UNIVERSITY OF AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Enriching Textual Data with Document Structure For Text Classification

by
Maarten de Jonge
10002109

September 21, 2018

42 EC
01 July 2017, 31 December 2017

Supervisor:
Dr Maarten Marx

Assessor:
Dr Maarten van Someren

Information and Language Processing Systems Group
Informatics Institute

Contents

1 Introduction **2**

2 Problem Statement **4**
2.1 Dataset . . . 7
2.2 Research Question . . . 9

3 Related Work **9**

4 Methodology **10**
4.1 Unsupervised . . . 10
4.1.1 Hierarchical Agglomerative Clustering . . . 11
4.1.2 Clustering . . . 13
4.1.3 Dirichlet Process Clustering . . . 14
4.2 Supervised . . . 16
4.2.1 Optimisation process . . . 20
4.2.2 Regularisation . . . 21
4.2.3 Convolutional Neural Networks . . . 22
4.2.4 Difference between convolutional and recurrent neural networks 24

5 Experimental Setup **24**
5.1 Dataset . . . 24
5.2 Testing performance . . . 25
5.3 CNN performance . . . 25
5.4 Generalisation . . . 25

6 Evaluation **25**
6.1 Parameter Exploration . . . 27
6.2 Training Set Size . . . 27

7 Conclusion **30**

Abstract

Proceedings of parliamentary debates are often published as unstructured PDF files, making them unsuitable for indexing into a database or querying for specific information. Digitizing them into a structured format is challenging; doing so manually is labor-intensive, doing so through a rule-based system is error-prone. Manually annotating a small number of documents can provide a training set that allows a convolutional neural network to accurately classify the elements that denote the structure of the document (e.g. the start of a new speech), using purely the textual content of the document. Adding a preprocessing step that clusters pieces of text based on the physical layout of the document improves the classification performance in a minor, but statistically significant way.

1 Introduction

A healthy democracy relies on a transparent political process that is open to the common populace

hier valt vast iets leuks te quoten van een politiek filosoof

. A common step towards achieving this is by publishing the proceedings of the parliament's debates, such as the *Bundestag* in Germany[1] or the *Tweede Kamer* in the Netherlands[2]. These proceedings can be useful in various ways, for example:

- Double-checking whether a certain politician's actions in the parliament are consistent with their public stance.
- Using them as a source of data for text analysis, such as classifying political ideology[1].
- Tracking the change in ideological leaning of a politician or party over time[2].

In each of these cases it would be hugely beneficial if the data was properly indexed. If you want to know John Doe's stance on immigration, you would ideally simply query a database for speeches by John Doe regarding the topic of immigration without having to manually skim over hundreds of documents to find the relevant speeches. It is unfortunate then that many of these debates are published solely in unstructured formats, such as PDF or plain text. Projects such as Political Mashup[3] handle this by writing systems to parse and then index these documents. The semantic information required for indexing is currently recovered using rule-based methods. In

[1]https://www.bundestag.de/protokolle
[2]https://www.tweedekamer.nl/kamerstukken
[3]http://search.politicalmashup.nl/about.html

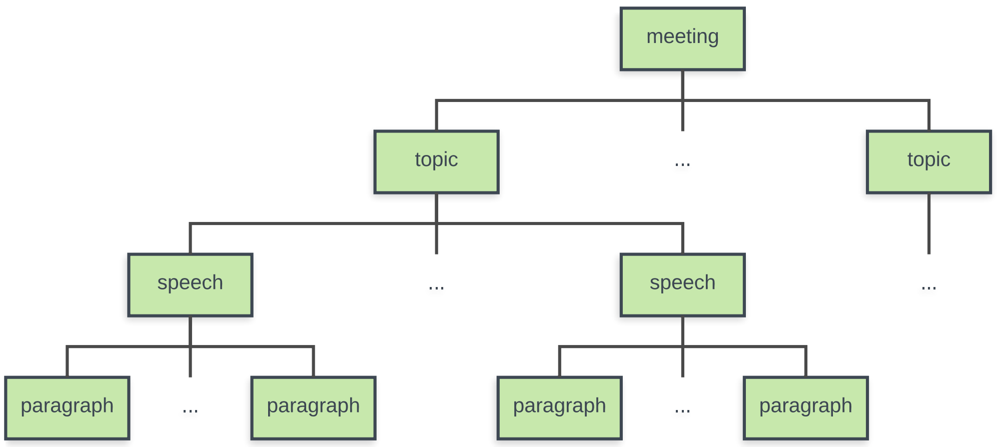


Figure 1 – The structure of a parliamentary debate is that of a shallow tree.

the case of a PDF document, this is fairly challenging. The data often gets transcribed by a human typist, compiled to a PDF, and then goes back into a PDF decompiler for easier processing; this adds a lot of places where minor variations can occur in the output even though the document itself uses a consistent layout. Dealing with this in a rule-based system entails using either highly general rules that lead to a large probability of false positives, or a large amount of highly specific rules which can quickly lead to a spaghetti-like mess of special cases and is very fragile to unseen cases.

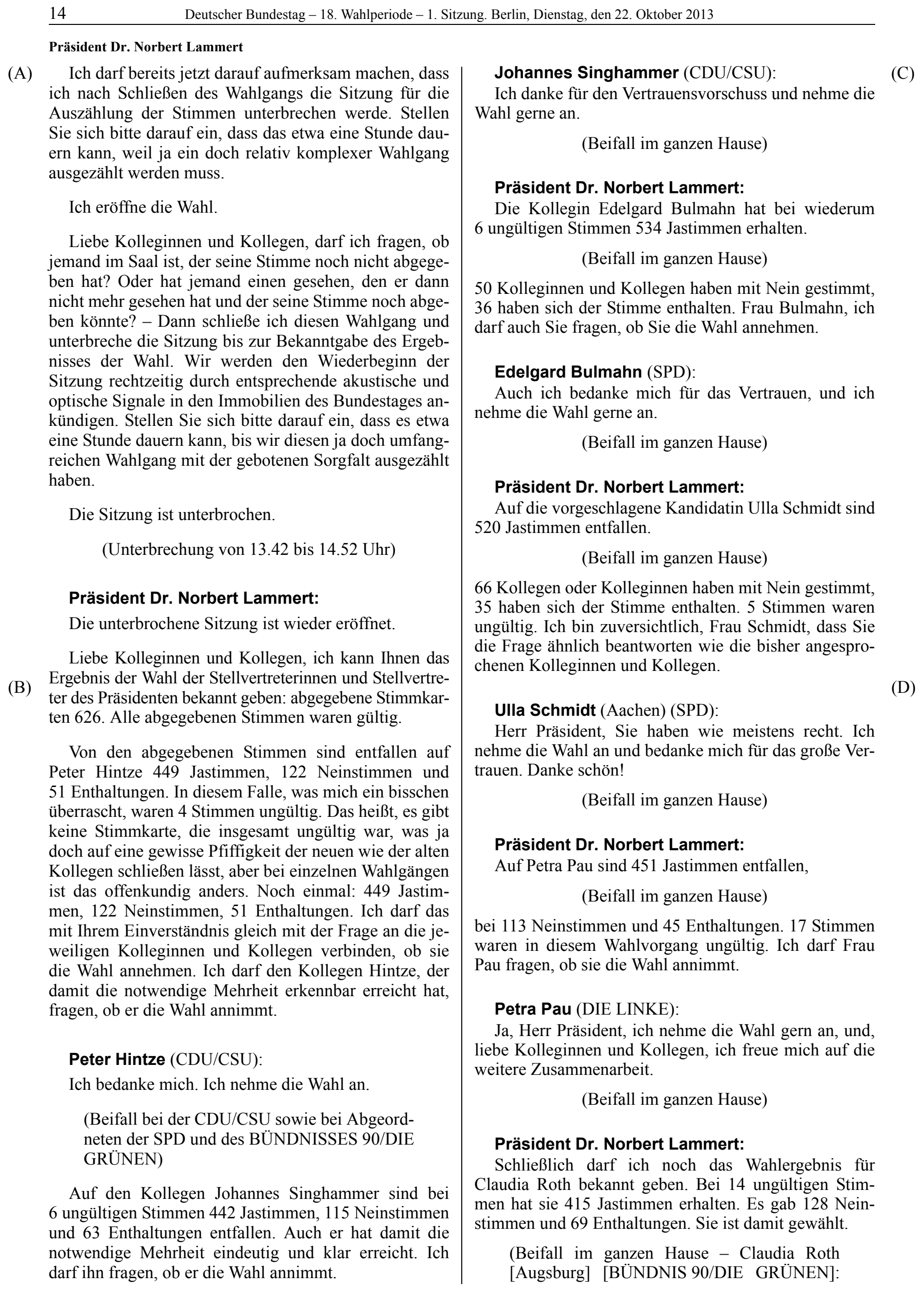
I propose that by using a small number of manually annotated documents as a dataset, a machine learning algorithm can learn to classify fragments of text in a way that allows it to segment a document into its constituent parts, while being more robust to noise than its rule-based counterpart. While the structure of a parliamentary debate is tree-shaped (a general example in shown in fig. 1), this tree is generally both shallow and very rigidly structured. As a result, simply classifying the positions in the text where a new structural element begins is enough to reconstruct the full tree given the domain knowledge about this particular set of documents; some variations can occur depending on the particular conventions used by the creator of the documents, for example in how speech interruptions by noisy colleagues are handled. After the document's layout has been extracted, it is a matter of obtaining each element's metadata (such as the speaker and their political affiliation for each speech element). This step is outside of the scope of this thesis.

The common ways to do sentence classification (e.g. convolutional neural networks [3], recurrent neural networks or the simpler bag-of-words models) operate on sentences in a vacuum, considering only their linguistic contents and ignoring any contextual information that might be contained in the layout in which the text might have been embedded. This is to be expected considering that most of the common datasets in this area really *are* just small bits of text in a vacuum; often-used datasets involve Twitter messages or short product reviews. In this case however, the sentences come from a document with a rich structure providing a lot of context. Anecdotally, as a human it is trivial to discern section headers in a document even when the document is in a foreign language; simply the fact that the section header might be printed in bold and centered rather than left-aligned gives it away. Incorporating this structural data into the learning process, using a clustering approach inspired by Klampfl *et al.* [4], will hopefully increase the performance of the system, either by simply scoring better on the used metrics, or perhaps more indirectly by requiring less data or training time to achieve the same score.

2 Problem Statement

The German parliament, called the *Bundestag*, publishes the proceedings of their meetings, as an effort to open up the political process to the common people. These proceedings have been continously published since the inception of the German Empire in 1871, when the parliament was called the *Reichstag*, up to 1942; publishing resumed after the conclusion of World War 2 with the inception of the Bundestag in 1949. Of course, the further back in the time you go,the more difficult the documents become to use. While the more recent documents are fully digital, documents prior to 1998 are scans of physical documents and require optical character recognition, which becomes even more problematic in the documents from the 1800s where a thick Gothic font is used. Figure 2 shows a sample page from one of these proceedings; the left column contains a continuation of a speech from the previous page as well as two moderately sized speeches, while the right column contains a large number of very short speeches. Figure 3 shows the same page seen in fig. 2, but with a number of regions of interest (manually) colored in. Unfortunately this data is only available as PDF files, which in terms of internal representation are entirely unstructured. That means that none of the rich structure highlighted in fig. 3 is actually present in a computer-usable way.

The central problem in this thesis is the extraction of speeches from the documents, transforming each document into a structured series of speeches that can be serve as a useful entry point into further research. As a first step, the structural layout (as in fig. 3) will be extracted using unsupervised clustering algorithms. The textual contents of the document are then fed into supervised text classifier, where each piece of text is augmented with the corresponding layout information previously obtained. More details on



14 Deutscher Bundestag – 18. Wahlperiode – 1. Sitzung. Berlin, Dienstag, den 22. Oktober 2013

Präsident Dr. Norbert Lammert

(A) Ich darf bereits jetzt darauf aufmerksam machen, dass ich nach Schließen des Wahlgangs die Sitzung für die Auszählung der Stimmen unterbrechen werde. Stellen Sie sich bitte darauf ein, dass das etwa eine Stunde dauern kann, weil ja ein doch relativ komplexer Wahlgang ausgezählt werden muss.

Ich eröffne die Wahl.

Liebe Kolleginnen und Kollegen, darf ich fragen, ob jemand im Saal ist, der seine Stimme noch nicht abgegeben hat? Oder hat jemand einen gesehen, den er dann nicht mehr gesehen hat und der seine Stimme noch abgeben könnte? – Dann schließe ich diesen Wahlgang und unterbreche die Sitzung bis zur Bekanntgabe des Ergebnisses der Wahl. Wir werden den Wiederbeginn der Sitzung rechtzeitig durch entsprechende akustische und optische Signale in den Immobilien des Bundestages ankündigen. Stellen Sie sich bitte darauf ein, dass es etwa eine Stunde dauern kann, bis wir diesen ja doch umfangreichen Wahlgang mit der gebotenen Sorgfalt ausgezählt haben.

Die Sitzung ist unterbrochen.

(Unterbrechung von 13.42 bis 14.52 Uhr)

Präsident Dr. Norbert Lammert:
Die unterbrochene Sitzung ist wieder eröffnet.

(B) Liebe Kolleginnen und Kollegen, ich kann Ihnen das Ergebnis der Wahl der Stellvertreterinnen und Stellvertreter des Präsidenten bekannt geben: abgegebene Stimmkarten 626. Alle abgegebenen Stimmen waren gültig.

Von den abgegebenen Stimmen sind entfallen auf Peter Hintze 449 Jastimmen, 122 Neinstimmen und 51 Enthaltungen. In diesem Falle, was mich ein bisschen überrascht, waren 4 Stimmen ungültig. Das heißt, es gibt keine Stimmkarte, die insgesamt ungültig war, was ja doch auf eine gewisse Pfiffigkeit der neuen wie der alten Kollegen schließen lässt, aber bei einzelnen Wahlgängen ist das offenkundig anders. Noch einmal: 449 Jastimmen, 122 Neinstimmen, 51 Enthaltungen. Ich darf das mit Ihrem Einverständnis gleich mit der Frage an die jeweiligen Kolleginnen und Kollegen verbinden, ob sie die Wahl annehmen. Ich darf den Kollegen Hintze, der damit die notwendige Mehrheit erkennbar erreicht hat, fragen, ob er die Wahl annimmt.

Peter Hintze (CDU/CSU):
Ich bedanke mich. Ich nehme die Wahl an.

(Beifall bei der CDU/CSU sowie bei Abgeordneten der SPD und des BÜNDNISSES 90/DIE GRÜNEN)

Auf den Kollegen Johannes Singhammer sind bei 6 ungültigen Stimmen 442 Jastimmen, 115 Neinstimmen und 63 Enthaltungen entfallen. Auch er hat damit die notwendige Mehrheit eindeutig und klar erreicht. Ich darf ihn fragen, ob er die Wahl annimmt.

Johannes Singhammer (CDU/CSU): (C)
Ich danke für den Vertrauensvorschuss und nehme die Wahl gerne an.

(Beifall im ganzen Hause)

Präsident Dr. Norbert Lammert:
Die Kollegin Edelgard Bulmahn hat bei wiederum 6 ungültigen Stimmen 534 Jastimmen erhalten.

(Beifall im ganzen Hause)

50 Kolleginnen und Kollegen haben mit Nein gestimmt, 36 haben sich der Stimme enthalten. Frau Bulmahn, ich darf auch Sie fragen, ob Sie die Wahl annehmen.

Edelgard Bulmahn (SPD):
Auch ich bedanke mich für das Vertrauen, und ich nehme die Wahl gerne an.

(Beifall im ganzen Hause)

Präsident Dr. Norbert Lammert:
Auf die vorgeschlagene Kandidatin Ulla Schmidt sind 520 Jastimmen entfallen.

(Beifall im ganzen Hause)

66 Kollegen oder Kolleginnen haben mit Nein gestimmt, 35 haben sich der Stimme enthalten. 5 Stimmen waren ungültig. Ich bin zuversichtlich, Frau Schmidt, dass Sie die Frage ähnlich beantworten wie die bisher angesprochenen Kolleginnen und Kollegen. (D)

Ulla Schmidt (Aachen) (SPD):
Herr Präsident, Sie haben wie meistens recht. Ich nehme die Wahl an und bedanke mich für das große Vertrauen. Danke schön!

(Beifall im ganzen Hause)

Präsident Dr. Norbert Lammert:
Auf Petra Pau sind 451 Jastimmen entfallen,

(Beifall im ganzen Hause)

bei 113 Neinstimmen und 45 Enthaltungen. 17 Stimmen waren in diesem Wahlvorgang ungültig. Ich darf Frau Pau fragen, ob sie die Wahl annimmt.

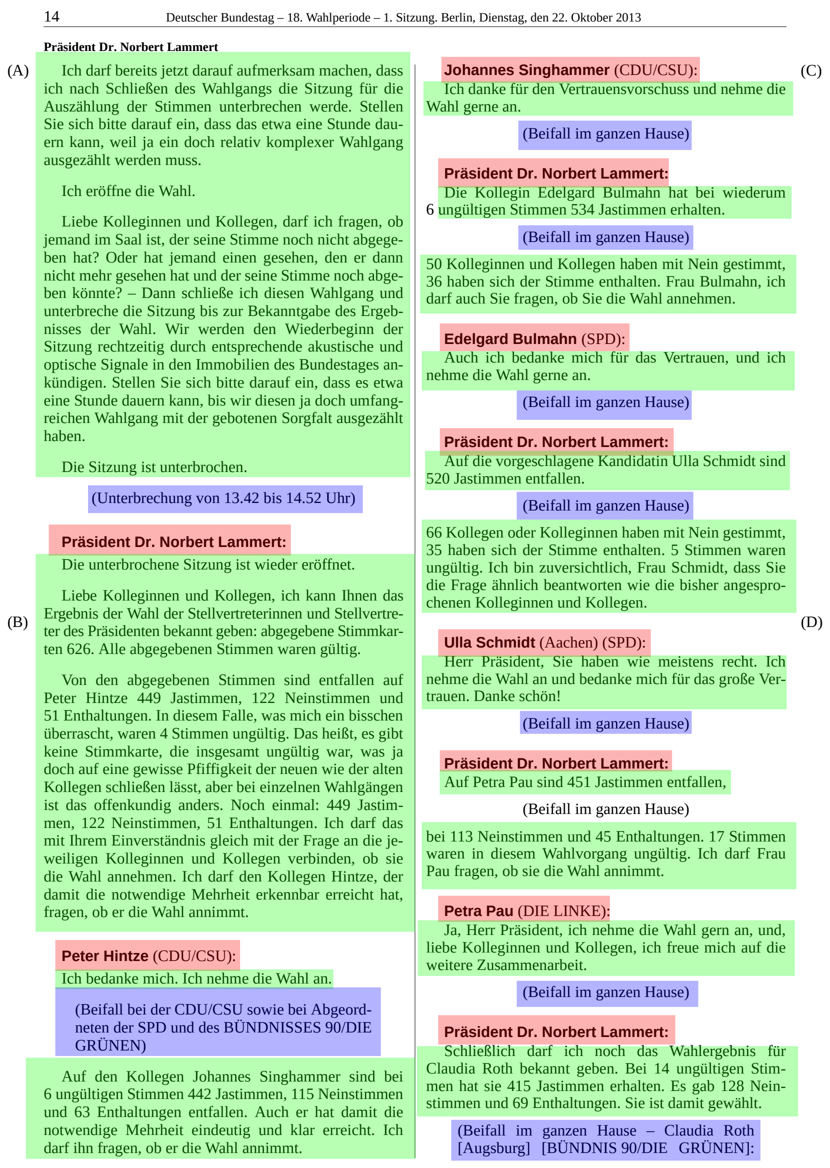
Petra Pau (DIE LINKE):
Ja, Herr Präsident, ich nehme die Wahl gern an, und, liebe Kolleginnen und Kollegen, ich freue mich auf die weitere Zusammenarbeit.

(Beifall im ganzen Hause)

Präsident Dr. Norbert Lammert:
Schließlich darf ich noch das Wahlergebnis für Claudia Roth bekannt geben. Bei 14 ungültigen Stimmen hat sie 415 Jastimmen erhalten. Es gab 128 Neinstimmen und 69 Enthaltungen. Sie ist damit gewählt.

(Beifall im ganzen Hause – Claudia Roth [Augsburg] [BÜNDNIS 90/DIE GRÜNEN]:

Figure 2 – A sample page from one of the Bundestag proceedings.



14 Deutscher Bundestag – 18. Wahlperiode – 1. Sitzung. Berlin, Dienstag, den 22. Oktober 2013

Präsident Dr. Norbert Lammert

(A) Ich darf bereits jetzt darauf aufmerksam machen, dass ich nach Schließen des Wahlgangs die Sitzung für die Auszählung der Stimmen unterbrechen werde. Stellen Sie sich bitte darauf ein, dass das etwa eine Stunde dauern kann, weil ja ein doch relativ komplexer Wahlgang ausgezählt werden muss.

Ich eröffne die Wahl.

Liebe Kolleginnen und Kollegen, darf ich fragen, ob jemand im Saal ist, der seine Stimme noch nicht abgegeben hat? Oder hat jemand einen gesehen, den er dann nicht mehr gesehen hat und der seine Stimme noch abgeben könnte? – Dann schließe ich diesen Wahlgang und unterbreche die Sitzung bis zur Bekanntgabe des Ergebnisses der Wahl. Wir werden den Wiederbeginn der Sitzung rechtzeitig durch entsprechende akustische und optische Signale in den Immobilien des Bundestages ankündigen. Stellen Sie sich bitte darauf ein, dass es etwa eine Stunde dauern kann, bis wir diesen ja doch umfangreichen Wahlgang mit der gebotenen Sorgfalt ausgezählt haben.

Die Sitzung ist unterbrochen.

(Unterbrechung von 13.42 bis 14.52 Uhr)

Präsident Dr. Norbert Lammert:

Die unterbrochene Sitzung ist wieder eröffnet.

(B) Liebe Kolleginnen und Kollegen, ich kann Ihnen das Ergebnis der Wahl der Stellvertreterinnen und Stellvertreter des Präsidenten bekannt geben: abgegebene Stimmkarten 626. Alle abgegebenen Stimmen waren gültig.

Von den abgegebenen Stimmen sind entfallen auf Peter Hintze 449 Jastimmen, 122 Neinstimmen und 51 Enthaltungen. In diesem Falle, was mich ein bisschen überrascht, waren 4 Stimmen ungültig. Das heißt, es gibt keine Stimmkarte, die insgesamt ungültig war, was ja doch auf eine gewisse Pfiffigkeit der neuen wie der alten Kollegen schließen lässt, aber bei einzelnen Wahlgängen ist das offenkundig anders. Noch einmal: 449 Jastimmen, 122 Neinstimmen, 51 Enthaltungen. Ich darf das mit Ihrem Einverständnis gleich mit der Frage an die jeweiligen Kolleginnen und Kollegen verbinden, ob sie die Wahl annehmen. Ich darf den Kollegen Hintze, der damit die notwendige Mehrheit erkennbar erreicht hat, fragen, ob er die Wahl annimmt.

Peter Hintze (CDU/CSU):

Ich bedanke mich. Ich nehme die Wahl an.

(Beifall bei der CDU/CSU sowie bei Abgeordneten der SPD und des BÜNDNISSES 90/DIE GRÜNEN)

Auf den Kollegen Johannes Singhammer sind bei 6 ungültigen Stimmen 442 Jastimmen, 115 Neinstimmen und 63 Enthaltungen entfallen. Auch er hat damit die notwendige Mehrheit eindeutig und klar erreicht. Ich darf ihn fragen, ob er die Wahl annimmt.

Johannes Singhammer (CDU/CSU): (C)

Ich danke für den Vertrauensvorschuss und nehme die Wahl gerne an.

(Beifall im ganzen Hause)

Präsident Dr. Norbert Lammert:

Die Kollegin Edelgard Bulmahn hat bei wiederum 6 ungültigen Stimmen 534 Jastimmen erhalten.

(Beifall im ganzen Hause)

50 Kolleginnen und Kollegen haben mit Nein gestimmt, 36 haben sich der Stimme enthalten. Frau Bulmahn, ich darf auch Sie fragen, ob Sie die Wahl annehmen.

Edelgard Bulmahn (SPD):

Auch ich bedanke mich für das Vertrauen, und ich nehme die Wahl gerne an.

(Beifall im ganzen Hause)

Präsident Dr. Norbert Lammert:

Auf die vorgeschlagene Kandidatin Ulla Schmidt sind 520 Jastimmen entfallen.

(Beifall im ganzen Hause)

66 Kollegen oder Kolleginnen haben mit Nein gestimmt, 35 haben sich der Stimme enthalten. 5 Stimmen waren ungültig. Ich bin zuversichtlich, Frau Schmidt, dass Sie die Frage ähnlich beantworten wie die bisher angesprochenen Kolleginnen und Kollegen. (D)

Ulla Schmidt (Aachen) (SPD):

Herr Präsident, Sie haben wie meistens recht. Ich nehme die Wahl an und bedanke mich für das große Vertrauen. Danke schön!

(Beifall im ganzen Hause)

Präsident Dr. Norbert Lammert:

Auf Petra Pau sind 451 Jastimmen entfallen,

(Beifall im ganzen Hause)

bei 113 Neinstimmen und 45 Enthaltungen. 17 Stimmen waren in diesem Wahlvorgang ungültig. Ich darf Frau Pau fragen, ob sie die Wahl annimmt.

Petra Pau (DIE LINKE):

Ja, Herr Präsident, ich nehme die Wahl gern an, und, liebe Kolleginnen und Kollegen, ich freue mich auf die weitere Zusammenarbeit.

(Beifall im ganzen Hause)

Präsident Dr. Norbert Lammert:

Schließlich darf ich noch das Wahlergebnis für Claudia Roth bekannt geben. Bei 14 ungültigen Stimmen hat sie 415 Jastimmen erhalten. Es gab 128 Neinstimmen und 69 Enthaltungen. Sie ist damit gewählt.

(Beifall im ganzen Hause – Claudia Roth [Augsburg] [BÜNDNIS 90/DIE GRÜNEN]:

Figure 3 – The same page as in Figure 2, hand-annotated with interesting regions that play a significant role in understanding the layout. The red regions are the headers the signify that someone is starting a speech, and contain information about who the speaker is. The blue regions are little interruptions (*Heiterkeits*), often signifying approval or displeasure (the regularly seen *Beifall* means applause). The green regions indicate plain blocks of text.

this process will be supplied in section 4.

As annotating data by hand is an expensive process, a big focus is on limiting the amount of required training data as much as possible; it would be preferable if the system was able to learn sufficiently from a handful (say, less than 5) of hand-annotated files.

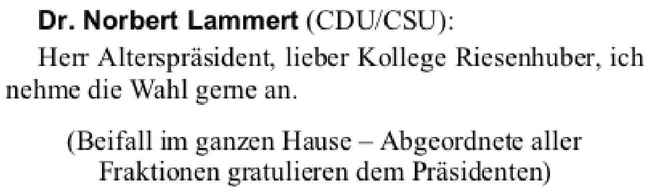
2.1 Dataset

The dataset contains 11 hand-annotated documents, comprising 2052 positive samples and 76,944 negative samples. The average document contains about 200 positive samples. Seeing as the source documents are PDF files, some transformations are needed to extract text from them in a way that is suitable for use as a machine learning dataset. This is not as trivial as it might seem, given that PDF files only contain instructions for drawing certain characters at certain coordinates, with no internal concept of paragraphs, lines or even words.

For the supervised portion of the system, the PDF files are transformed using the `pdftohtml` utility from the Poppler PDF rendering library[4]. This utility takes a PDF file and uses a number of heuristics to output lines of text occurring inside the file. In this context, a *line* refers to any number of characters that occur on the same height on a page while preserving reading order (which is relevant when dealing with documents that have a two-column layout) and is explicitly not the same as a sentence. These lines are output in an XML format which includes metadata on the geometry of the line (position and size) and its font. An example of a portion of text from the dataset and the corresponding XML output from `pdftohtml` is shown in fig. 4. Since this process is based on heuristics, it can and does go wrong; there are instances of mistakes such as a single line being broken up into multiple XML nodes, or lines of two side-by-side columns being taken as a single XML node. This is not terribly common (the frequency of such errors is perhaps one per file on average), but it does make the dataset inherently noisy and is one of the issues that rule-based systems have trouble with.

For the unsupervised clustering, the PDF files are taken directly as input and clustering is done using individual characters as the basic unit. This is just as easy as clustering on the lines produced by `pdftohtml`, but has the benefit of not being dependent on the imperfect line-extraction heuristics.

[4]https://poppler.freedesktop.org/



Dr. Norbert Lammert (CDU/CSU):

Herr Alterspräsident, lieber Kollege Riesenhuber, ich nehme die Wahl gerne an.

(Beifall im ganzen Hause – Abgeordnete aller Fraktionen gratulieren dem Präsidenten)

(a) A portion of the source PDF.

```
<text top="122" left="125" width="143" height="16" font="3">
     Dr. Norbert Lammert
</text>
<text top="122" left="269" width="83" height="17" font="4">
     (CDU/CSU):
</text>
<text top="142" left="125" width="328" height="17" font="4">
     Herr Alterspräsident , lieber Kollege Riesenhuber , ich
</text>
<text top="158" left="108" width="156" height="17" font="4">
     nehme die Wahl gerne an.
</text>
<text top="186" left="141" width="278" height="17" font="4">
     (Beifall im ganzen Hause  Abgeordnete aller
</text>
<text top="203" left="158" width="242" height="17" font="4">
     Fraktionen gratulieren dem Präsidenten)
</text>
```

(b) XML created by running `pdftohtml`, corresponding to the PDF excerpt in Figure 4a. The contents of the `text` elements are used as inputs for the classification algorithm; the layout data contained in the properties is not used, as a separate software pipeline is used for the unsupervised clustering.

Figure 4 – A sample excerpt from a source PDF, along with its XML representation created by `pdftohtml`.

2.2 Research Question

A baseline model can be created by leaving out the clustering step, leaving us with two models:

1. A baseline model that classifies based on purely text
2. A model that classifies based on both text and layout information

There are two ways in which the second model can improve upon the baseline: either it performs better (using a metric such as the F1 score), or it requires less data to reach the same performance. This naturally leads to two research questions:

Research Question 1 *Does augmenting a text classification system with layout information obtained by unsupervised clustering of the input data improve the F1 score of the classifier?*

Research Question 2 *Does augmenting a text classification system with layout information obtained by unsupervised clustering of the input data allow the classifier to reach its peak performance using less input data?*

3 Related Work

The task handled in this thesis is in a way similar to that of wrapper induction, which is the process of inferring a *wrapper* (a program that extracts data into a usable form) from a web page. A fairly recent survey of the state of this field is done by Ferrara *et al.* [5], who note that a big problem is keeping up with the constantly changing nature of web pages. A novel approach to combat this is that of Gogar *et al.* [6]. They do wrapper induction by combining the textual content of a webpage with a screenshot of the rendered webpage in an effort to do wrapper induction on previously unseen web pages. The text is encoded in a way that maintains spatial information, a model they refer to as *Text Maps* or *Spatial bag of words.* The text maps and the screenshot are fed into separate convolutional networks, after which the output is combined for a final classification. In a test of extracting product names and prices from web pages, the system obtains very high score comparable to systems that do use site-specific initialization.

In terms of analyzing document structure, Klampfl *et al.* [4] introduce a method to analyze scientific articles, detecting blocks of text, labeling them (as e.g. section headers, tables or references) and determining the reading order — all in an unsupervised manner. The text block detection is done using a sequence of different clustering algorithms, while the labeling is done using a heuristic approach.

For text classification, various forms of convolutional neural networks are commonly used. The most basic architecture is described by Kim [3], where the input is tokenized and the tokens are embedded into a higher-dimensional representation before passing them to the convolutional neural network. Using this same architecture, Zhang & Wallace [7] perform additional exploration of the parameters and their effects on various datasets. Comparable results are achieved by Zhang *et al.* [8] by operating on the character-level rather than the word-level, bypassing the overhead of using word embedding (either in extra training time or in finding suitable pretrained embeddings) as well as freeing the researcher from having another layer in their network to tune. All the previously mentioned architectures use a single convolutional layer; this is contrary to current trends in computer vision, where popular models such as ResNet[9] go as deep as 152 layers. This difference is explored by Conneau *et al.* [10], who take a character-level CNN and show that adding more layers improves performance, before leveling out at 29 layers. They hypothesize that the difference in effective depth between computer vision and language processing might be due to the difference in datasets. The common ImageNet dataset used in computer vision deals with 1000 classes; in contrast, sentiment analysis datasets vary between 2 and 25 classes. In addition, they note that the deeper networks do require a larger amount of data to train.

4 Methodology

As described in section 2, the input data comes in the form of a PDF file which can be converted into plaintext XML data containing lines of text along with each line's bounding box. There are two systems in play here; the main system is a convolutional neural network classifier that acts on the XML data. Further details on this are provided in section 4.2. The second system is the unsupervised preprocessor whichs acts on the PDF data and writes its output as an additional property into the XML data, which is elaborated upon in section 4.1. Figure 5 shows a high-level overview of the two systems and how they interact.

4.1 Unsupervised

The unsupervised algorithm is inteded to detect and classify blocks of text in the PDF file; Figure 6 shows an example. This approach is based on work by Klampfl *et al.* [4], and consists of two separate clustering steps. First, individual characters (the fundamental objects available in a PDF file) are clustered together into blocks of semantically relevant text. These could be, for example, paragraphs, section headers or page decoration. By using the bounding boxes of the blocks, they can be clustered based on their shape and some additional metadata (e.g. occurrence of font types and sizes). The rest of this section will go into details on the two clustering steps.

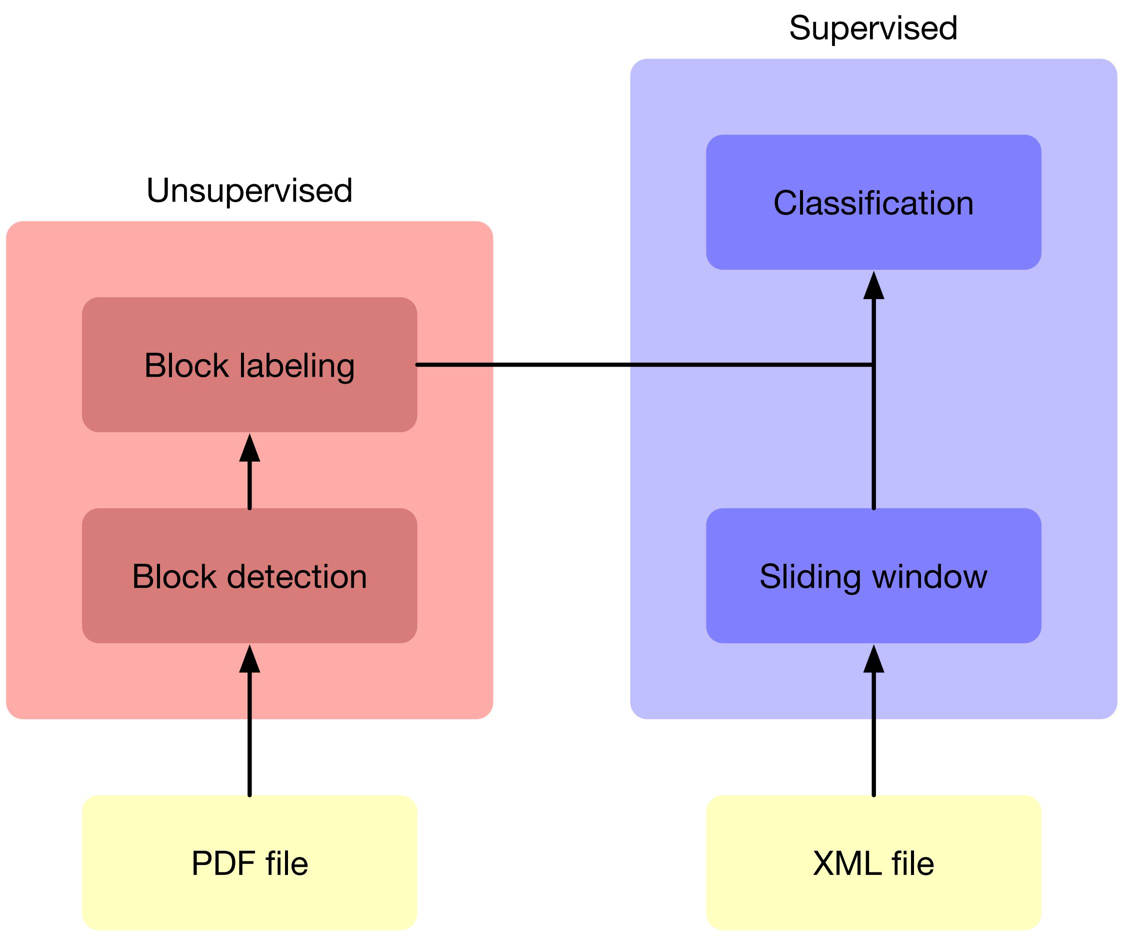


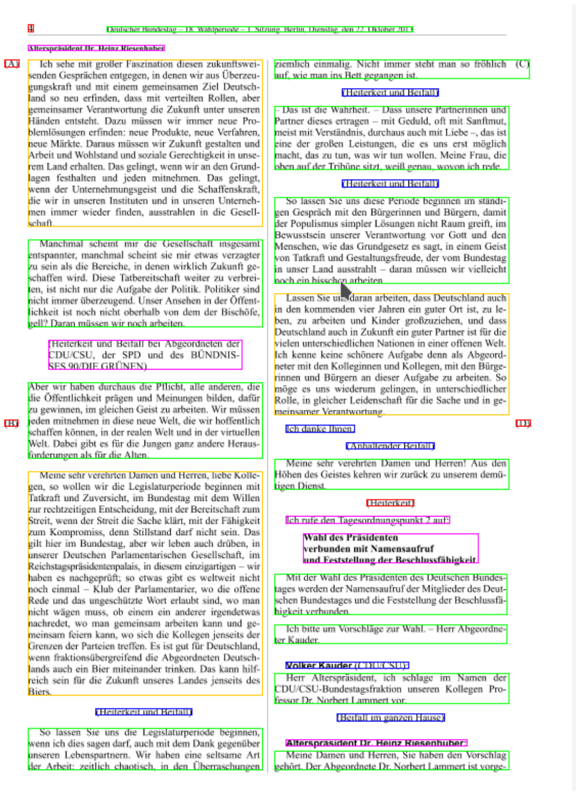
Figure 5 – A high-level overview of the system. The unsupervised block augments the input to the classifier.

4.1.1 Hierarchical Agglomerative Clustering

The first step is performed using hierarchical agglomerative clustering (HAC), an unsupervised bottom-up clustering algorithm that constructs a hierarchical tree of clusters (in this context referred to as a *dendrogram*). An example is shown in fig. 7. The algorithm gets fed the individual characters present in the PDF files, then iteratively groups the two closest clusters (the initial inputs being regarded as clusters of one element) together until only a single cluster remains. This process involves two parameters:

1. The distance function between two characters.
2. The distance function between two clusters of characters.

The first parameter is trivially chosen to be the Euclidian distance between the coordinates of the two characters. The second parameter is called the



4 Deutscher Bundestag – 18. Wahlperiode – 1. Sitzung. Berlin, Dienstag, den 22. Oktober 2013

Alterspräsident Dr. Heinz Riesenhuber

(A) Ich sehe mit großer Faszination diesen zukunftsweisenden Gesprächen entgegen, in denen wir aus Überzeugungskraft und mit einem gemeinsamen Ziel Deutschland so neu erfinden, dass mit verteilten Rollen, aber gemeinsamer Verantwortung die Zukunft unter unseren Händen entsteht. Dazu müssen wir immer neue Problemlösungen erfinden: neue Produkte, neue Verfahren, neue Märkte. Daraus müssen wir Zukunft gestalten und Arbeit und Wohlstand und soziale Gerechtigkeit in unserem Land erhalten. Das gelingt, wenn wir an den Grundlagen festhalten und jeden mitnehmen. Das gelingt, wenn der Unternehmungsgeist und die Schaffenskraft, die wir in unseren Instituten und in unseren Unternehmen immer wieder finden, ausstrahlen in die Gesellschaft.

Manchmal scheint mir die Gesellschaft insgesamt entspannter, manchmal scheint sie mir etwas verzagter zu sein als die Bereiche, in denen wirklich Zukunft geschaffen wird. Diese Tatbereitschaft weiter zu verbreiten, ist nicht nur die Aufgabe der Politik. Politiker sind nicht immer überzeugend. Unser Ansehen in der Öffentlichkeit ist noch nicht oberhalb von dem der Bischöfe, gell? Daran müssen wir noch arbeiten.

(Heiterkeit und Beifall bei Abgeordneten der CDU/CSU, der SPD und des BÜNDNISSES 90/DIE GRÜNEN)

Aber wir haben durchaus die Pflicht, alle anderen, die die Öffentlichkeit prägen und Meinungen bilden, dafür zu gewinnen, im gleichen Geist zu arbeiten. Wir müssen (B) jeden mitnehmen in diese neue Welt, die wir hoffentlich schaffen können, in der realen Welt und in der virtuellen Welt. Dabei gibt es für die Jungen ganz andere Herausforderungen als für die Alten.

Meine sehr verehrten Damen und Herren, liebe Kollegen, so wollen wir die Legislaturperiode beginnen mit Tatkraft und Zuversicht, im Bundestag mit dem Willen zur rechtzeitigen Entscheidung, mit der Bereitschaft zum Streit, wenn der Streit die Sache klärt, mit der Fähigkeit zum Kompromiss, denn Stillstand darf nicht sein. Das gilt hier im Bundestag, aber wir leben auch drüben, in unserer Deutschen Parlamentarischen Gesellschaft, im Reichstagspräsidentenpalais, in diesem einzigartigen – wir haben es nachgeprüft; so etwas gibt es weltweit nicht noch einmal – Klub der Parlamentarier, wo die offene Rede und das ungeschützte Wort erlaubt sind, wo man nicht wägen muss, ob einem ein anderer irgendetwas nachredet, wo man gemeinsam arbeiten kann und gemeinsam feiern kann, wo sich die Kollegen jenseits der Grenzen der Parteien treffen. Es ist gut für Deutschland, wenn fraktionsübergreifend die Abgeordneten Deutschlands auch ein Bier miteinander trinken. Das kann hilfreich sein für die Zukunft unseres Landes jenseits des Biers.

(Heiterkeit und Beifall)

So lassen Sie uns die Legislaturperiode beginnen, wenn ich dies sagen darf, auch mit dem Dank gegenüber unseren Lebenspartnern. Wir haben eine seltsame Art der Arbeit: zeitlich chaotisch, in den Überraschungen ziemlich einmalig. Nicht immer steht man so fröhlich (C) auf, wie man ins Bett gegangen ist.

(Heiterkeit und Beifall)

– Das ist die Wahrheit. – Dass unsere Partnerinnen und Partner dieses ertragen – mit Geduld, oft mit Sanftmut, meist mit Verständnis, durchaus auch mit Liebe –, das ist eine der großen Leistungen, die es uns erst möglich macht, das zu tun, was wir tun wollen. Meine Frau, die oben auf der Tribüne sitzt, weiß genau, wovon ich rede.

(Heiterkeit und Beifall)

So lassen Sie uns diese Periode beginnen im ständigen Gespräch mit den Bürgerinnen und Bürgern, damit der Populismus simpler Lösungen nicht Raum greift, im Bewusstsein unserer Verantwortung vor Gott und den Menschen, wie das Grundgesetz es sagt, in einem Geist von Tatkraft und Gestaltungsfreude, der vom Bundestag in unser Land ausstrahlt – daran müssen wir vielleicht noch ein bisschen arbeiten.

Lassen Sie uns daran arbeiten, dass Deutschland auch in den kommenden vier Jahren ein guter Ort ist, zu leben, zu arbeiten und Kinder großzuziehen, und dass Deutschland auch in Zukunft ein guter Partner ist für die vielen unterschiedlichen Nationen in einer offenen Welt. Ich kenne keine schönere Aufgabe denn als Abgeordneter mit den Kolleginnen und Kollegen, mit den Bürgerinnen und Bürgern an dieser Aufgabe zu arbeiten. So möge es uns wiederum gelingen, in unterschiedlicher Rolle, in gleicher Leidenschaft für die Sache und in gemeinsamer Verantwortung.

Ich danke Ihnen. (D)

(Anhaltender Beifall)

Meine sehr verehrten Damen und Herren! Aus den Höhen des Geistes kehren wir zurück zu unserem demütigen Dienst.

(Heiterkeit)

Ich rufe den Tagesordnungspunkt 2 auf:

Wahl des Präsidenten verbunden mit Namensaufruf und Feststellung der Beschlussfähigkeit

Mit der Wahl des Präsidenten des Deutschen Bundestages werden der Namensaufruf der Mitglieder des Deutschen Bundestages und die Feststellung der Beschlussfähigkeit verbunden.

Ich bitte um Vorschläge zur Wahl. – Herr Abgeordneter Kauder.

Volker Kauder (CDU/CSU):

Herr Alterspräsident, ich schlage im Namen der CDU/CSU-Bundestagsfraktion unseren Kollegen Professor Dr. Norbert Lammert vor.

(Beifall im ganzen Hause)

Alterspräsident Dr. Heinz Riesenhuber:

Meine Damen und Herren, Sie haben den Vorschlag gehört. Der Abgeordnete Dr. Norbert Lammert ist vorge-

Figure 6 – An example of clustered blocks of text. Blocks with the same outline color belong to the same cluster.

linkage and has several common options, the most basic of which are:

- Single-linkage: The distance between clusters is based on the closest two elements:

$$d(A, B) = \min\{d(a, b) : a \in A, b \in B\}$$

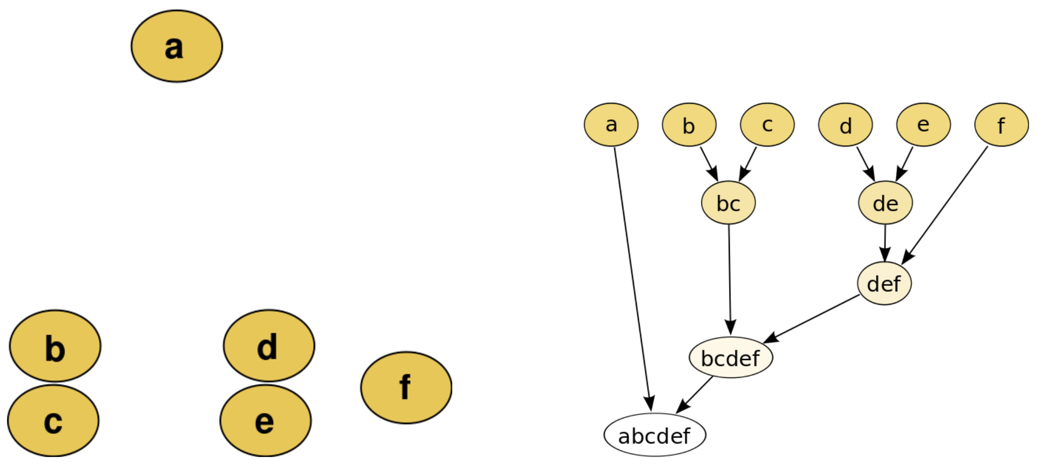
- Maximum-linkage: The distance between clusters is based on the furthest two elements:

$$d(A, B) = \max\{d(a, b) : a \in A, b \in B\}$$

- Average-linkage: The distance between clusters is based on the average distance of its elements:

$$d(A, B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$$

Additionally there are more involved linkage criteria, such as the Ward method which minimizes the variance within clusters. Although these more complex



(a) Before (b) After

Figure 7 – An example of hierarical agglomerative clustering, where the nodes are clustered by distance.

methods would generally be favored, in this specific context single-linkage clusters is actually the best option[4]. This is due to its tendency to form long, thin clusters; this mirrors the nature of text, in particular words and sentences (which are just long, thin strings of letters and words respectively). As an additional bonus, it is the most computationally efficient method. While the general time complexity for HAC is in $\mathcal{O}(n^3)$, clever algorithms exist for single-linkage clustering that fall in $\mathcal{O}(n^2)$ [11], making it far more usable on larger datasets.

After the dendrogram is constructed, it has to be cut at some level to obtain the desired blocks of text. Clustering can optionally be rerun using the newly found clusters as basic elements. This way, the document can incrementally be clustered from characters into words, words into lines, and finally lines into paragraphs. Both the level at which to cut the tree and the number of times to recluster are to be manually tuned based on the particular set of documents.

4.1.2 Clustering

The extracted blocks from the previous step are then clustered according to their similarity based on the following metrics:

- Width of the block
- Height of the block
- The ID of the most common font occurring in the block
- The size of the most common font occurring in the block

This is implemented two different ways, with their performance to be compared later on. The first method is a simple K-Means clustering, which works as follows:

1. Randomly initialize k cluster centroids.
2. Assign each observation to its nearest centroid.
3. Recompute each centroid to be the mean of all of its assigned observations.
4. Repeat steps 2 and 3 until the assignments stop changing.

After the algorithm has converged, the clustering is defined by each observation's nearest cluster centroid; with k clusters, each observation is assigned a k-dimensional one-hot vector.

4.1.3 Dirichlet Process Clustering

K-Means clustering can be generalized to a *mixture of Gaussians* model. Whereas K-Means clustering defines each cluster by a centroid and each assignment by its nearest cluster, a mixture of Gaussians — as the name implies — defines each cluster by a Gaussian distribution and each assignment by a vector indicating the probabilities of belonging to each cluster. This adds a degree of uncertainty to the representation, which is lost in K-Means' hard assignments. While this already improves upon K-Means by increasing the amount of information gained, it still requires a suitable value for the number of distributions, analogous to the K in K-Means clustering. This is, at least in this case, an unintuitive parameter that essentially has to be guessed and evaluated in order to choose a suitable value. Luckily it can be eliminated by using an infinite mixture model. As the name implies, an infinite mixture model is similar to a Gaussian mixture model, except using an infinite amount of distributions, thereby removing the most significant parameter.

The core component of this infinite mixture model is a *Dirichlet process*. This is conceptually similar to the well-known Dirichlet distribution, with one key difference. Whereas the Dirichlet distribution generates discrete probability distributions with a finite amount of possible values, the Dirichlet process generates distributions with an infinite amount of possible values. In a Dirichlet process, the proportions of how many samples are assigned to each cluster are generated using a *stick-breaking process* (alternatively, with a *Pólya urn scheme* or a *Chinese restaurant process*)[12]. In the stick-breaking process, the probability w_k of a sample being assigned to the k'th distribution is given by

$$w_k = \beta_k \cdot \prod_{i=1}^{k-1} (1 - \beta_i)\,, \tag{1}$$

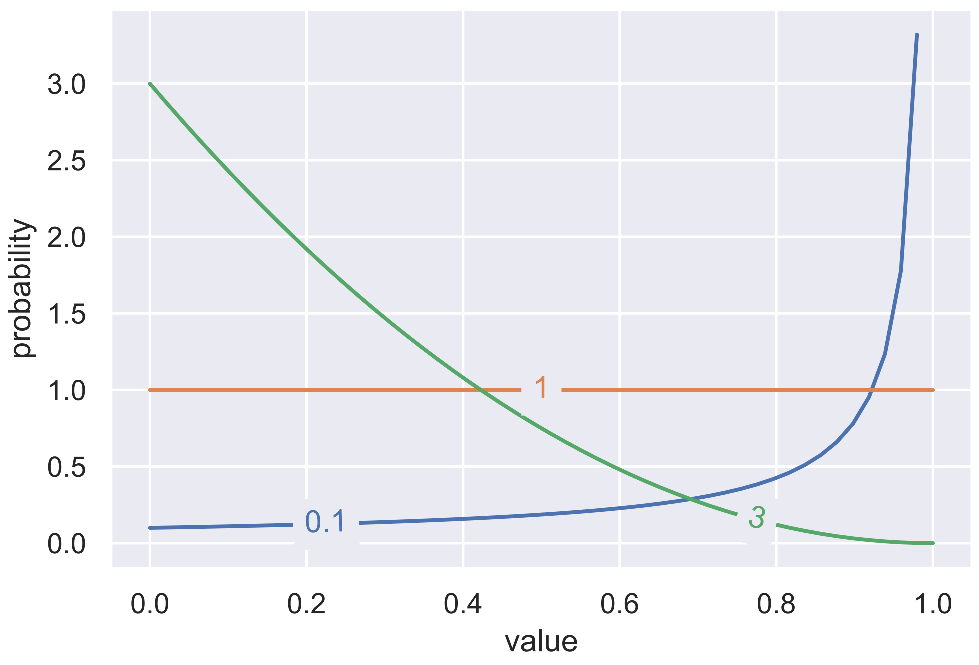


Figure 8 – The Beta$(1, \alpha)$ distribution for various values of α. As the value of α increases, the probability mass of the distribution shifts towards zero.

where β_k is a random variable drawn from the distribution Beta$(1, \alpha)$. Since the Beta distribution is defined on the interval $[0, 1]$, the β_k values can be considered as portions of a unit-length stick. When the first cluster is assigned to, a piece of size β_k is broken off of this unit-length stick. On subsequent assignments, the new sample is either assigned to an existing cluster with a probability proportional to the length of that cluster's portion of the stick, or a new cluster is assigned to. In the latter case, the new cluster gets a β_k-sized portion of the remains of the stick. This way, cluster assignments tend towards a long-tailed distribution. The α parameter in the prior distribution Beta$(1, \alpha)$ is called the weight- concentration parameter. As shown in fig. 8, the probability mass of the distribution is inversely proportional to the value of α. For lower values of α, big portions of the stick are likely to be broken off, concentrating the cluster assignments into different clusters; conversely, higher values of α lead to smaller portions and more diversity in the cluster assignments.

The tendency to continuously decrease the probability of assigning to a new cluster is a key factor in using Dirichlet process clustering in practice. Since at a certain point the probability of a new cluster being assigned becomes practically zero, it is sufficient to implement this using a finite "large enough" number of clusters, after which the clusters with very low probabilities can be pruned. This is illustrated with a simple example in fig. 9. Note that because of the Bayesian nature of the algorithm, even though the prior favors a long-tail distribution, it has no problem creating equally likely clusters if the data demands it.

The samples generated from each clusters are assumed to come from a Gaussian distribution; the rest of the probablistic model mostly consists of run-of- the-mill priors, without much semantic meaning and differing between different implementations of the algorithm. In this case an implementation from Scikit- Learn[13] was used, whose probabilistic model and subsequent derivation of the inference algorithm can be found online[5].

4.2 Supervised

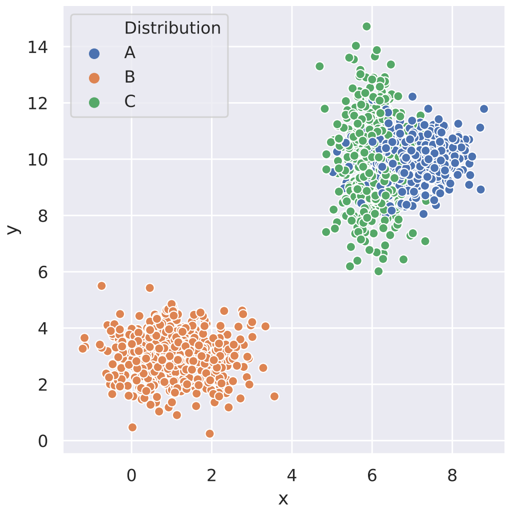
After the data is augmented by the previously described clustering algorithms, it's fed into a convolutional neural network for classification. Since the documents have a dual column layout (meaning each line of text is pretty short) and classification is based on the lines from the document, a sliding window is used to supply more context. The window consists of a variable amount of lines before and after the main line, which is the one supplying the label (whether or not the line starts a new speech). This sliding window is then used as input to a neural network, consisting of a convolutional part followed by a fully-connected neural network part. After the convolutional part is applied to the sliding window, the output is concatenated to the clustering distribution obtained previously (in the case of the mixture model, this is the actual probability distribution over each cluster; in the K-Means case, it is a one-hot vector). There are two different models, differing in their convolutional layer:

- WordCNN: A shallow word-based architecture[3], where the text is tokenized such that each token is either a word or a single punctuation mark (for example, "Hello-!" would produce the tokens "Hello", "-" and "!").
- CharCNN: A character-based architecture[8]. This is a deeper network, that operates on the raw characters without using an embedding layer.

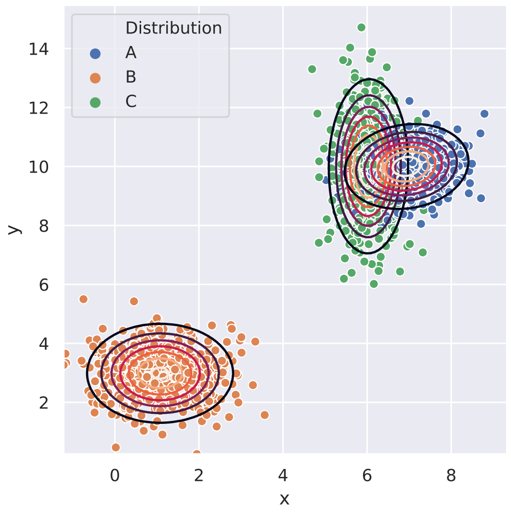
The WordCNN and CharCNN models are described in tables 1a and 1b ; the fully-connected neural network common to both models in described in table 1c. A high-level overview of how the different parts of the system fit together is shown in fig. 5.

The network is trained for 100 epochs, stopping early once no significant improvement has been made on a small validation set for 10 epochs in a row.

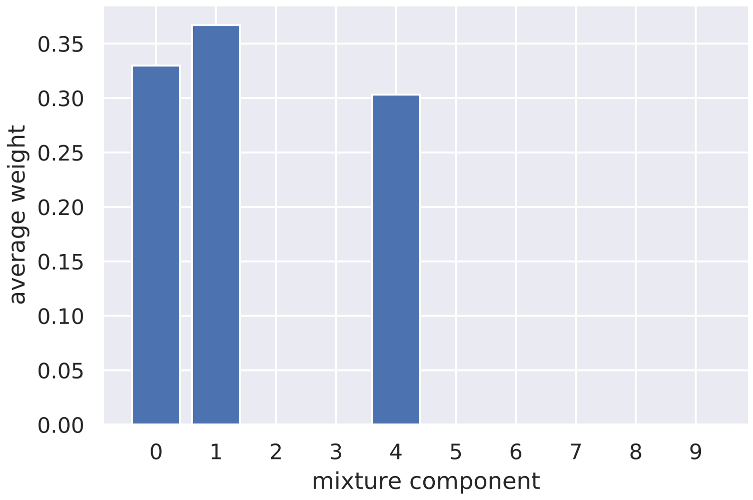
[5]http ://scikit-learn.org/stable/modules/dp-derivation.html



(a)



(b)



(c)

Figure 9 – Dirichlet Process mixture model clustering with a prior of 10 mixture components, performed on 1500 samples drawn from three different bivariate normal distributions (fig. 9a). The distributions found by the clustering algorithm are drawn in fig. 9b. As shown by averaging the component weights of each sample's assignment (fig. 9c), the Dirichlet Process rightfully assigned to just three mixture components, despite its prior of ten.

Layer	Type	Filters	Size	Pooling
1	Embedding	-	100	-
2	Conv	99	$3, 5, 7$	1-max

(a) The convolutional part of the WordCNN models consists of an embedding layer to embed the tokens into a higher-dimensional space, followed by a single convolutional layer. The convolutional layer uses 33 filters of each of three different sizes, with a stride of 1. The filters are concatenated for a total of 99 filters, with 1-max pooling applied to each filter such that the final output is a 99-dimensional vector.

Layer	Filters	Size	Pooling
1	256	7	3
2	256	7	3
3	256	3	-
4	256	3	-
5	256	3	-
6	256	3	3

(b) The convolutional part of the CharCNN model is a relatively deeply layered sequence of convolutions. Each convolution is followed by a ReLU activation function. The convolutions have a stride of 1, while the pooling layers are non-overlapping with a stride of 3.

Layer	Output Size	Activation	Dropout
1	1024	ReLU	Yes
2	1024	ReLU	Yes
3	1	Sigmoid	No

(c) Both architectures go through this fully connected neural network with 3 layers. For the layers using dropout, a rate of 0.5 is used.

Table 1 – The layouts of the neural networks. Table (a) corresponds to the convolutional part of the WordCNN model while Table (b) similarly corresponds to the CharCNN model; Table (c) describes the fully connected neural network common to both models.

The optimisation process is done using the Adadelta[14] algorithm[6], with binary cross-entropy as the loss function and the training data delivered in batches. Regularisation is done through a combination of dropout[16] and an upper limit on the L2 norm of each weight vector[17]. Further explanations of the optimisation process and the regularisation methods are provided in section 4.2.1 and section 4.2.2 respectively.

[6]Why not Adam[15]? The architecture[3] and survey paper[7] that this work is based on both date to roughly the year when the Adam paper was published, and thus likely predate the widespread acceptance of Adam as a de facto standard. Since a full analysis of different optimizers is outside the scope of this thesis and some quick informal tests showed no difference between using Adadelta or Adam, there is no reason to deviate from the survey.

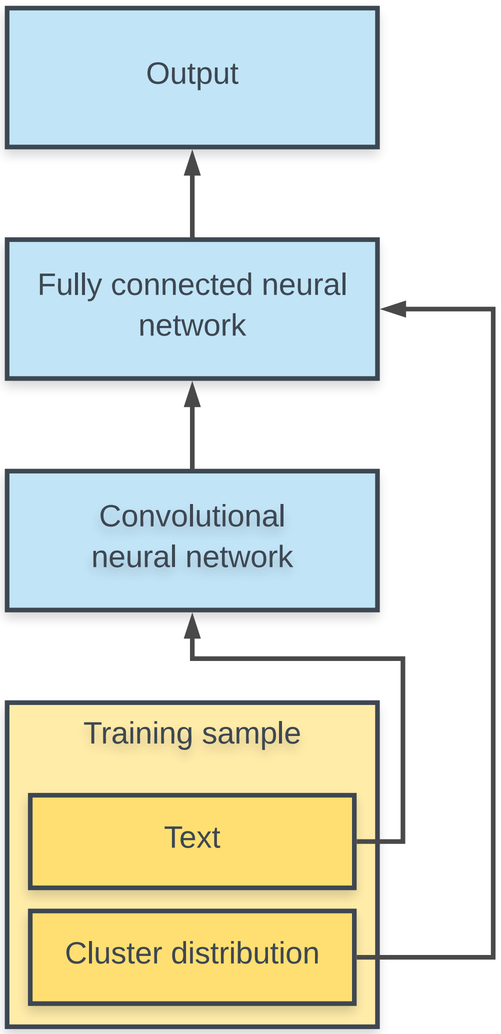


Figure 10 – The layout of the supervised portion of the system. The input data contains text and a cluster type assigned by the unsupervised portion. The text gets put into a convolutional neural network, the output of which is fed together with the cluster type into a fully connected neural network. The sigmoid function is applied to the output of this final neural network to obtain the classification.

4.2.1 Optimisation process

Optimisation is done using the Adadelta update rule[14]. This is easiest to explain by starting with the basic mini-batch stochastic gradient descent algorithm (SGD). At each iteration t, the network parameters θ are updated based on some calculated difference:

$$\theta_{t+1} = \theta_t - \Delta\theta_t\,. \tag{2}$$

The difference between optimization algorithms is how $\Delta\theta$ is calculated. With SGD, it is simply

$$\Delta\theta_t = \mu\nabla_{\theta_t}\mathcal{L}(\theta_t)\,, \tag{3}$$

where $\mathcal{L}$ is the loss function (in this case, the binary cross entropy between the predicted labels and the true labels), and μ is an arbitrary learning rate between 0 and 1. This learning rate is a tricky parameter to set; too low and learning will take ages, too high and the network will fail to converge because the steps taken are too big. One way to improve this is by using the Adagrad[18] algorithm. While SGD uses a single learning rate for the entire parameter vector, Adagrad (which stands for "adaptive gradient") adapts the learning rate for each individual parameter; frequently updated parameters get a lower learning rate, while less frequently updated parameters are updated with a higher learning rate. This is done by simply dividing the learning rate with the L2 norm of the sum of all previous gradients:

$$g_t = \nabla_{\theta_t}\mathcal{L}(\theta_t) \tag{4}$$

$$\Delta\theta_t = \frac{\mu}{\sqrt{\sum_{\mathcal{T}=1}^{t} g_{\mathcal{T}}^2}}\,g_t\,. \tag{5}$$

This has been found to work very well, in particular in natural language processing and computer vision where features are often sparse, but it has two drawbacks:

1. A suitable value for the global learning rate μ has to be manually provided.
2. Since the learning rate is rescaled using a monotonically increasing sum of previous gradient magnitudes, the learning rate will converge to zero.

Adadelta nullifies these drawbacks by eliminating the global learning rate and restricting the accumulation of gradients to a window of recent updates. First of all, the sum over all previous gradients is replaced by an exponentially decaying average of the squared gradients, referred to as $E[g^2]$:

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2\,. \tag{6}$$

Here ρ is a constant representing the rate of decay; this decaying sum serves as a more efficient approximation of an actual window of past gradients, which would require far more memory to store (considering both the huge number of parameters in a neural network and the memory constraints of running on a GPU). Substituting this into the Adagrad algorithm gives us:

$$\begin{aligned}\Delta\theta_t &= \frac{\mu}{\sqrt{E[g^2]_t}} g_t &(7)\\ &= \frac{\mu}{RMS[g]_t} g_t\,. &(8)\end{aligned}$$

The quantity $\sqrt{E[g^2]_t}$ is called the *root mean square* of g, which occurs often enough in optimisation algorithms that it is often abbreviated as $RMS[g]$. As a final step, the learning rate μ is eliminated by replacing it with a decaying average of the previous gradient updates, similar to eq. (6):

$$\Delta\theta_t = \frac{RMS[\theta]_{t-1}}{RMS[g]_t} g_t\,. \tag{9}$$

4.2.2 Regularisation

Dropout[16] is a Regularisation method that is rather crude at first glance; on each batch update, each node in the neural network has a probability p of outputting zero (i.e. the node being "disabled"). As a result, nodes cannot rely on the output of any other node being present, preventing co-adaptation and as a result reducing the probability of overfitting on the training data. Dropout is only active during training; when running the network in evaluation mode, all nodes are active and the outputs are rescaled by a factor of $1-p$ to account for the now higher activation values. There is another way to view dropout. It is commonly known that neural network (or really any machine learning classifier) performance can be improved by training a large number of them and averaging their outputs. Since every combination of nodes disabled by dropout could be considered a unique neural network, dropout acts as computationally cheap approximation to averaging multiple networks.

In addition to dropout, a maximum L2 norm is imposed on each node's incoming weight vector. If the weight vector's L2 norm exceeds this limit, it is rescaled so that its new norm is equal to the maximum allowed norm; otherwise the weight vector is left alone:

$$\mathbf{w} = \begin{cases} \mathbf{w} & \text{if } ||\mathbf{w}||^2 \leq n \\ n\frac{\mathbf{w}}{||\mathbf{w}||^2} & \text{otherwise} \end{cases} \tag{10}$$

This differs from the more common L2-regularisation — where the combined L2 norm of all weight vectors is added to the loss function — in that the weights are not being continuously pushed towards zero, making it a milder form of regularisation that allows for a bit more complexity in the model.

4.2.3 Convolutional Neural Networks

In addition to explaining how a convolutional neural network works, this section will additionally go into why the choice was made to use a convolution neural network, rather than a recurrent neural network (which is more strongly associated with text processing) or any other machine learning algorithm. Starting with the basics, text is inherently troublesome because it produces 3-dimensional data: there is a feature vector for each word, and some (either variable or predetermined through padding) amount of words per text. This makes each training sample a 2-dimensional matrix, which then gets stacked in the depth dimension to produce 3-dimensional training data. This clashes with the tendency of usual machine learning algorithms to work on 2-dimensional data, assuming a feature vector for each sample rather than a matrix. There are three common methods to deal with this:

- Bag of words
- Convolutional neural networks
- Recurrent neural networks

Aside from being completely different methods, they differ in a major way in how they handle the sequential nature of text (i.e. the words in a sentence have a meaningful order). The bag of words approach is the simplest in that it simply disregards this sequential nature entirely, instead creating what is essentially a histogram of word occurrences. This downsamples each sample from a feature matrix to a feature vector, allowing the use of conventional machine learning algorithms (most commonly support vector machines). While the sequential information can be kept to some degree by using histograms of n-grams rather than words (unigrams), this causes the size of the input data to scale exponentially with the value of n.

Convolutional neural networks (CNNs) work by taking a number of filters (sometimes called kernels or feature maps) of a specified size and convolving these over the input data. A simplified example using one filter is shown in fig. 11. In this example, the input text is convolved with a filter with a width of 3 and a stride of 1 — that is, each application of the filter considers three subsequent input elements, after which the window is shifted one space to the right. This filter is essentially a small neural network mapping three items to one output value, whose weights are reused for each application of the filter. Reusing the weights in this way (weight sharing) prevents the number of parameters in the network from spiraling out of control. [19] After the application of this convolution layer, the responses of the filter form a new sequence of roughly the same size as the input (minus a few due to missing the edges). The next step is to downsample this sequence by means of a *max pooling* layer, which maps all values within a window to the maximum value amongst those values. While conceptually similar to a convolution, this step

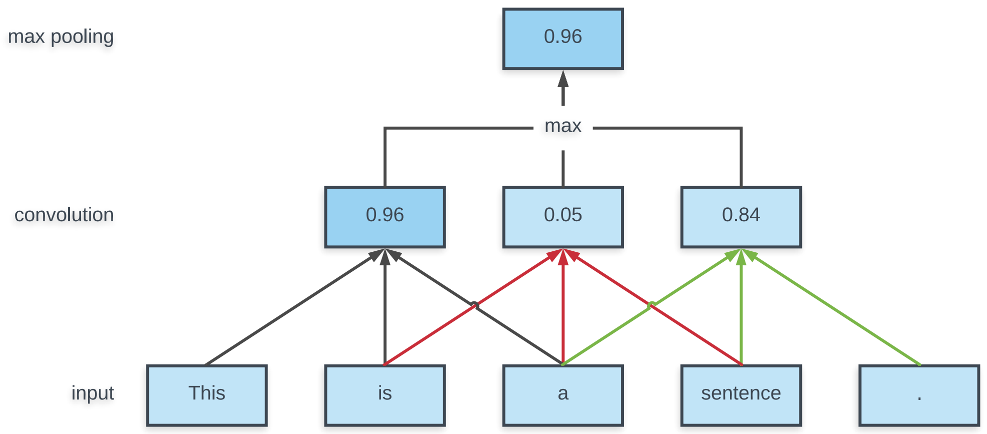


Figure 11 – A simplified convolutional neural network with one filter and max pooling.

generally does not involve overlap, instead either setting the stride to the same value as the window size (usually 2) or reducing the entire sequence to 1 value (1-max pooling). The reason for this is twofold:

1. It downsamples the data, reducing the amount of parameters required further on in the network.
2. It adds translation invariance to the feature detected by this filter. The example filter of fig. 11 reacts strongly to the first three words. Without the pooling layer, changing the location of these words in the input would similarly change the location of the high activation in the intermediate representation; this would be translation *equivariance*. The more aggressively the pooling is applied, the higher the degree of invariance (with full translational invariance being achieved with 1-max pooling).

This combination of convolution followed by pooling can be repeated multiple times as desired or until there is only a single value left as output from the filter. Finally, the outputs of all filters are concatenated and fed into a regular neural network.

While CNN architectures in computer vision are generally very deep, they tend to be very shallow in natural language processing; commonly just a single convolution followed by 1-max pooling [7].

4.2.4 Difference between convolutional and recurrent neural networks

Recurrent neural networks (in particular LSTMs or GRUs) are seemingly the most natural fit for language processing, since they process an entire sequence and are therefore fully conditioned on the word order (as opposed to the convolutional neural networks which tend to learn translation invariant ngram features). In spite of this, the decision to use convolutional neural networks is based on two factors:

1. In practice, the performance for classification tasks does not differ between the convolutional and recurrent neural networks.[20]
2. The computations in convolutional networks are highly independent of each other, allowing for great parallelization (in particular with regards to running on a GPU). In contrast, LSTMs are bottlenecked by the fact that each calculation is dependent on the previous calculations. As a result, CNNs achieve far higher training speeds.[21]

5 Experimental Setup

Experiments are focused on the difference in performance between the baseline CNN model without clustering information (referred to from here on as `CNN`) and the model augemented with clustering information (which will be reffered to as `CNN-cluster`). Performance will be measured with regards to the following three metrics:

1. Number of training epochs until convergence
2. The F1 score or average precision metrics on a test set (see Section 5.2)
3. The number of training samples required to attain a specific F1/average precision score

In each case, the experiment will be repeated 10 times by means of 10-fold cross validation followed by a Student's T-test to gauge the probability of the following null hypothesis being true:

Null Hypothesis 1 *Adding clustering information to the CNN model does not change the performance of the model.*

5.1 Dataset

Referring back to Figure **??**, the average document has between 100 and 300 positive samples. Since a secondary concern is to minimize the number of documents that would have to be annotated as training data, the tested dataset sizes will be very low, with the number of positive samples being one of 100, 200, 500 and 1000. Due to the relative abundance of negative samples and to prevent overfitting on the distribution of the labels, stratified sampling will be used to keep a 1:1 ratio of positive to negative samples. In addition to the size, the number of cluster types (the k in k-means) will be varied to examine its impact on the performance.

5.2 Testing performance

All models will be tested on a test set containing 1000 positive samples and 10000 negative samples, all of which are guaranteed not to be in the training set. Performance on this set is measured by constructing a precision-recall curve, and calculating two values:

1. The average precision (which is equivalent to the area under the curve)
2. The F1 score of the point on the curve maximizing the F1 score

5.3 CNN performance

Although less central to the thesis than the difference between the `CNN` and `CNN-cluster` models, some experimentation will be done with the parameters of the convolutional network in an attempt to optimize the performance. These parameters include the dimensionality of the word embeddings, the number of filters, the pooling strategy (1-max versus a smaller region) and the number of convolutional layers.

5.4 Generalisation

This particular dataset has the quirk that the performance of a rule-based system created based on recent documents decreases in performance when used on older documents, the older the document the worse it performs. This occurs despite the layout being visually the same all the way back to the 1950s. A number of files from old election periods has been labeled (and manually verified for correctness) in order to test

1. whether the CNN models handle this better than the rule-based system does.
2. Whether the clustering-augmented CNN model performs better on this task than the baseline CNN.

6 Evaluation

To recap the previous sections, we are dealing with two models here:

- "WordCNN", a convolutional neural network operating on tokenized words and punctuation,
- "CharCNN", a convolutional neural network operating on tokenized characters,

as well as three variations for each model:

- No clustering.
- The data is augmented with K-Means clustering.
- The data is augmented with clustering through an infinite mixture model.

These variations will be referred to as "Baseline", "K-Means" and "Infinite Mixture model" respectively. Although K-fold cross validation would seem to be a good way of comparing their relative performance, there are two properties of cross validation that make it less desirable in this case:

- One of the more interesting variables in this context is the size of the training set, but with cross validation it is directly tied to the number of iterations (e.g. with a dataset of 1000 samples and 10 folds, each fold would have a training set of 100 samples).
- The training set and the test set are sampled from the same pool of data. In our case, we want the training set and the test set to be sampled from different electoral periods, because:
  - A change of electoral period means at least a partial change in parliamentary members, which reduces the ability of the model to overfit on the names of members.
  - Although older documents use the same layout as newer ones, the PDF files were generated using different software (electoral period 13 started in 1994 and was the first period to use truly digital documents — older documents are scanned versions of printed paper). Since PDF decompilation is already a flaky process, this results in the PDF decompiler making different mistakes than it does on the newer documents. This is a big problem for rule-based systems, but hopefully the probabilistic nature of neural networks makes them more robust to this issue.

Therefore, a variation of cross validation is used instead. There is a pool of training data consisting of documents from the 18th electoral period of the *Bundestag*, and a test set containing documents from the electoral periods 13 to 17. For 5 iterations, k samples are pulled from the pool of training data. Each of the three models is trained on these samples and then tested on the full test set.

6.1 Parameter Exploration

Before directly comparing the two models, a good baseline value has to be determined for the two parameters in the model that are the most difficult to determine:

1. the size of the sliding window applied to the lines of text in the input data
2. the number of clusters to detect in the blocks of text (i.e. the "k" in k-means)

Although there are more parameters, they are related to the neural network, meaning there is a large amount of prior knowledge available to set the parameters.

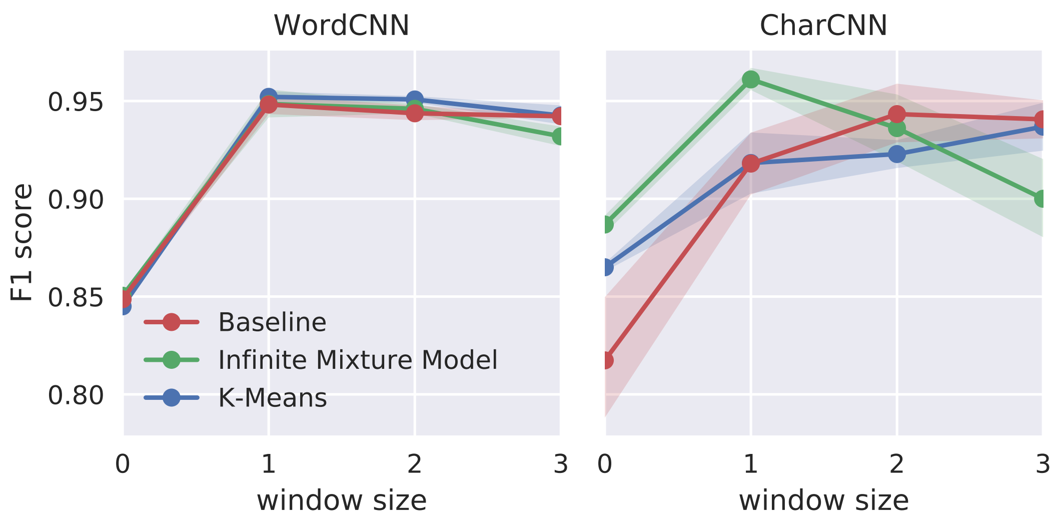
Plot opnieuw genereren met een logischere unit op de x-as, en testen met window sizes die alleen naar voren kijken

The effect of the sliding window size is shown in fig. 12a. In this figure, the window size refers to how many neighbouring lines of text are considered: 0 means just the line of text by itself, 1 means the next line is included, 2 means both the next and previous lines are included, et cetera. For each model, there is a very sharp increase in performance when going from zero to one neighbouring element, followed by a gradual decline as more elements are added.

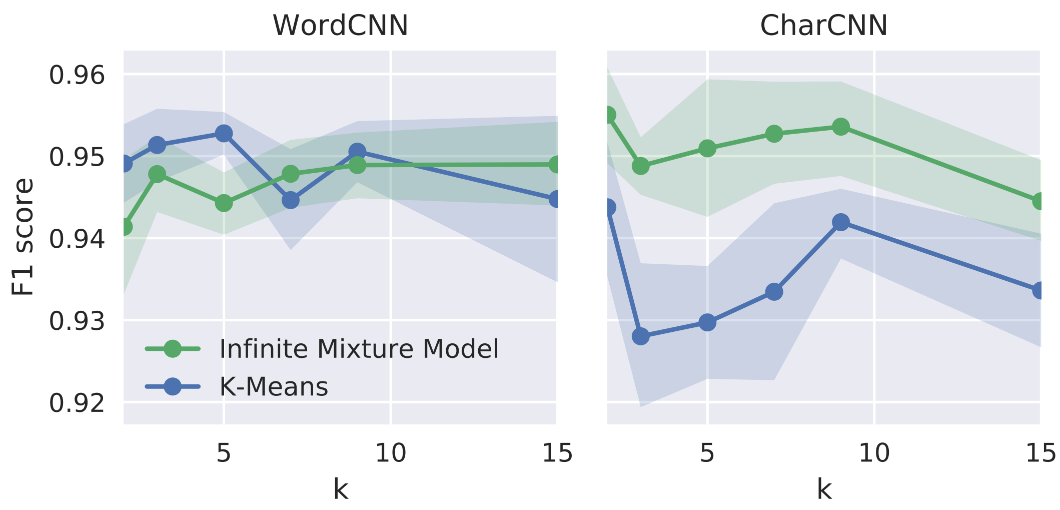
Results for the number of clusters are given in fig. 13. In this case, the window size was fixed to the previously determined optimum of 2. The baseline model is not tested here due to not using clustering, and the model using infinite mixture clustering is not tested due to its fixed amount of clusters. The result varies little, and given the large variances no value can really be said to outperform the rest.

6.2 Training Set Size

Using the ideal parameters obtained in section 6.1, the influence of the training set size is shown in fig. 14. This shows a number of interesting differences. First of all, the Baseline and K-Means models perform similarly for both CNN models at every input size, while the infinite mixture version outperforms them only when using the CharCNN model. Additionally, the WordCNN and CharCNN models respond much differently to variations of the training set's size; WordCNN maxes out its performance at roughly 800 samples, while CharCNN requires around 2000 samples to reach the same performance. However, when augmented using the infinite mixture model, CharCNN peaks at a slightly higher score than WordCNN.



(a) The F1 score with regards to the sliding window size for each model.



(b) The F1 score with regards to the number of clusters for the K-Means model.

Figure 12 – These figures show the performance with regards to the sliding window size and the number of clusters, using a training set of 2000 samples. In fig. 12a, the K-Means model was trained using 9 clusters. Each figure's F1 score is averaged over 5 trials; the translucent bands around the lines indicates the confidence intervals, meaning that based on the observed F1 scores and assuming normality, the true mean is 95% likely to fall within that interval. The dots on the lines indicate measurements.

Figure 13 – This figure shows the performance with regards to the number of cluster types for each model trained on 1200 training samples with a window size of 5. The vertical axis shows the F1 score, averaged over 10 trials; the horizontal axis shows the number of cluster types considered in the final clustering step. The translucent bands around the lines indicates the confidence intervals, meaning that based on the observed F1 scores and assuming normality, the true mean is 95% likely to fall within that interval. The dots on the lines indicate measurements.

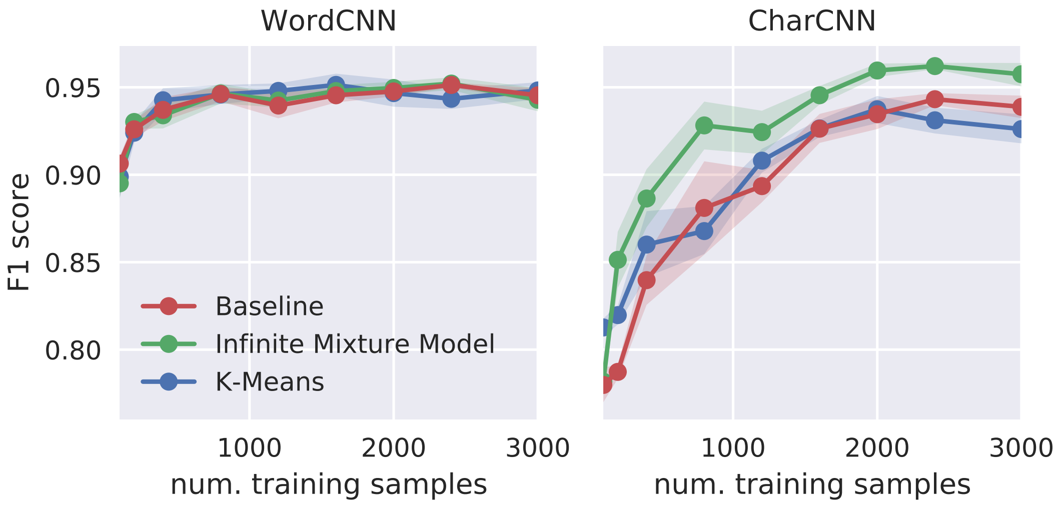


Figure 14 – This figure shows the performance with regards to the size of the training set, using a window size of 2 of 9 clusters for the K-Means models. The F1 scores are averaged over 5 trials; the translucent bands around the lines indicates the confidence intervals, meaning that based on the observed F1 scores and assuming normality, the true mean is 95% likely to fall within that interval. The dots on the lines indicate measurements.

7 Conclusion

Segmenting political proceedings through the use of machine learning is a viable approach, reaching peak F1 scores of 0.96. When training data is very sparse, word-level convolutional neural networks are the best option; augmenting them with information regarding the document layout, whether through K-Means clustering or an infinite mixture model, offers no notable increase in performance. When more training data (roughly 1000 positive samples) is available, character-based convolutional neural networks outperform their word-based brethren only when augmented using infinite mixture model clustering.

References

1. Iyyer, M., Enns, P., Boyd-Graber, J. & Resnik, P. *Political ideology detection using recursive neural networks* in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* **1** (2014), 1113–1122.
2. Gross, J. H., Acree, B., Sim, Y. & Smith, N. A. Testing the Etch-a-Sketch Hypothesis: A Computational Analysis of Mitt Romney's Ideological Makeover During the 2012 Primary vs. General Elections (2013).
3. Kim, Y. Convolutional Neural Networks for Sentence Classification. *CoRR* **abs/1408.5882.** arXiv: `1408.5882`. <`http://arxiv.org/abs/1408.5882`> (2014).
4. Klampfl, S., Granitzer, M., Jack, K. & Kern, R. Unsupervised document structure analysis of digital scientific articles. *International Journal on Digital Libraries* **14,** 83–99 (2014).
5. Ferrara, E., Meo, P. D., Fiumara, G. & Baumgartner, R. Web Data Extraction, Applications and Techniques: A Survey. *CoRR* **abs/1207.0246.** arXiv: `1207.0246`. <`http://arxiv.org/abs/1207.0246`> (2012).
6. Gogar, T., Hubacek, O. & Sedivy, J. *Deep Neural Networks for Web Page Information Extraction* in *Artificial Intelligence Applications and Innovations* (eds Iliadis, L. & Maglogiannis, I.) (Springer International Publishing, Cham, 2016), 154–163. ISBN: 978-3-319-44944-9.

7. Zhang, Y. & Wallace, B. C. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *CoRR* **abs/1510.03820.** arXiv: `1510.03820`. <`http://arxiv.org/abs/1510.03820`> (2015).

8. Zhang, X., Zhao, J. & LeCun, Y. *Character-level convolutional networks for text classification* in *Advances in neural information processing systems* (2015), 649–657.

9. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **abs/1512.03385.** arXiv: `1512.03385`. <`http://arxiv.org/abs/1512.03385`> (2015).

10. Conneau, A., Schwenk, H., Barrault, L. & LeCun, Y. Very Deep Convolutional Networks for Natural Language Processing. *CoRR* **abs/1606.01781.** arXiv: `1606.01781`. <`http://arxiv.org/abs/1606.01781`> (2016).

11. Sibson, R. SLINK: an optimally efficient algorithm for the single-link cluster method. *The computer journal* **16,** 30–34 (1973).

12. Frigyik, B. A., Kapila, A. & Gupta, M. R. Introduction to the Dirichlet distribution and related processes. *Department of Electrical Engineering, University of Washignton, UWEETR-2010-0006* (2010).

13. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).

14. Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *CoRR* **abs/1212.5701.** arXiv: `1212.5701`. <`http://arxiv.org/abs/1212.5701`> (2012).

15. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *CoRR* **abs/1412.6980.** arXiv: `1412.6980`. <`http://arxiv.org/abs/1412.6980`> (2014).

16. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* **15,** 1929–1958 (2014).

17. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* **abs/1207.0580.** arXiv: `1207.0580`. <`http://arxiv.org/abs/1207.0580`> (2012).

18. Duchi, J., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* **12,** 2121–2159 (2011).

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

20. Yin, W., Kann, K., Yu, M. & Schütze, H. Comparative Study of CNN and RNN for Natural Language Processing. *CoRR* **abs/1702.01923.** arXiv: `1702.01923`. <`http://arxiv.org/abs/1702.01923`> (2017).

21. Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y. N. Convolutional Sequence to Sequence Learning. *CoRR* **abs/1705.03122.** arXiv: `1705.03122`. <`http://arxiv.org/abs/1705.03122`> (2017).