantee that the article distribution obtained by clustering still maintains the shape of the pattern used to produce the dataset. It depends on the quality of the clustering method. When noise is introduced directly into the generated timeline, the shape of the pattern is not modified. Thus, the data used to train a neural network is cleaner than those used for validation. This way, we are able to generate more samples for training and avoid unexpected behavior from the unsupervised process.

A training set of 100K artificial timelines is generated with artificial noise introduced by policy in section Data Sampling.

The input for our models is a sequence of frequencies in a time-series. There are 100 timepoints, each has the frequency normalized to [0,1] range.

The model produces two outputs: a binary prediction of whether a time-series is stable or non-stable and a sequence of the same range, where the value at each time point is the probability that the time point belongs to a non-stable pattern. To generate this sequence in the training data, we set all values to 1 between the pattern start and end period, while all other values are set to 0. If the time-series is stable, all values in the output sequence are zeros, which corresponds to zero value for the first output.

Stable and non-stable time-series are sampled equally for the training. Out of 100k samples in the training set, 5k are used as a development set and the rest as a training set.

4.4.2 RNN model

As has been explained in the previous section, the data is in time-series format, which contains the articles frequency in each time point. As already discussed in the Background Theories, the RNN and its variations are designed for the temporal data by memorizing the important information for each time step [30]. In this section, we apply RNN model to predict whether a cluster is a changing theme and to detect the pivot points of the changing periods. The input to this model is a matrix with the shape $(N, 100)$ where N is the batch size and 100 is the length of the time-series. Each example is a sequence of numbers in the range $[0, 1]$.

The model structure is presented in Figure 4.3. Instead of using the original RNN, the study applied an RNN variant—bidirectional Long Short Term Memory (bi-LSTM) to solve the problem. Bi-LSTM is capable of learning long-term dependencies which is useful for handling long sequence. Furthermore, it is able to capture the features from both directions of the sequence. In the experiment, the bi-LSTM layer uses 256 hidden units. The output of the bi-LSTM is a sequence of outputs and a hidden vector containing sequence information. The output sequence is then flattened and passed through a fully connected (FC) layer. Between bi-lSTM layer and FC layer, a dropout layer is introduced to reduce the overfitting.

The FC layer is connected to two output layers: one to predict the probability that the input is non-stable and the other to predict a sequence of pivot probabilities. Both output

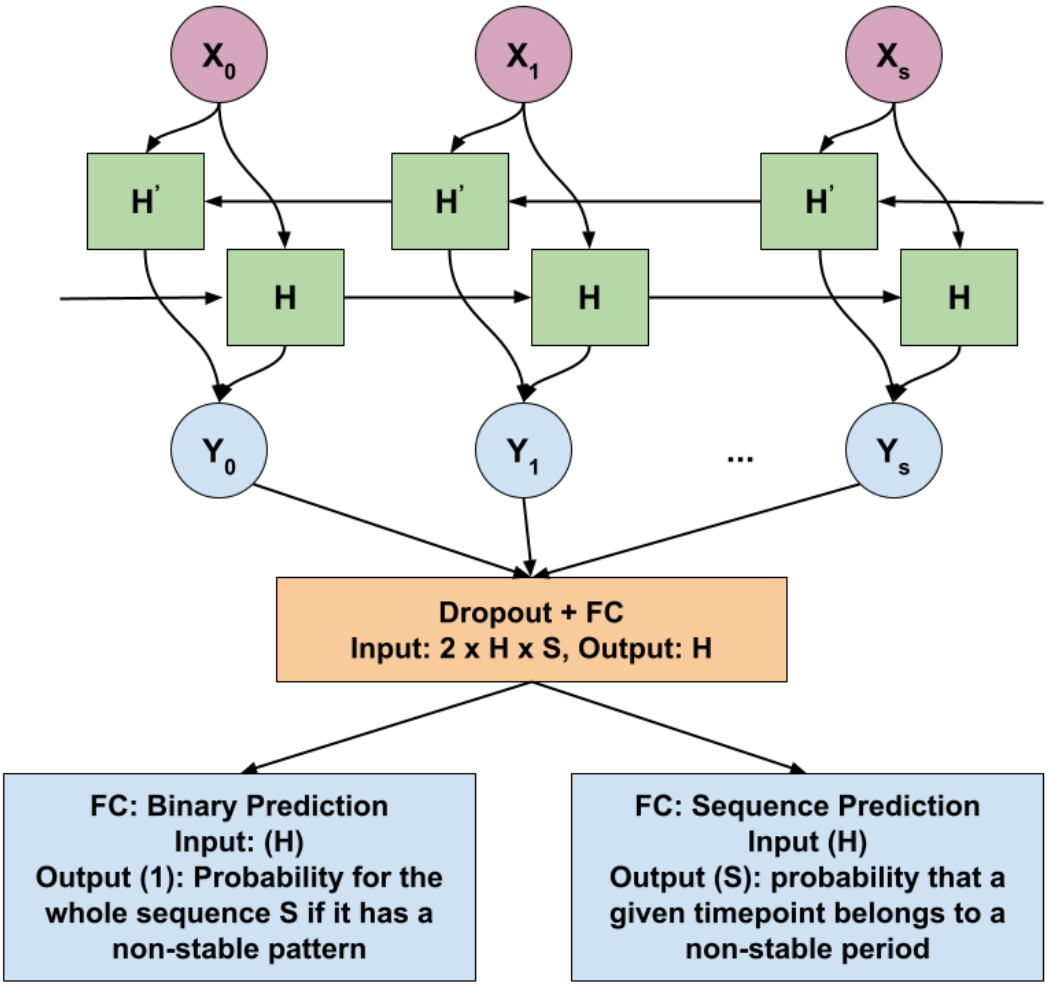

Figure 4.3: RNN network architecture with biLSTM layer. The prediction Y from all time steps are used for the FC layer.

layers use the Sigmoid activation function to get probability values. Because we have two different output layers, the losses are also calculated separately for each. The first output layer uses Binary Cross Entropy (BCE) as the cost function. On the other hand, BCE loss was calculated for each probability value of sequence in the second output layer and then averaged for the whole sequence. The losses from two output layers are then summarized for a backpropagation process. The model was trained with learning rate $1.e^{-4}$, Adam optimizer and converged after 30 epochs.

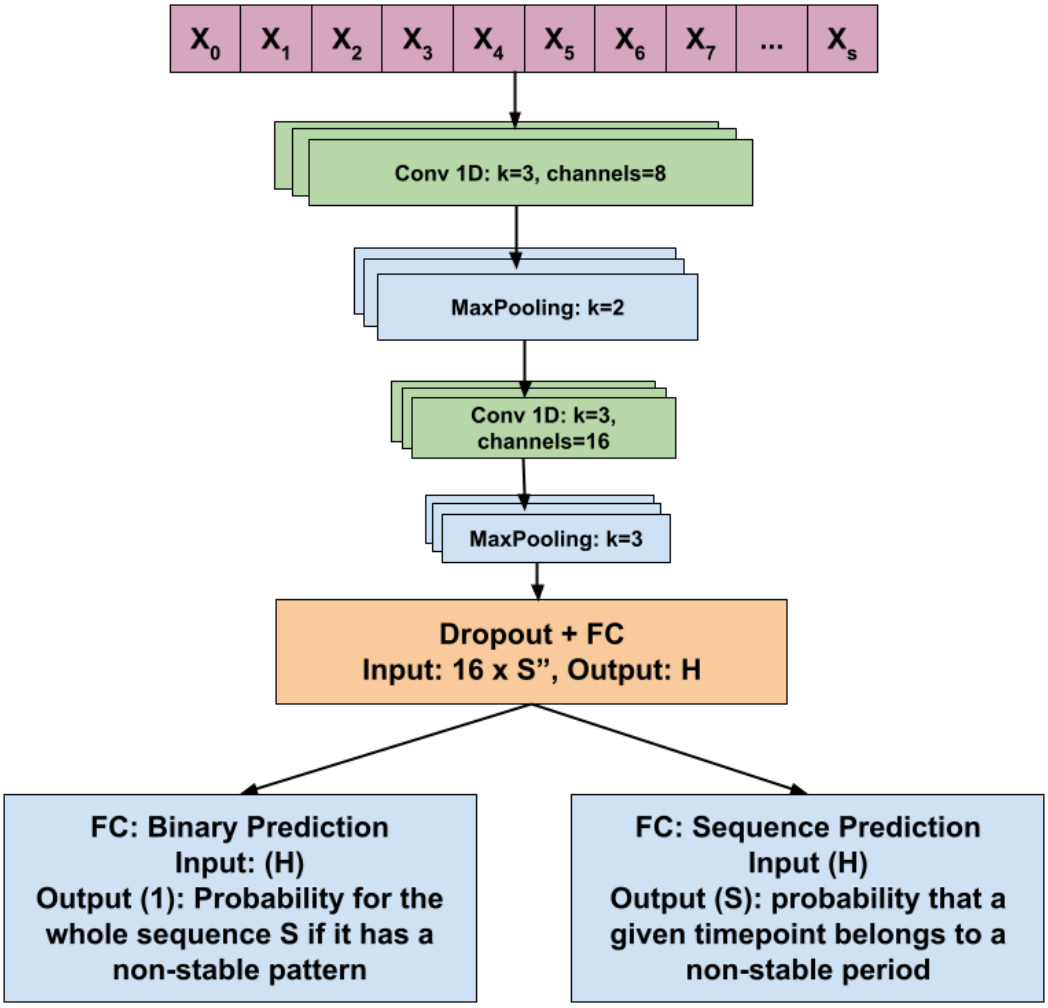

Figure 4.4: CNN network architecture. Where k is the kernel size, H is the hidden size, and S" is the length of sequence after going through the convolutional layers.

4.4.3 CNN model

Instead of using RNN, this section will describe how to apply convolutional neural network model to time-series data. As mentioned in Background Theories, the CNN is intended for capturing local features for image recognition [30]. The time-series data, in a different point of view, can be seen as an image representing the discourse dynamic. The idea is to use CNN ability to detect patterns in the sequence data. The CNN model is shown in Figure 4.4.

The input and output is the same as one described for the RNN model. Because the formatted sequence data is the one-dimensional vector, the 1D CNN layers are suitable for feature extraction in this case. In the setup, there are two convolutional layers stacked

with max pooling layers. Each layer has a kernel size of 3, this means the filter slides over the timeline to cover 3 time points at a certain time step. The extracted features are then flattened and passed to a FC layer. Next, the output from the FC layer is again used for two output layers. Similar to RNN model, one output layer is to predict whether the sequence is a non-stable pattern and the other returns the probability that a given time point belongs to a non-stable period.

The first layer of CNN has 8 output channels while the second one expands to 16 channels. The 1D Max Pooling layer is set the kernel size to 2 after each convolutional layer to extract the most significant information. As the rest is inherited from RNN model, the first output layer has the dimension (N, 1) where N is the batch size and 1 is probability value, the second output layer the dimension (N, 100), where 100 is the number of probabilities for 100 time steps. Finally, the model was trained with a learning rate $1.e^{-4}$, Adam optimizer and converged after 30 epochs.

4.4.4 Combined Model

While RNN is good at handling temporal information, CNN has the strength at local pattern detection. However, in a local region, if a non-stable shape is spawned accidentally due to the noise, the CNN model might mix it with a valid pattern. RNN can handle longer sequences due to its ability to "memorize" the sequence state. The strengths of both models are leveraged to produce a combined model that might be more robust at pivot point detection.

The architecture of the combined model (which is further denoted as RCNN) is presented in Figure 4.5.CNN and bi-LSTM layers are identical to those used in the separate models. Then the hidden state output of the bi-LSTM layer is concatenated with the output of the last convolutional layer, flattened, and passed to the FC layer.

For each time step in the input, the Bi-LSTM layer will output a prediction and update the hidden state. Note that when the RNN model is used alone, we use all predictions from Bi-LSTM layer. However, the combined RCNN model only takes the hidden state for the next step and all the time step predictions are discarded. After concatenating RNN and CNN outputs, the rest of the model is organized identical to the previous cases.

In all three neural network models the dropout probability is set to 0.5. The combined model is also trained with the batch size of 64 and converged after around 30 epochs. Again, Adam optimizer with learning rate $1 \cdot e^{-4}$ is used. Similarly, the loss is calculated

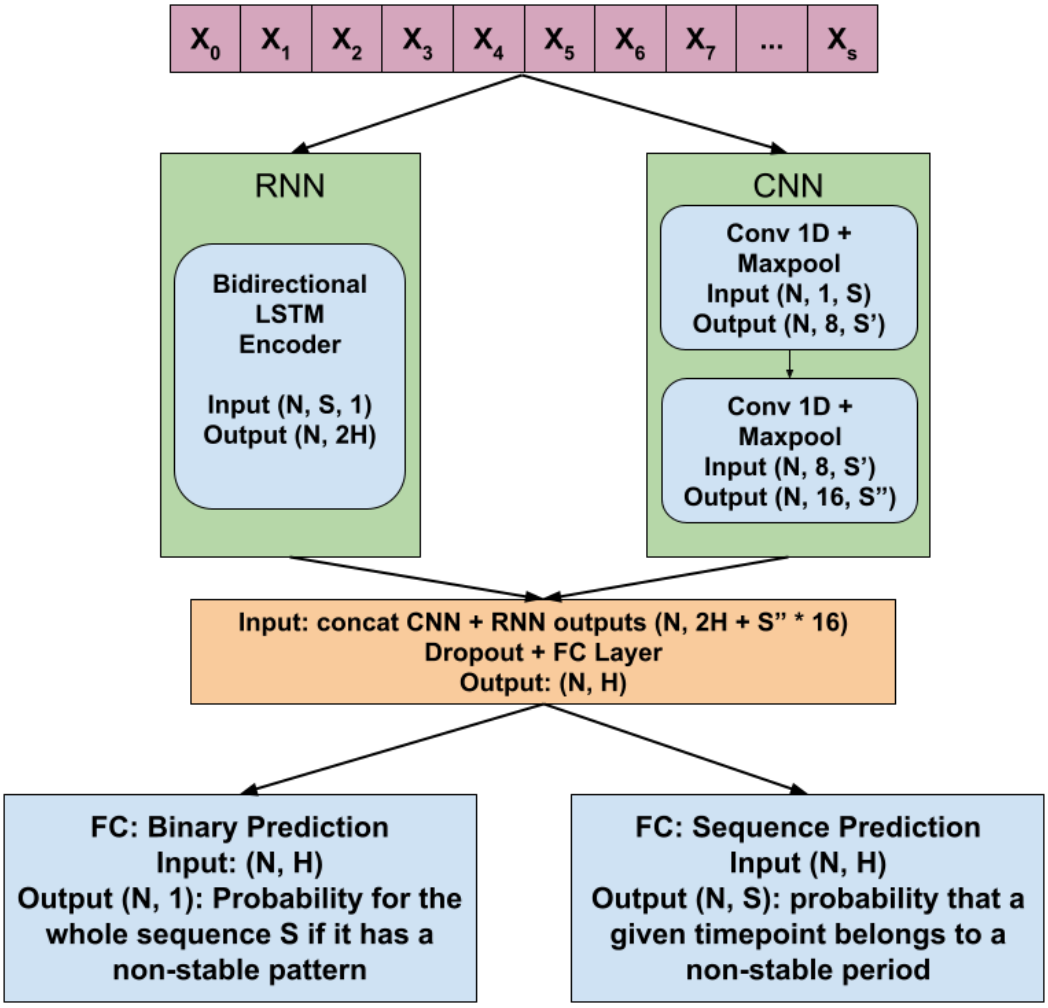

Figure 4.5: Combined (RCNN) network architecture. Where N is the batch size, H is the hidden size. S is the length of input sequence, S" is the length of convolution output. The hidden states from LSTM are used for the next layer instead of the predicted outputs.

by summarising two binary cross entropy (BCE) loss functions, one for binary classification and other for sequence of pivots prediction.

5 Evaluation

In this chapter, the study proposes several evaluation metrics and shows the result on the testing synthetic datasets.

The system performance is evaluated at different levels:

- *Dataset level* – calculate a percentage of datasets, where a model found at least one non-stable cluster in true non-stable clusters;
- *Category level* – use accuracy to measure how often the major category within a detected non-stable cluster is the true non-stable category;
- *Document level*, measure the proportion of true category documents within clusters detected as non-stable. For this we use recall, precision, and F-measure;
- *Time-point level*, apply the RandIndex metric to identify how close the predicted pivot points are to the real pivot points.

Since evaluation is done on synthetic datasets the ground truth for all these measures is directly available by the generation process. The following sections will discuss each of these measures in more detail.

5.1 Dataset-Level Metric

Even though all synthetic datasets contain exactly one non-stable category, trained models do not make any assumption on the number of non-stable patterns. Thus the methods will generalize to real-world use cases where the number of changing discourses is not known in advance. As a consequence, an output can contain an arbitrary number of non-stable patterns, or none. As a first rough estimation of the model performance, this metric is computed as a percentage of datasets, where a model predicted at least one non-stable cluster correctly, which means a cluster has a major category same as ground truth category of changing pattern.

5.2 Category-Level Metric

Category-level accuracy measures how well a model can detect a non-stable category. For each cluster classified as non-stable, a *major category* is defined as a category that has a highest count in this cluster. If this major category is the same as the target category used for the dataset generation, then prediction is considered to be correct. For each dataset, the study calculated a ration of correct non-stable clusters to all non-stable clusters. If a model does not find any non-stable cluster for the dataset the accuracy is set to 0. Thus, the model is punished for not finding any changing trends but also punished for finding too many of them. However, it is not affected if a non-stable category is split into two clusters or if a non-stable cluster contains many documents from other categories.

5.3 Document-Level Metric

Precision, recall and F-measure are used to measure how "clean" are subsets of documents that form non-stable patterns. For this evaluation, we use all clusters that are predicted to be non-stable, even if their major category is incorrect.

For each non-stable cluster, precision is calculated as a proportion of documents from the target category in this clusters:

$$precision = \frac{C \cap N}{N}$$

, where C is a set of articles in the target category and N is a set of articles in the non-stable cluster.

The dataset precision is the mean of all non-stable cluster precisions. A model is penalized for splitting a target category in two clusters even if each of them does not contain any noise. On the other hand, precision for a cluster could be non-zero even if its major category is incorrect.

Recall is the proportion of documents from the non-stable cluster in the target category:

$$recall = \frac{C \cap N}{C}$$

Similar to precision, recall for all non-stable clusters is averaged, and a split of the target category leads to decrease of this measure.

F-measure is computed as the harmonic mean of recall and precision. If all clusters are predicted to be stable then precision, recall and F-measure are set to zero. Then all three measures are averaged across datasets.

Note that evaluation is focused on non-stable clusters and a target category. The distribution of all other categories among clusters does not affect any of the measures. The reason for that is that our task is to extract dynamic trends from the data, rather than describe a collection as a whole.

5.4 Timepoint-Level Metric

For each cluster that is classified as non-stable, a model also must output pivot points, i.e. time points where the non-stable pattern starts and ends. These pivot points segment a timeline into several periods. Then each pair of time points could belong either to the same or to different time periods.

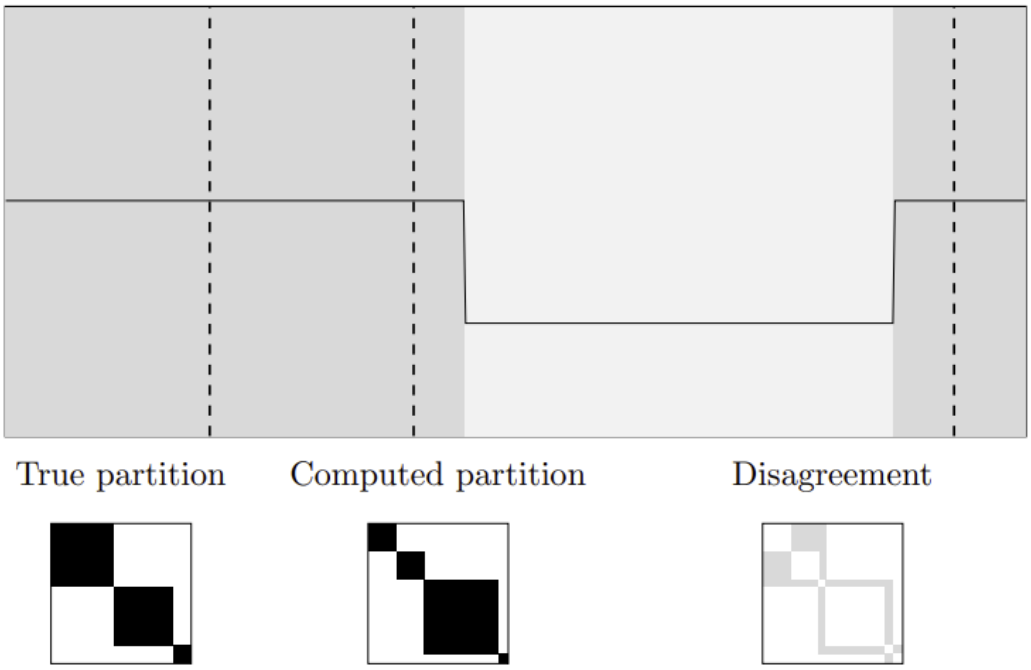

Figure 5.1: RandIndex. Top: gray areas are the ground truth segmentation A, the dashed line mark the predicted segmentation $\hat{A}$. Below: representations of associated adjacency matrices and disagreement matrix between true segmentation and predicted one. The RandIndex is demonstrated as white area in the disagreement matrix. Source: [35]

RandIndex is computed as a proportion of time-point pairs correctly put either in the same

or in the different periods. Since each timeline consists of 100 time points, shifting a pivot point by 1-2 positions from the true point slightly decreases RandIndex. However, radical misplacement or finding an incorrect number of pivot points results in a large performance drop. The formula below is used to calculate the RandIndex:

$$RandIndex = \frac{\sum_{i<j} \mathbb{1}(A_{ij} = \hat{A}_{ij})}{T(T-1)/2}$$

This formula is from the Rupture package documentation * and also is mentioned in Truong, Oudre, and Vayatis [35] works. The intuitive explanation for the formula is illustrated in figure 5.1, where the A denotes a matrix of all pair indexes for ground truth segmentation, $\hat{A}$ is a matrix of segmentation predicted by a model. In a segmentation, for example A_{ij}, if a pair of indexes A_{ij} are in the same segment, the value of A_{ij} is marked as 1, otherwise 0. After that, the matrix values are compared for two segmentations A and $\hat{A}$ in each pair index. If this value is equal for both segmentations, it is counted for the RandIndex. T is the length of the sequence or the number of indexes in the matrix which is used for normalizing RandIndex values. The RandIndex returns a value between 0 and 1, where 0 indicates that the two segmentations have no match on any pair of points and 1 indicates that the two segmentations are identical.

RandIndex is averaged for all non-stable clusters in the dataset. If all clusters are classified as stable, RandIndex is zero. This measure is then averaged across all datasets.

Note that this evaluation is orthogonal to the document-level measures, since it is possible to place pivot points to correct positions even if a cluster is noisy or incomplete.

5.5 Results Analysis

Table 5.1 shows results obtained on the synthetic datasets with aforementioned measures. One of the most important results for us is the diversity of the model performance: this means that synthetic datasets are adequately complex and allow for method comparison.

The best performing model is the proposed combination of RNN and CNN (RCNN), which gives the highest results in combination with both K-means and LDA. The only exception is the dataset coverage metrics, which is highest for the CNN model, though the difference is not significant. The best performance is obtained by applying the combined model

*https://centre-borelli.github.io/ruptures-docs/user-guide/metrics/randindex/

on top of the K-means output. On top of LDA the combination also yields the highest performance.

Comparing K-means and LDA, K-means works better for most of the models and measurements. In both cases, we can see CNN is better than RNN at non-stable pattern detection. However, RNN yields a much higher RandIndex, which means better at pivot point detection.

Method		**Dataset**	**Category**	**Precision**	**Recall**	**F1**	**Rand**
STEP 1	STEP2	**coverage**	**accuracy**				**index**
K-means	Regression	90.55	52.78	43.98	34.73	37.04	42.52
	RNN	95.33	73.63	60.55	46.33	50.43	73.17
	CNN	**96.59**	75.17	61.46	46.56	51.49	67.79
	RCNN	95.10	**78.43**	**63.77**	**51.69**	**55.22**	**73.26**
LDA	Regression	88.81	41.88	31.56	31.26	27.14	41.04
	RNN	89.51	38.65	30.48	31.84	27.53	65.04
	CNN	92.07	47.73	36.41	33.26	31.87	53.27
	RCNN	90.05	51.46	37.22	43.94	36.03	60.43

Table 5.1: Result obtained on 1000 synthetic datasets

The difference between LDA and K-means would need deeper investigation in the future. The models applied on top of LDA yield low document-level F-measure, and especially low precision. The RandIndex is also lower than for the K-means results though higher than could be expected judging from the document-level performance. For example, the LDA+RNN model yields a RandIndex close to the K-means+CNN one, even though for LDA+RNN F-measure is much lower. This confirms our assumption that RandIndex is independent of the document-level performance.

Obviously, LDA is much more than just a clustering technique: LDA is a Bayesian model, which outputs topic distribution over documents. In our experiments, this distribution is converted into hard labels and used indirectly. It is likely that a higher performance could be achieved by other ways of combining topic modelling with neural networks. There is another difficulty related to morphologically rich languages like Finnish, where words have many variants and compounds are frequently used [6]. This makes LDA hard to handle the semantic relationship, even with the lemmatized texts. Thus, the quality of clustering result from LDA might be affected.

6 Real-world Experiment

For a qualitative assessment, this chapter demonstrates how the method is applied for real-world datasets. There are two different datasets used for this experiment. The first one is a Finnish corpus: The Finnish News Agency (STT) Archive, which is freely available for research use via Kielipankki[†]. The second dataset is The State of the Union (SOTU) dataset[‡]. The datasets details are explained later in following sections. The study used the best model RCNN which was trained on synthetic datasets to apply for these datasets without any modification.

6.1 STT Dataset

Figure 6.1: A non-stable cluster obtained on the STT data. Automatically detected pivot points show with red vertical lines. The documents within this cluster are about sport and competitions. The green rectangles show dates of the Hockey World Championship (left) and the Olympic games (right).

The STT corpus consists of the Finnish news articles for the period between 1992-2018.

[†] http://urn.fi/urn:nbn:fi:lb-2019041501

[‡] https://www.gutenberg.org/files/5050/5050.txt

In this experiment, only the data from years 2007-2008 is taken, which does not overlap in time with the YLE dataset. The data consist of approximately 250,000 documents.

For the experiments, the two-year data is splitted into weeks, excluding the first and the last two weeks, which gives us 100 weeks. Thus the timeline has the same length as the synthetic datasets and we could directly apply models trained on synthetic data.

Before applying the RCNN model, K-means clustering was trained with 20 clusters for this dataset. Out of those 20 clusters 6 were classified as unstable. After having the time-series from the clustering process, the trained RCNN model directly used to predict the cluster change and pivots. The study briefly scanned the documents within these clusters and found a couple for which could be found interpretation.

Figure 6.1 shows a cluster, which contains articles about sport competitions. The periods of non-stability—between red vertical line in the plot—roughly correspond to two major sport events: the 2007 Hockey World Championship* and the 2008 Olympic games. There is also another sport-related cluster spotted, where a period of instability roughly coincides with the Olympic games.

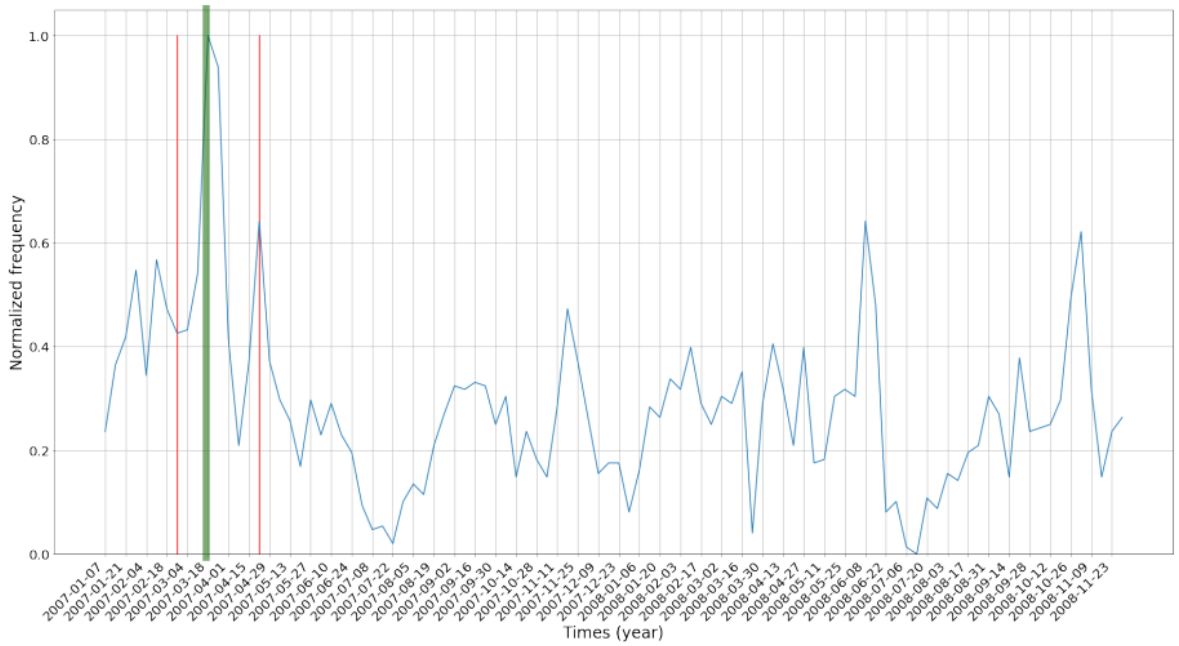

Figure 6.2: A non-stable cluster obtained on the STT data. Automatically detected pivot points show with red vertical lines. The documents within this cluster are mostly about politics and parties. The date of the Finnish Parliamentary elections is shown with green vertical line.

Another cluster, shown in Figure 6.2 is associated with party politics. The date of the

*https://en.wikipedia.org/wiki/2007_IIHF_World_Championship

Finish parliamentary elections is shown with the green vertical line. This date is positioned between two automatically determined pivot points—it seems natural that elections are actively discussed in the news some time before and after the event.

In the experiments with the STT data, one cluster contains hundreds of documents lying between pivot points which takes attention. The clusters themselves are quite noisy, which could be expected from relatively low document-level F-measure for synthetic data. Thus, finding interpretations for non-stable behaviour is a tedious task. However, the study could combine the proposed method with other automatic description techniques, such as finding the most prominent keywords for each cluster or generating a summary.

6.2 SOTU Dataset

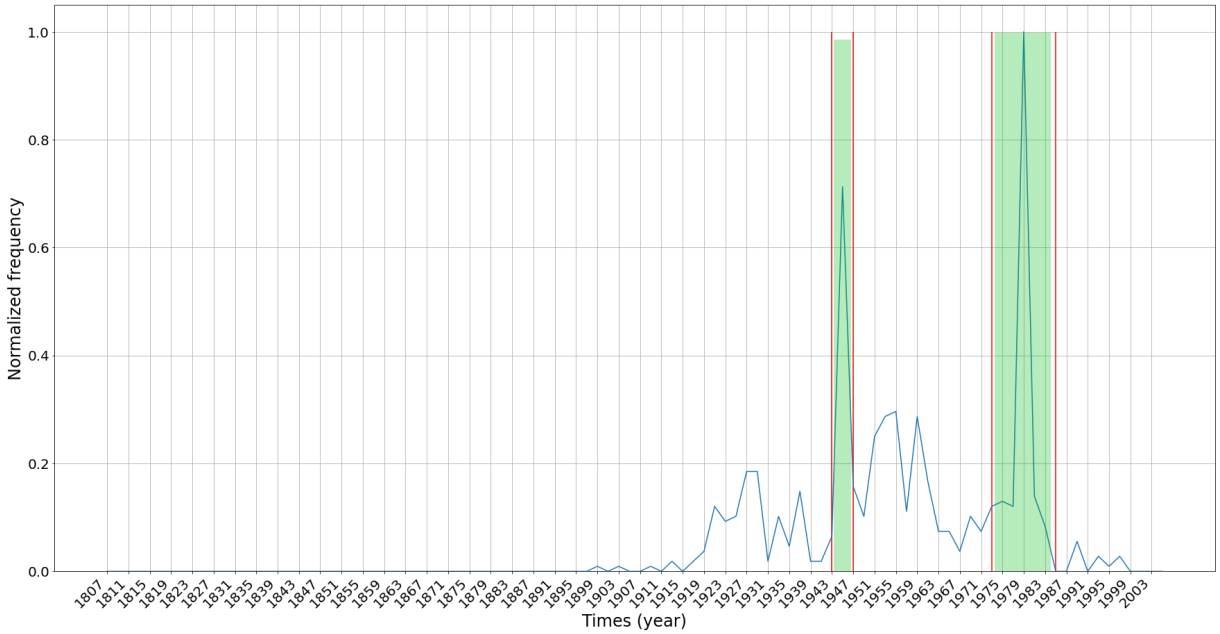

Figure 6.3: A non-stable cluster obtained on the SOTU data. Automatically detected pivot points show with red vertical lines. The first green segment contains the documents about post WWII plan. The second green segment contains the documents related to the energy crisis event.

The SOTU corpus is an annual message presented by the President to Congress, describing the state of the country and his plan for the future. This dataset contains 6564 documents from 1790 to 2006 (from George Washington to George W. Bush). These documents are part of the speeches, which were separated into paragraphs so that each paragraph is processed as a document. Because many topics are mentioned in a speech, by dividing them into paragraphs, we can reduce overlapping themes in one document, which is helpful

for the clustering process. This splitting procedure is also used in X. Wang and McCallum [38] works to have more documents and improve the robustness of the discovered topics.

As the trained model RCNN takes 100 time steps as an input, only documents from 1807 to 2006 are considered for further analysis. With this time range, it is possible to create a timeline of 100 years as the data points. After pruning, there are 6406 documents available for further process. The text then again goes through a cleaning step with punctuations and stopwords removed before training a Doc2Vec model. After that, the output vectors from Doc2Vec model are used for K-means clustering with 20 clusters. Similar to STT dataset, the study also applied the best model RCNN for SOTU dataset. Notice that, the language of SOTU is English which is different from STT dataset, however, both datasets are applied the same model RCNN which was trained by synthetic datasets before.

In the 20 clusters, RCNN model classified 12 ones as having a changing pattern. As already stated, the real-world datasets are manually analysed because an annotation not only doesn't exist but it would be extremely different to label that amount of data. After scanning through the result, we picked up two clusters which contain interesting information.

The figure 6.3 demonstrates the time-series of the first cluster. Two unstable segments were detected in this cluster. In the first segment, which is shown in the green area from 1945 to 1949, there are documents about weapons, defense industry, wage and job. Based on the time and the text in those documents, we can guess those related to World World II (WWII) and plan for employment after the war.

The second segment which is highlighted from 1974 to 1985 contains many keywords related to energy and oil. We assume this is related to the energy crisis event in the 1970s [*]. The detected pivot points from the model are very close to the time range of this event, which started from 1973 until the mid 1980s. De Groot [4] study showed that the energy crisis event was strongly related to the Cold War between Soviet and United States, which happened after WWII. The study's assumption is that the energy crisis event significantly affected the US economy and employment.

Turning now to the second cluster which is shown in figure 6.4, where one non-stable pattern was detected. In this cluster, the keywords about the Vietnam war and Soviet have been mentioned frequently. The documents between two red lines are from 1974 to 1981. Compared with the historical timing, the U.S. forces withdrew in 1973 and the

[*]https://www.history.com/topics/1970s/energy-crisis

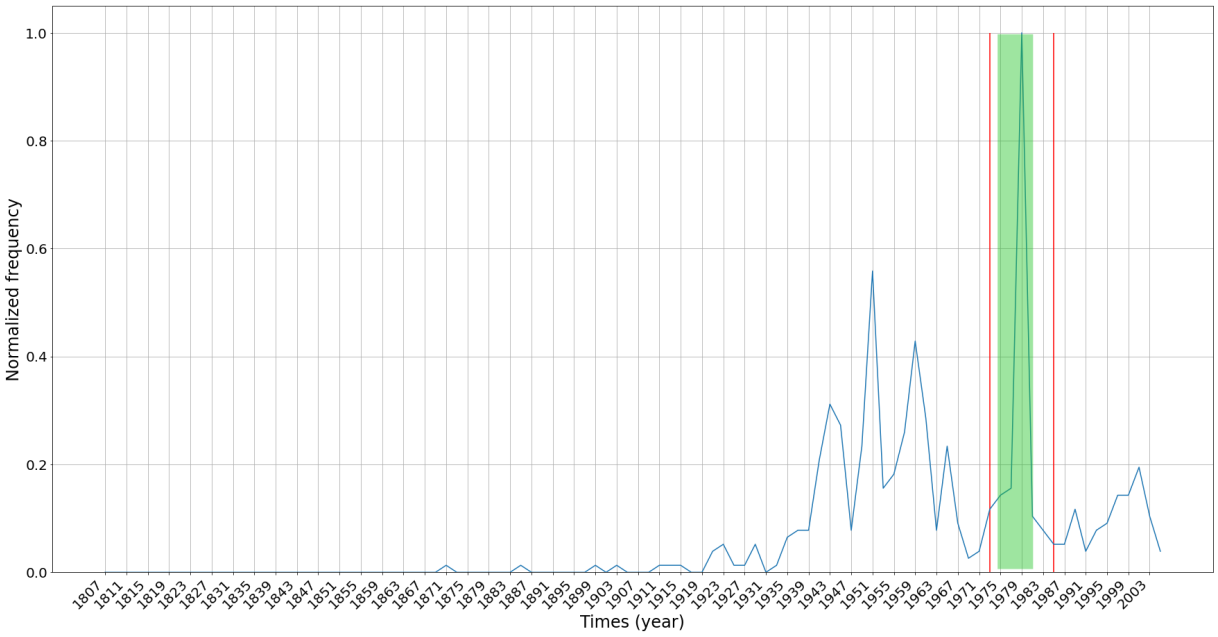

Figure 6.4: A non-stable cluster obtained on the SOTU data. Automatically detected pivot points show with red vertical lines. The green area contains the documents about post Vietnam war.

Vietnam war ended in 1975. This study assumes the speeches are related to aftermath of the Vietnam war* with the conflicts between United State and Soviet Union time†.

This study uses this experiment to demonstrate that a model trained on synthetic datasets, which are generated using the proposed procedure, is able to extract meaningful results from real-world data. Whether these results would be relevant for digital humanities or computational social science research is yet to be found. Most probably they would be interested in less obvious cases than sport events. It is not unlikely that cases difficult for us to interpret would be the most interesting for the experts.

Since training is done using synthetic data nothing prevents us from using more or less than a hundred datapoints. The number of clusters can also vary since each cluster is processed independently. This allows us to use different levels of granularity in discourse analysis, which would impact applicability and interpretability of results.

*https://www.archives.gov/research/vietnam-war

†https://www.trumanlibrary.gov/public/TrumanCIA_Timeline.pdf

7 Conclusions

This study presented the novel task of automatic detection of discourse change in large text collections. This task is relevant for many research questions in digital humanities and computational social science and could potentially have applications in media monitoring. However, computational methods to tackle this type of problems are not yet established.

One of the main obstacles is the lack of training data and fundamental difficulty to annotate corpus-level phenomena. To overcome this issue the study proposed a methodological framework that leans to discourse change simulation with synthetic data.

Synthetic data allows us to train supervised models. Moreover, the procedure which was proposed in this study generates sufficiently complex datasets so that the problem cannot be solved by simple means, such as regression. This allows for evaluation, comparison, and improvement of the methods, impossible on most typical use cases where ground truth is not accessible.

The research proposed a combination of clustering with a neural sequence-to-sequence model to extract non-stable trends and find periods of instability in the data. The best-performing method yields 78% accuracy in non-stable trend detection and 73% RandIndex for pivot time points detection. Nevertheless, the complexity of the task leaves much space for improvement even on synthetic data.

Finally, the thesis demonstrated that a model trained on synthetic data is able to find change in real news content, without fine-tuning or any other adjustments for the data. This is a valuable property, since digital humanities use cases often involve a single unique dataset, which makes it difficult to optimise models or tune hyper-parameters.

The study has many potential directions for future work. While the study demonstrated some example use cases, there could be collaboration with humanities and social science specialists to test applicability of the proposed method in practice. Moreover, it also leaves room for further improvement of the models using synthetic datasets. This does not need any manual evaluation and to a large extent could be done independently from domain specialists. However, the model for finding discourse change and pivot can be enhanced to gain more accuracy as well as robustness on different use cases. For example, instead of building time-series data by K-means or LDA, another clustering method can be applied.

One obvious—though not trivial—next step could be using texts as an input, in addition to one-dimensional time-series utilized in the current implementation. The study will also investigate how to utilize full LDA output rather than hard topic assignments.

Bibliography

[1] D. M. Blei and J. D. Lafferty. "Dynamic topic models". In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 113–120.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent dirichlet allocation". In: *J. Mach. Learn. Res.* 3 (2003), pp. 993–1022. ISSN: 1532-4435. DOI: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL: http://portal.acm.org/citation.cfm?id=944937.

[3] R. A. Blythe and W. Croft. "S-curves and the mechanisms of propagation in language change". In: *Language* (2012), pp. 269–304.

[4] M. De Groot. "The Soviet Union, CMEA, and the Energy Crisis of the 1970s". In: *Journal of Cold War Studies* 22.4 (2020), pp. 4–30.

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

[6] Q. Duong, M. Hämäläinen, and S. Hengchen. *An Unsupervised method for OCR Post-Correction and Spelling Normalisation for Finnish*. 2020. arXiv: 2011.03502 [cs.CL].

[7] J. C. B. Gamboa. "Deep Learning for Time-Series Analysis". In: *CoRR* abs/1701.01887 (2017). arXiv: 1701.01887. URL: http://arxiv.org/abs/1701.01887.

[8] M. Gao, T. Li, and P. Huang. "Text Classification Research Based on Improved Word2vec and CNN". In: *Service-Oriented Computing – ICSOC 2018 Workshops*. Ed. by X. Liu, M. Mrissa, L. Zhang, D. Benslimane, A. Ghose, Z. Wang, A. Bucchiarone, W. Zhang, Y. Zou, and Q. Yu. Cham: Springer International Publishing, 2019, pp. 126–135. ISBN: 978-3-030-17642-6.

[9] D. Hall, D. Jurafsky, and C. D. Manning. "Studying the history of ideas using topic models". In: *Proceedings of the 2008 conference on empirical methods in natural language processing*. 2008, pp. 363–371.

[10] Z. S. Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[11] S. Hengchen, R. Ros, and J. Marjanen. "A data-driven approach to the changing vocabulary of the nation in English, Dutch, Swedish and Finnish newspapers, 1750-1950". In: *Proceedings of the Digital Humanities (DH) conference*. 2019.

[12] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[13] D. H. Hubel and T. N. Wiesel. “Receptive fields and functional architecture of monkey striate cortex”. In: *The Journal of physiology* 195.1 (Mar. 1968). PMC1557912[pmcid], pp. 215–243. ISSN: 0022-3751. DOI: 10.1113/jphysiol.1968.sp008455. URL: https://doi.org/10.1113/jphysiol.1968.sp008455.

[14] M. Kestemont, F. Karsdorp, and M. During. “Mining the twentieth century’s history from the Time magazine corpus”. In: *Abstract book of EACL 2014: the 14th Conference of the European Chapter of the Association for Computational Linguistics*. 2014, p. 62.

[15] A. Kutuzov, L. Øvrelid, T. Szymanski, and E. Velldal. “Diachronic word embeddings and semantic shifts: a survey”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 1384–1397.

[16] Q. Le and T. Mikolov. “Distributed representations of sentences and documents”. In: *International conference on machine learning*. 2014, pp. 1188–1196.

[17] Y. Li, P. Nair, Z. Wen, I. Chafi, A. Okhmatovskaia, G. Powell, Y. Shen, and D. Buckeridge. “Global Surveillance of COVID-19 by mining news media using a multi-source dynamic embedded topic model”. In: *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. 2020, pp. 1–14.

[18] R. Light and J. Cunningham. “Oracles of peace: Topic modeling, cultural opportunity, and the Nobel peace prize, 1902–2012”. In: *Mobilization: An International Quarterly* 21.1 (2016), pp. 43–64.

[19] S. Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[20] J. Marjanen, E. Zosa, S. Hengchen, L. Pivovarova, and M. Tolonen. “Topic modelling discourse dynamics in historical newspapers”. In: (2021).

[21] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. URL: http://arxiv.org/abs/1301.3781.

[22] Mustaqeem and S. Kwon. “A CNN-Assisted Enhanced Audio Signal Processing for Speech Emotion Recognition”. In: *Sensors* 20.1 (2020). ISSN: 1424-8220. DOI: 10.3390/s20010183. URL: https://www.mdpi.com/1424-8220/20/1/183.

[23] D. J. Newman and S. Block. "Probabilistic topic decomposition of an eighteenth-century American newspaper". In: *Journal of the American Society for Information Science and Technology* 57.6 (2006), pp. 753–767.

[24] K. O'Shea and R. Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE].

[25] I. Parker. "Discourse: Definitions and Contradictions". In: *Critical Discursive Psychology*. London: Palgrave Macmillan UK, 2015, pp. 148–164. ISBN: 978-1-137-50527-9. DOI: 10.1057/9781137505279_13. URL: https://doi.org/10.1057/9781137505279_13.

[26] K. Pawar, R. S. Jalem, and V. Tiwari. "Stock Market Price Prediction Using LSTM RNN". In: *Emerging Trends in Expert Applications and Security*. Ed. by V. S. Rathore, M. Worring, D. K. Mishra, A. Joshi, and S. Maheshwari. Singapore: Springer Singapore, 2019, pp. 493–503. ISBN: 978-981-13-2285-3.

[27] A. Rosenfeld and K. Erk. "Deep neural models of semantic shift". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 474–484.

[28] D. Schlechtweg, S. S. im Walde, and S. Eckmann. "Diachronic Usage Relatedness (DURel): A Framework for the Annotation of Lexical Semantic Change". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, pp. 169–174.

[29] D. Schlechtweg and S. S. i. Walde. "Simulating Lexical Semantic Change from Sense-Annotated Data". In: *The Evolution of Language: Proceedings of the 13th International Conference (EvoLang13)*. 2020.

[30] J. Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. URL: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[31] M. Schuster and K. Paliwal. "Bidirectional Recurrent Neural Networks". In: *Trans. Sig. Proc.* 45.11 (Nov. 1997), pp. 2673–2681. ISSN: 1053-587X. DOI: 10.1109/78.650093. URL: https://doi.org/10.1109/78.650093.

[32] P. Shoemark, F. F. Liza, D. Nguyen, S. Hale, and B. McGillivray. "Room to Glo: A systematic comparison of semantic change detection approaches with word embeddings". 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)*. 2019, pp. 66–76.

[33] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[34] X. Tang. "A state-of-the-art of semantic change computation". In: *Natural Language Engineering* 24.5 (2018), pp. 649–676.

[35] C. Truong, L. Oudre, and N. Vayatis. "Selective review of offline change point detection methods". In: *Signal Processing* 167 (2020), p. 107299. ISSN: 0165-1684.

[36] A. Tsakalidis and M. Liakata. "Sequential Modelling of the Evolution of Word Representations for Semantic Change Detection". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 8485–8497.

[37] L. Viola and J. Verheul. "Mining ethnicity: Discourse-driven topic modelling of immigrant discourses in the USA, 1898–1920". In: *Digital Scholarship in the Humanities* 35.4 (2020), pp. 921–943.

[38] X. Wang and A. McCallum. "Topics over time: a non-Markov continuous-time model of topical trends". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 424–433.

[39] K. Xu, H. Wang, and P. Tang. "Image captioning with deep LSTM based on sequential residual". In: *2017 IEEE International Conference on Multimedia and Expo (ICME)*. 2017, pp. 361–366. DOI: 10.1109/ICME.2017.8019408.

[40] T.-I. Yang, A. Torget, and R. Mihalcea. "Topic modeling on historical newspapers". In: *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*. 2011, pp. 96–104.