UNIVERSITY OF AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Enriching Textual Data with Document Structure For Sentence Classification

by
Maarten de Jonge
10002109

June 17, 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

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 significantly improves the classification performance.

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

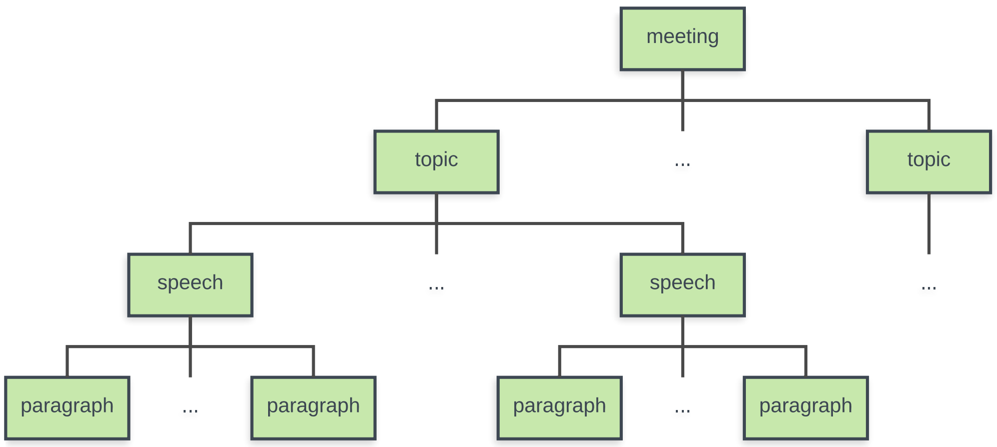


Figure 1 – The structure of a parliamentary debate is that of a shallow tree.

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.

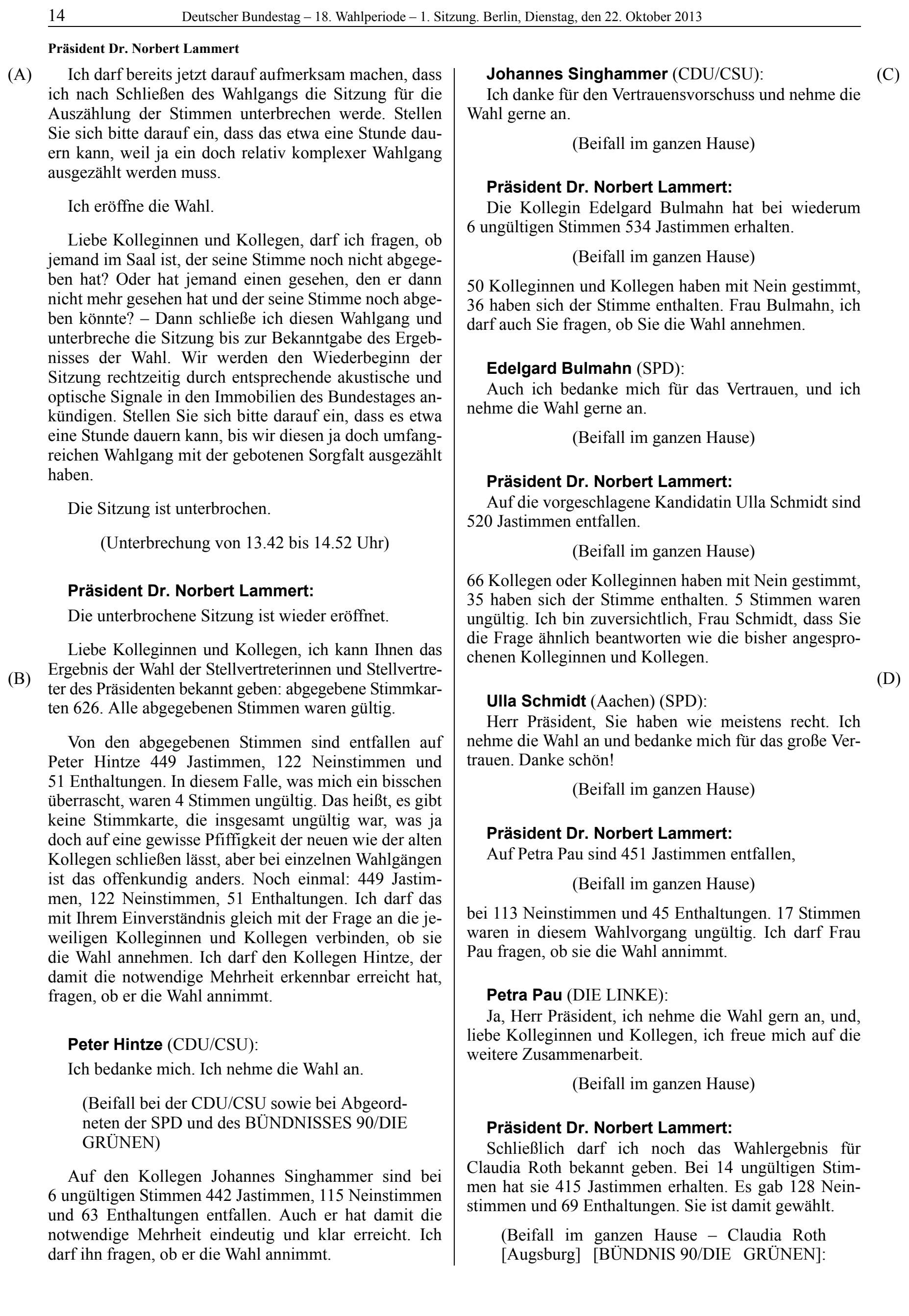
I propose that by using a small number of manually annotated documents as a dataset, a machine learning algorithm can learn to classify fragments of text in a way that allows it to segment a document into its constituent parts, while being more robust to noise than its rule-based counterpart. While the 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 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 this process will be supplied in section 4.


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.

(C) **Johannes Singhammer** (CDU/CSU):
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.

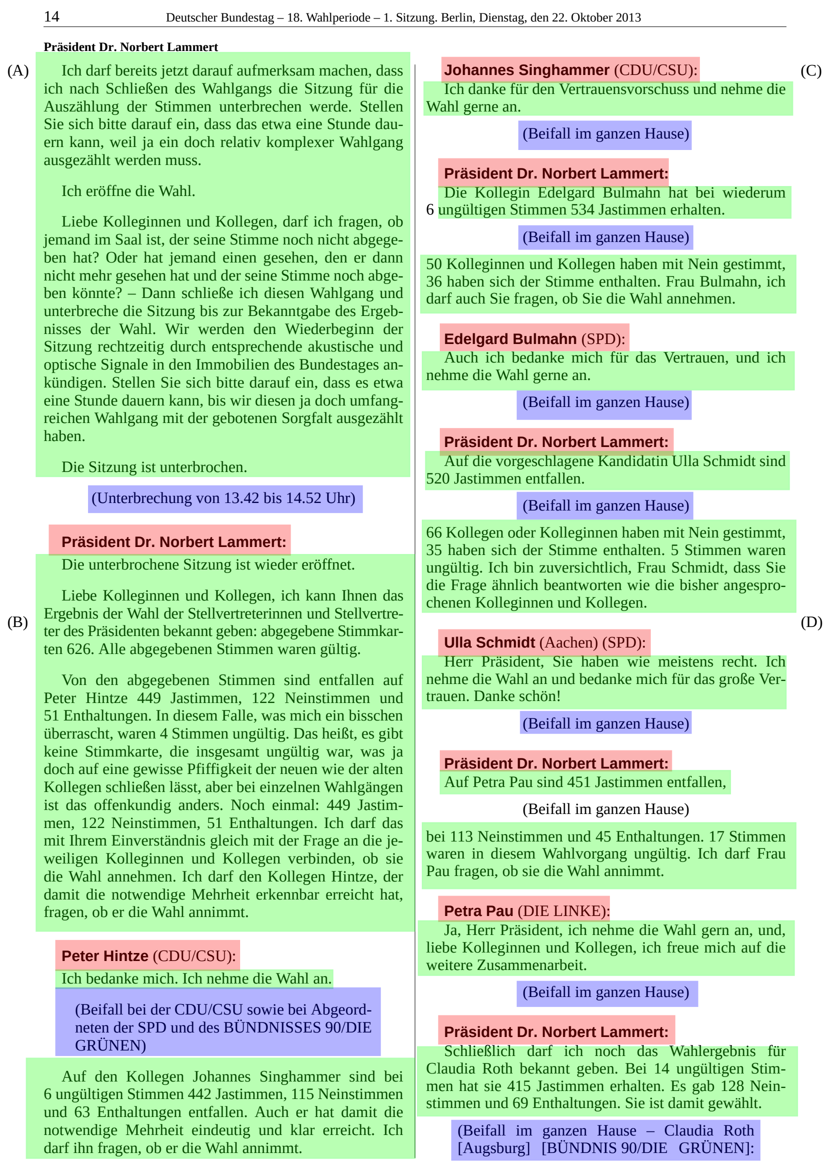
Petra Pau (DIE LINKE):
Ja, Herr Präsident, ich nehme die Wahl gern an, und, liebe Kolleginnen und Kollegen, ich freue mich auf die weitere Zusammenarbeit.

(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.

Ulla Schmidt (Aachen) (SPD): (D)

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.

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.

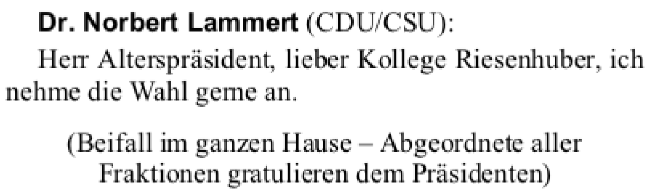
2.1 Dataset

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.* [4], 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.* [5]. 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.* [6] 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 words are tokenized and embedded 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 intends 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.* [6], 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 would be paragraphs, section headers, page decoration, etc. 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 following subsections will go into details on the two clustering steps.

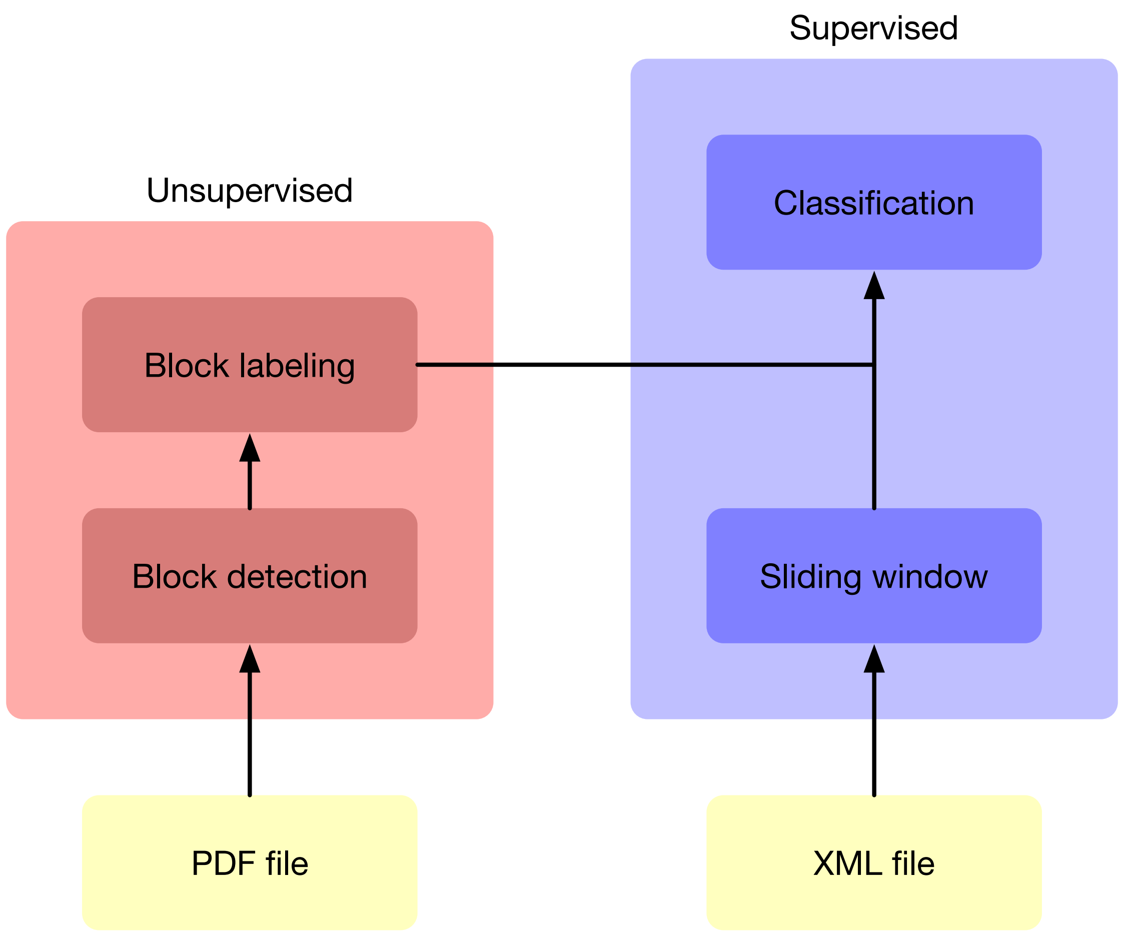


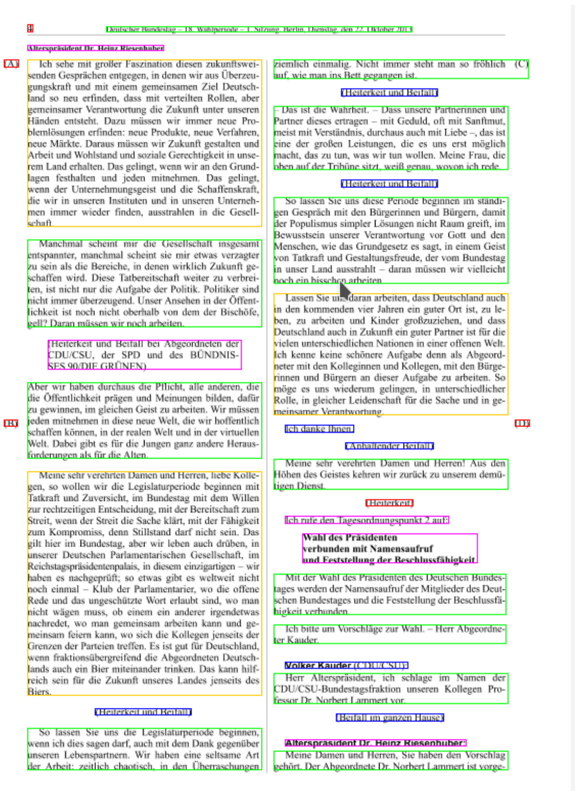
Figure 5 – A high-level overview of the system. The unsupervised block augments the input to the classifier.

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:

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

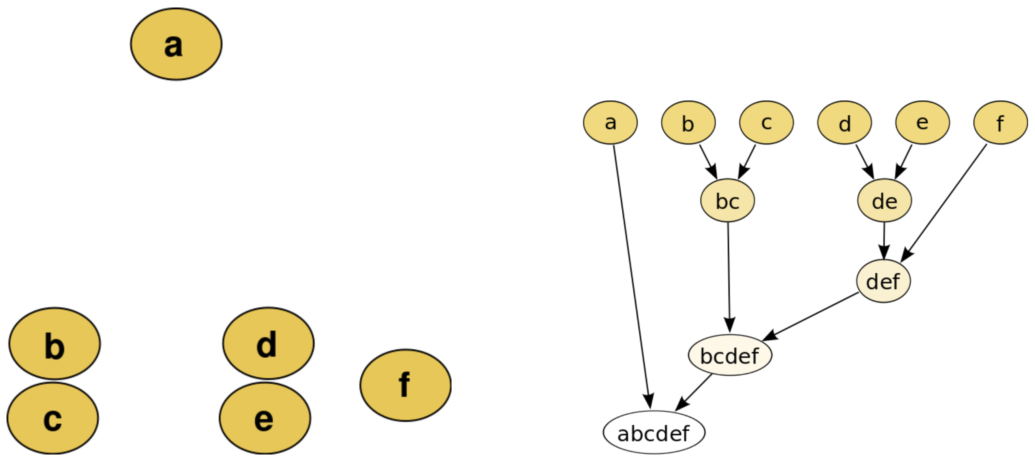
- Maximum-linkage: The distance between clusters is based on the furthest two elements:

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

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

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

As per Klampfl *et al.* [6], single-linkage clustering performs best for this task due to its tendency to form long thin clusters. This is beneficial since



(a) Before **(b)** After

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

text is somewhat long and thin in nature (especially words and sentences). As an additional bonus, while the general time complexity for HAC is in $\mathcal{O}(n^3)$, single-linkage clustering can be done in $\mathcal{O}(n^2)$ [11], making it far more usable on realistic 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 determined by trial and error based on the particular set of documents.

4.1.2 K-Means

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

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

This is done using K-means clustering, with the value of k being varied for experimental purposes.

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 source PDFs have a dual column layout with rather short lines, a sliding window is used to add valuable context, with the center element in the window supplying the label (i.e. does or does not start a speech). The text content of this window is then entered into a standard convolutional neural network architecture similar to the one proposed by Kim [3]. First an embedding layer is used to learn a high-dimensional representation of the words (no pre-trained embeddings were used because of both the specialized political domain of the data as well as a lack of quality pre-trained German models), followed by a number of convolutional filters with 1-max pooling. The output of the filters is then concatenated into a single feature vector, which is combined with the clustering data in one of three ways, which will all be evaluated:

1. The cluster type of the center element of the window is added to the feature vector.
2. The cluster types of each element in the window are added to the feature vector.
3. An additional smaller CNN is applied to the cluster types of each element in the window, the output of which is concatenated with the feature vector.

The resulting feature vector is then fed into a standard fully-connected neural network. This network features one hidden layer with a ReLU activation ($\text{relu}(x) = \max(0, x)$) and a single output node with a sigmoid activation. The layout of this system is detailed in fig. 8 along with its parameters and their baseline value.

The network is trained for 100 epochs, stopping early once no significant improvement in training loss has been made for 10 epochs in a row. The optimisation process is done using the Adadelta[12] algorithm[5], with binary cross-entropy as the loss function and the training data delivered in batches. The loss might exhibit some up and down fluctuations after a minimum has been reached; to account for this, the best loss so far along with the corresponding model parameters is kept track of throughout the learning process. Once all the epochs have been completed, the output is the model instantiated with the parameters corresponding to the lowest loss that was

[5]Why not Adam[13]? 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.

obtained. Regularisation is done through a combination of dropout[14] and an upper limit on the L2 norm of each weight vector[15].

4.2.1 Optimisation process

Optimisation is done using the Adadelta update rule[12]. 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 calculation:

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

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

$$\Delta\theta_t = \mu \nabla_{\theta_t} \mathcal{L}(\theta_t) \tag{2}$$

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[16] 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:

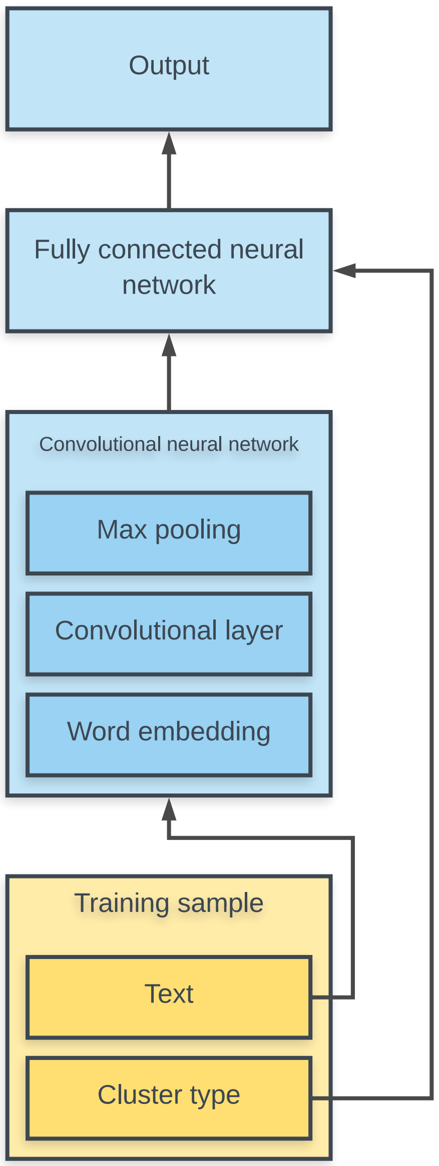
$$g_t = \nabla_{\theta_t} \mathcal{L}(\theta_t) \tag{3}$$

$$\Delta\theta_t = \frac{\mu}{\sqrt{\sum_{\mathcal{T}=1}^{t} g_{\mathcal{T}}^2}} g_t \tag{4}$$

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.



(a) 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.

Parameter	Default value
Window size	9
Word embedding size	300
Number of filters	100
Filter sizes	3, 5, 7
Activation	ReLU
Pooling type	1-max

(b) The default parameters used in the convolutional neural network.

Parameter	Default value
Number of hidden layers	1
Hidden layer size	50
Number of outputs	1
Output activation	Sigmoid

(c) The default parameters used in the fully-connected neural network.

Parameter	Default value
Batch size	50
Dropout rate	0.5
Learning rate	1.0
ρ	0.9
Maximum L2 norm	3

(d) The default parameters used for the optimisation process.

Figure 8 – The model and its parameters.

First of all, the sum over all previous gradients is replaced by an exponentially decaying average of the squared gradients, referred to as $E[g^2]$:

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho)g_t^2 \tag{5}$$

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:

$$\Delta\theta_t = \frac{\mu}{\sqrt{E[g^2]_t}} g_t \tag{6}$$

$$= \frac{\mu}{RMS[g]_t} g_t \tag{7}$$

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. (5).

$$\Delta\theta_t = \frac{RMS[\theta]_{t-1}}{RMS[g]_t} g_t \tag{8}$$

4.2.2 Regularisation

Dropout[14] 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:

$$\mathbf{w} = \begin{cases} \mathbf{w} & \text{if } ||\mathbf{w}||^2 \leq n \\ n\frac{\mathbf{w}}{||\mathbf{w}||^2} & \text{otherwise} \end{cases} \tag{9}$$

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

This is kind of jumbled and out of place, might need to be deleted or moved to an appendix or something.

For the purposes of machine learning, text 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 is troublesome as the standard machine learning algorithms 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. The bag of words approach is the simplest in that it simply disregards this sequential nature, 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 normal machine learning algorithms (commonly support vector machines). While the sequential information can be kept to some degree by use 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. 9. 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. [17] 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

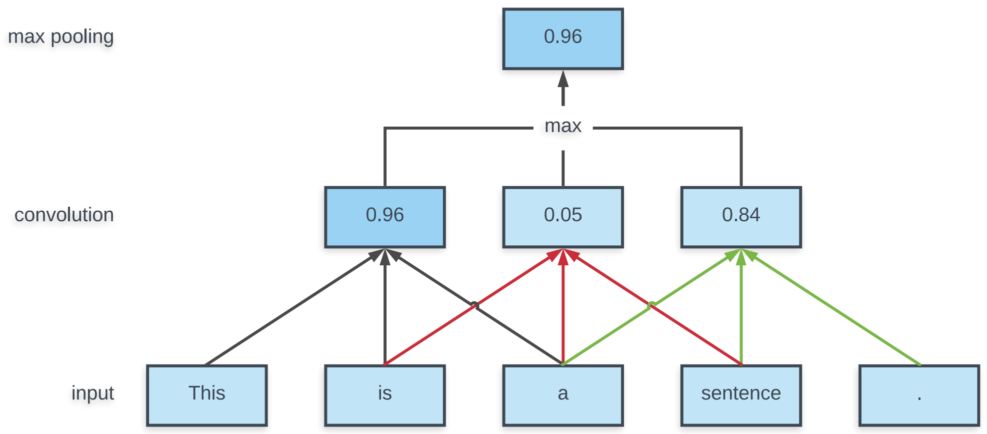


Figure 9 – A simplified convolutional neural network with one filter and max pooling.

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

1. It downsamples the number of inputs, 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. 9 appears to react strongly to the presence of the word “Hallo”. Without the pooling layer, changing the location of the word “Hallo” in the input would similarly change the location of the high activation in the intermediate representation; this would be *equivariance*. The more aggressively the pooling is applied, the higher the degree of invariance.

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 standard feedforward 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). Regardless, convolutional neural networks will be used in this research. This choice is based on two factors:

1. In practise, the performance for classification tasks does not differ between the two types of networks.[18]
2. The computations in convolutional networks are highly independent of each other, allowing for great paralellization (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.[19]

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

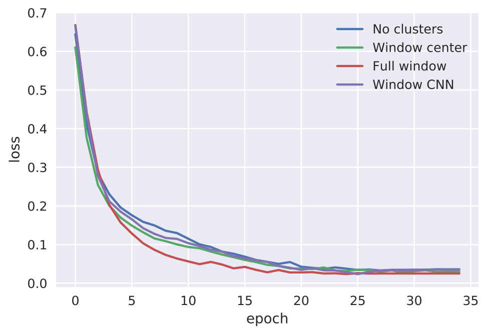
5.3 CNN performance

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.

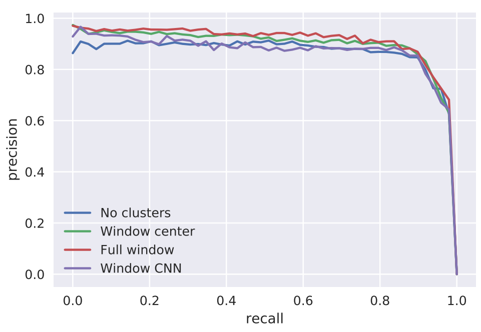
5.4 Generalisation

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.



(a) The average loss at each epoch.



(b) The average precision-recall curves for the various models.

Figure 10 – The loss over time (a) and the precision-recall curve (b). In both cases, the values are averaged over 10 trials.

6 Evaluation

6.1 Models

Here the models described earlier (TODO: beschrijven en ernaar verwijzen) are tested with the default parameters of the CNN. Based on the previous results, the dropout rate was set to 0.5 and the decay to 1×10^{-4}. The plots in Figure 10 show a subtle but observable difference; the model using the full window jumps out of the pack in both cases, converging a little bit faster and being on top for most of the precision-recall curve, although the model with an additional CNN catches up at higher recalls.

6.2 Regularisation

Since regularisation is very important to prevent overfitting, the common methods of regularisation are tested first to obtain a good baseline value for the rest of the experiments. These methods are:

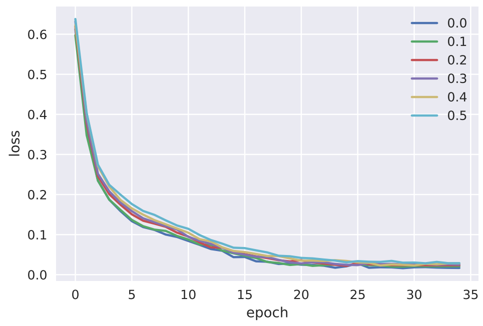
- Dropout, with values taken from the set $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.
- Weight decay, with values taken from the set

$$\left\{0.0, 1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}\right\}.$$

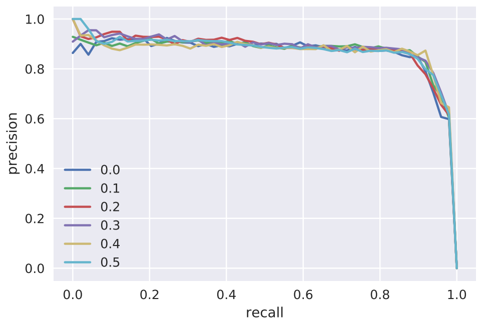
Further details on these methods are given in section 4.

6.2.1 Dropout

Figure 11 shows the training loss over time, as well as the resulting precision-recall curve on the training set. There does not seem to be any visible



(a) The average loss at each epoch.



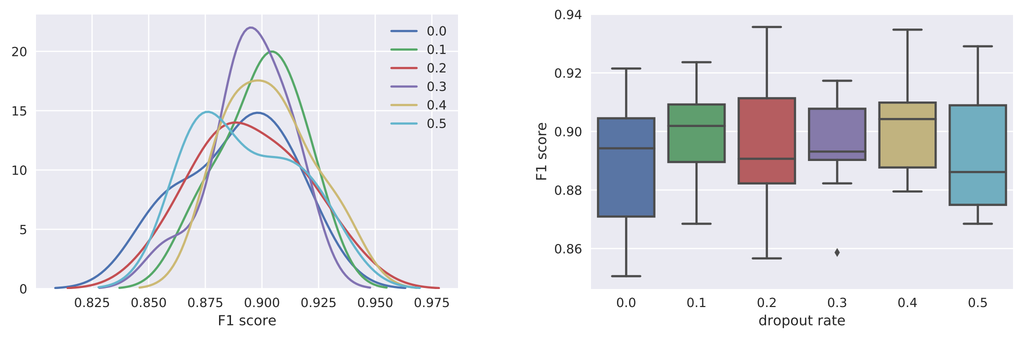
(b) The average precision-recall curves at various dropout values.

Figure 11 – The loss over time (a) and the precision-recall curve (b). In both cases, the values are averaged over 10 trials.

dropout rate	F1 mean	F1 stddev	AoC mean	AoC std	Area under averaged curve
0	0.888825	0.0220865	0.787263	0.0265971	0.866336
0.1	0.899671	0.0165874	0.817704	0.0531906	0.872387
0.2	0.89483	0.0224235	0.824571	0.0355852	0.877872
0.3	0.894933	0.0160834	0.832363	0.0457619	0.881924
0.4	0.903226	0.0177141	0.812464	0.0318788	0.869334
0.5	0.892828	0.0214581	0.835072	0.0545053	0.874904

Table 1 – The F1 and AoC scores at various dropout values.

difference in performance; this lack of effect is also shown in fig. 12. A dropout rate of 0.5 appears to score the best, although none of the distributions are different in a statistically significant way. The full data, with the mean and standard deviation of the F1 score and the area under the curve, are given in table 1.



(a) Kernel density estimation **(b)** Boxplot

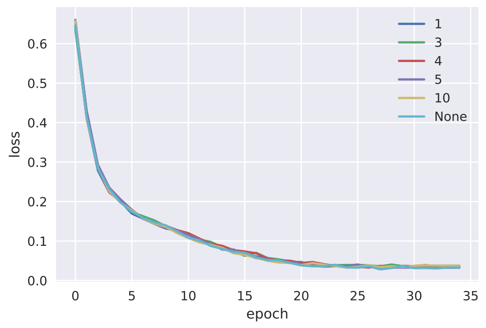
Figure 12 – A kernel density estimation and boxplot, based on the F1 score values over 10 repeated trials.

6.2.2 Weight decay

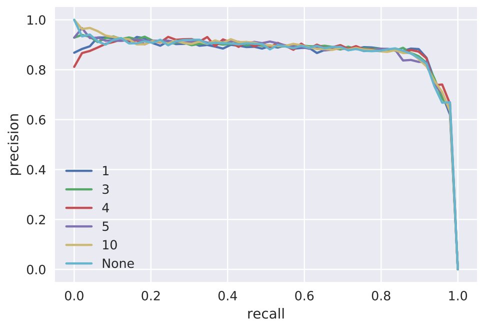
Similarly to the dropout plots, fig. 13 shows the training loss over time and the precision-recall curve at various weight decay values. There are again no significant differences, with a value of 1×10^{-4} performing just slightly better.

maximum L2 norm of weight vectors	F1 mean	F1 stddev	AoC mean	AoC std	Area unde
1	0.905442	0.021042	0.83455	0.0638687	
3	0.90467	0.0276169	0.822277	0.0430833	
4	0.906523	0.0196377	0.837767	0.0434736	
5	0.897128	0.0302341	0.830476	0.0322875	
10	0.892791	0.0236014	0.834981	0.0507315	
None	0.897733	0.0172551	0.83455	0.0355257	

Table 2 – The F1 and AoC scores at various decay values.

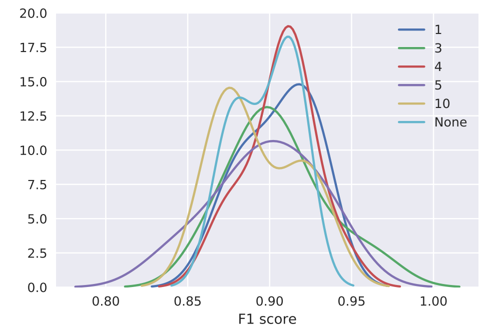


(a) The average loss at each epoch.

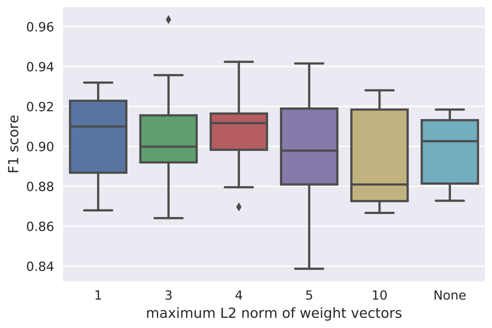


(b) The average precision-recall curves at various decay values.

Figure 13 – The loss over time (a) and the precision-recall curve (b). In both cases, the values are averaged over 10 trials.



(a) Kernel density estimation



(b) Boxplot

Figure 14 – A kernel density estimation and boxplot, based on the F1 score values over 10 repeated trials.

The boxplot in fig. 15b paints a clear picture with three tiers of performance: the model without cluster labels performs worst, adding only the cluster of center of the window improves it a bit, while using the full window performs best with and without a CNN applied to it. Figure 15 showns an interesting relation between the two top performing models. Their score distributions are clearly bimodal with both peaks occurring at roughly the same scores, indicating the possibly of getting caught in a local optimum. The real win of the simpler model without the CNN appears to be that it is far less likely to get stuck in that local optimum.

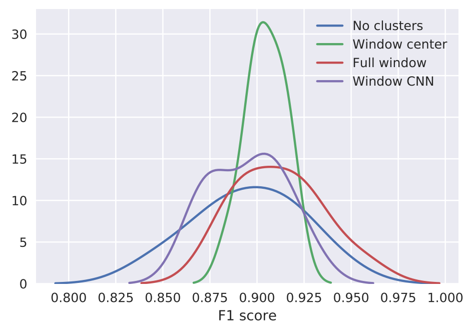
Significance values for the F1 score are given in table 4, which shows that every model is different from every other model using the common cutoff of $p <= 0.05$. The least convincing is the difference between the two best models (Full window and Window CNN), but given the still significant result it is reasonable to state that the Full window model is the clearly winner given its far more reliable ability to reach the global optimum.

6.3 Number of clusters

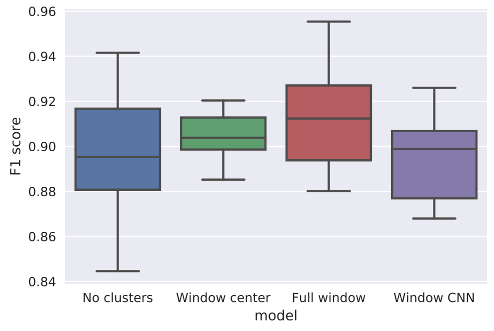
Figure 16 shows the results of varying the number of clusters created in the final k-means clustering step.

model	F1 mean	F1 stddev	AoC mean	AoC std	Area under averaged curve
No clusters	0.895236	0.0272969	0.827952	0.0603747	0.869437
Window center	0.904611	0.00996045	0.82878	0.024061	0.89951
Full window	0.91161	0.0219068	0.873494	0.0434952	0.913144
Window CNN	0.894705	0.0188724	0.818123	0.0375411	0.873791

Table 3 – The F1 and AoC scores of the various models.



(a) Kernel density estimation



(b) Boxplot

Figure 15 – A kernel density estimation and boxplot, based on the F1 score values over 10 repeated trials.

	No clusters	Window center	Full window	Window CNN
No clusters	nan	0.325886	0.216172	0.958424
Window center	0.325886	nan	0.418296	0.186375
Full window	0.216172	0.418296	nan	0.0754657
Window CNN	0.958424	0.186375	0.0754657	nan

Table 4 – The probability for each pair of models that their F1 scores are generated by the same underlying distribution (i.e. the probability of the null hypothesis that the models perform the same being true).

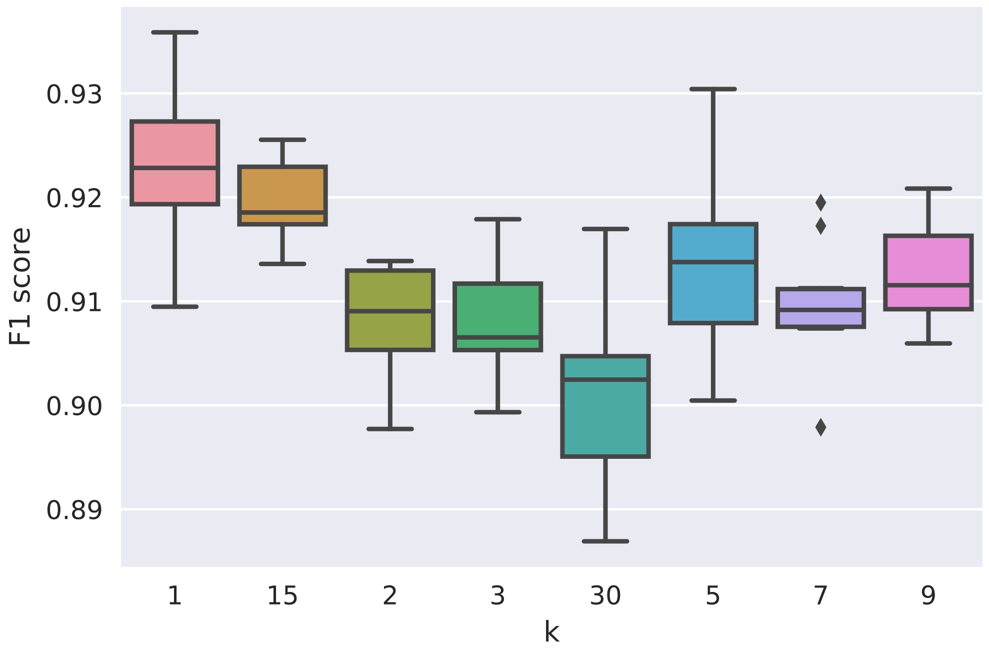


Figure 16 – A boxplot based on the F1 scores of 10 repeated trials for various amounts of clusters created by the k-means algorithm.

7 Conclusion

Todo: conclusion

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. 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).
5. 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.
6. 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).
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. Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *CoRR* **abs/1212.5701.** arXiv: `1212.5701`. `http://arxiv.org/abs/1212.5701` (2012).

13. 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).

14. 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).

15. 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).

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

17. 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).

18. 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).

19. 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).