University of Amsterdam

MSc Artificial Intelligence
Master Thesis

Enriching Textual Data with Document Structure For Sentence Classification

by
Maarten de Jonge
10002109

May 23, 2018

42 EC
01 July 2017, 31 December 2017

Supervisor:
Dr Maarten Marx

Assessor:
Dr To Be Determined

Information and Language Processing Systems Group
Informatics Institute

Contents

1 Introduction

The Political Mashup project[1] aims to digitize the world's political proceedings in order to make them easily accessible and searchable. Unfortunately, the published documents are often primarily intended to be human-readable, without the embedded semantic structure required to properly index this data in a digital way. This semantic information is currently recovered using rule-based methods. Since the data gets transcribed by a human typist, compiled to a PDF, and then goes back into an imperfect PDF decompiler, there is a lot of room for minor variations in the output even though the layout of the document itself is consistent. Dealing with this in a rule-based system entails using either broad rules that lead to a larger probability of false positives, or a large amount of narrow rules which can quickly lead to a spaghetti-like mess of special cases and is very fragile to unseen issues.

I propose that by using a small number of manually annotated documents as a dataset, a machine learning algorithm can learn to classify sentences 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. The common ways to do sentence classification (e.g. convolutional neural networks [1] , 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 present. 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 starting in 1949. Figure 1 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.

[1] http://search.politicalmashup.nl/about.html

Having a large corpus of political proceedings like this is wonderful and opens a lot of doors for research regarding political discourse. Figure 2 shows the same page seen in Figure 1, 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 Figure 2 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 Figure 2) 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.

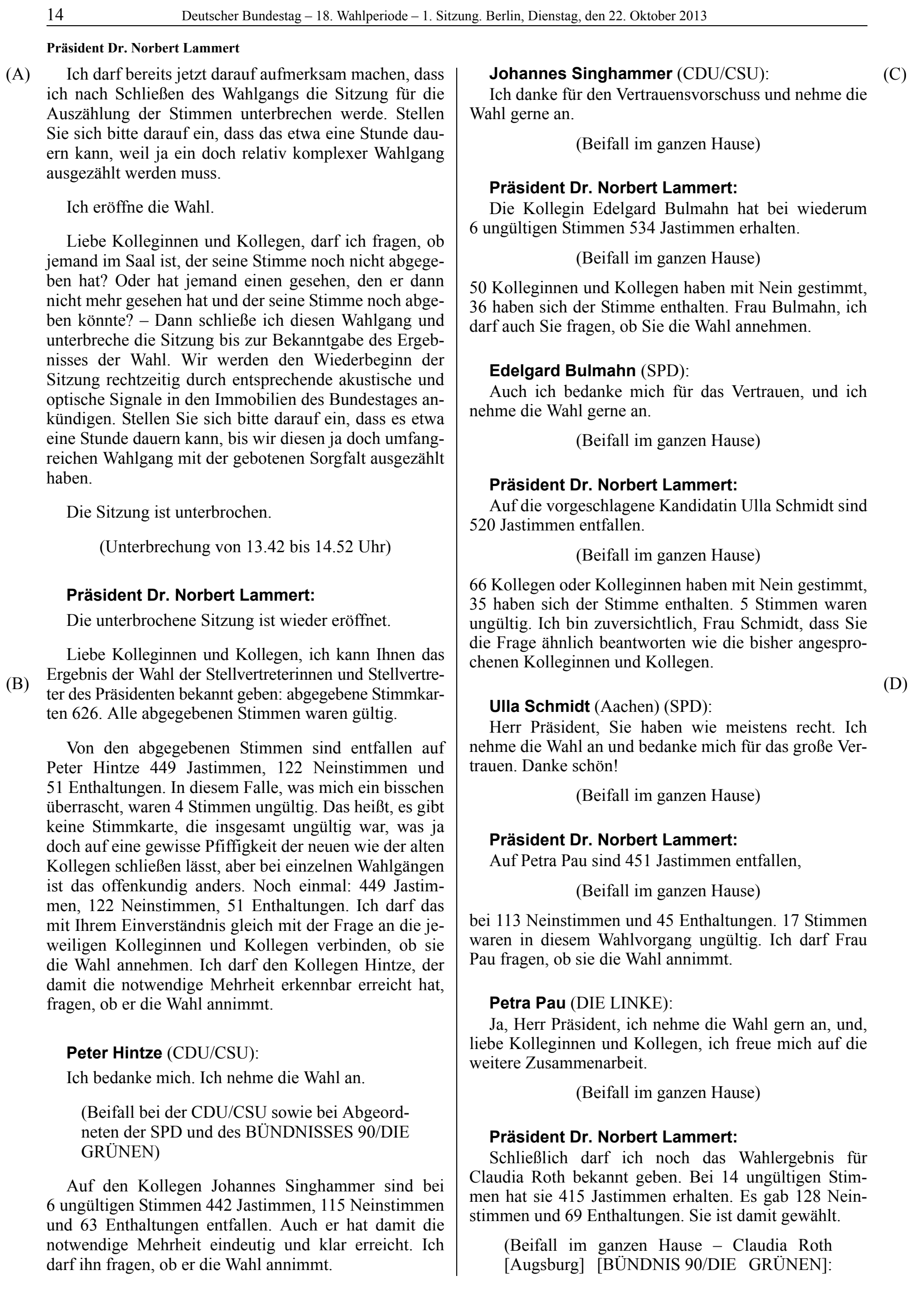
As annotating data by hand is an expensive process, a big focus is on limiting the amount of required training data as much as possible; it would be preferable if the system was able to learn sufficiently from a handful (say, less than 5) of hand-annotated files.

2.1 Dataset

PDF files are unfortunately rather difficult to work with; being a vector-based format, they have no internal concept of words or sentences. All that's available are instructions for drawing a certain character at a certain position. This means that even something as seemingly trivial as obtaining the lines of text from a PDF requires some fairly involved logic and heuristics (for instance, one would think that simply taking all characters on a page with the same y coordinate would be sufficient until realizing that many documents have a layout with two columns of text). This is dealt with by using the `pdftohtml` script from the Poppler PDF rendering library[2]. This script converts a PDF file to an XML file containing logical lines of text along with the coordinates and size of the line. Figure 3 shows an example of a portion of a PDF file and the corresponding XML. The contents of the `text` elements forms the plaintext that is fed into the classification algorithm. Although the layout information contained in the properties (`top`, `left`, `width`, `height` and `font`) could be used for the layout clustering, this would effectively tie the performance of the clustering to the performance of `pdftohtml`'s line-extraction heuristics. Instead, a separate system is used (the Apache pdfbox[3] library for Java). By using the bounding boxes of individual characters as the base for clustering, the troubles of line-extraction can be fully bypassed.

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

[3] https://pdfbox.apache.org/



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.

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. (B)

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 1 – 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.

(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 – The same page as in Figure 1, 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 speakers 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.

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">
    <b>Dr. Norbert Lammert </b>
</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 Alterspr sident , 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 Pr sidenten)
</text>
```


(b) XML created by running `pdftohtml`, corresponding to the PDF excerpt in Figure 3a. 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 3 – A sample excerpt from a source PDF, along with its XML representation created by `pdftohtml`.

The dataset was obtained through a rule-based system as described in the introduction, which annotates each `<text>` element of the XML files with a boolean flag indicating whether said element starts a new speech. The system was run on documents from the 18th electoral period of the *Bundestag*, consisting of 211 documents dating from 2013 to 2017. Together these documents contain 43,252 `<text>` elements indicating the start of speeches (that is, positive training samples), and 2,602,793 other elements (negative samples). This adds to a total of 2,646,045 training samples, taking up 503 MiB. This is a rather lopsided distribution (there are roughly 60 negative samples for each positive sample), which will have to be accounted for by, for instance, using stratified sampling. Figure 4 shows the distribution of the number of positive samples per file, giving a guideline as to how many files would have to be annotated to reach a desired amount of positive samples.

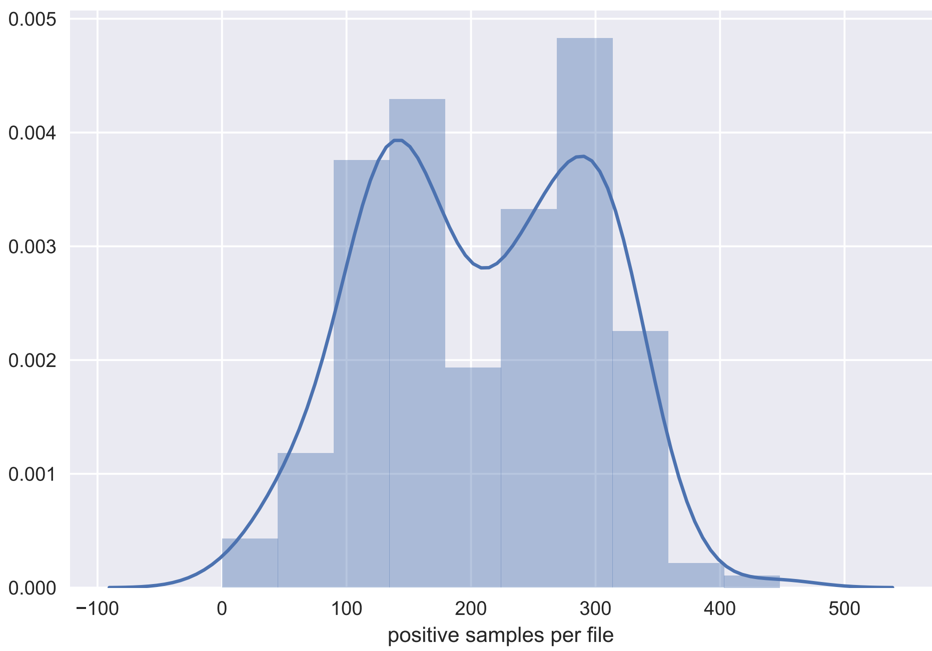


Figure 4 – Distribution of the number of positive samples per file

2.2 Research Question

As the application of supervised text classification is fairly basic and well-studied, academically speaking the interesting portion of this problem is figuring out what benefit is added by the unsupervised layout clustering, if any. A potential benefit could manifest itself in a couple of different ways:

- The performance (using a metric such as the F1 score or average precision, to be further elaborated on in Section 4) after training is increased.
- The peak performance is equal, but less training data is required to reach said peak.
- The peak performance is equal and a similar amount of training data is required, but the classifier needs less iterations to reach this performance.

This leads to the research question that is central to this thesis:

Research Question 1 *Does augmenting a text classification system with layout information obtained by unsupervised clustering of the input data improve the classifier with regards to:*

- *F1 score and average precision*
- *required volume of training data*
- *training speed*

3 Related Work

Todo: Expand section

Various forms of convolutional neural networks are commonly used for text classification. The most basic architecture is described by Kim [1], where the input words are tokenized and embedded before passing them to the convolutional neural network. Additional exploration of the parameter space and its effect on various datasets is done by Zhang & Wallace [2]. Comparable results are achieved by Zhang *et al.* [3] by operating on the character-level rather than the word-level, bypassing the issues overhead of using word embedding (either in extra training time or in finding suitable pretrained embeddings). All the previously mentioned architectures use a single convolutional layer; this is somewhat contrary to current trends in computer vision, where popular models such as ResNet[4] go as deep as 152 layers. This difference is explored by Conneau *et al.* [5], 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.

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. While their approach to block detection forms an integral part of this thesis, the rest is too specifically tied to the format of scientific articles to be applicable in this scenario.

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 next two subsections will go into details on the two clustering steps.

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 Figure 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:

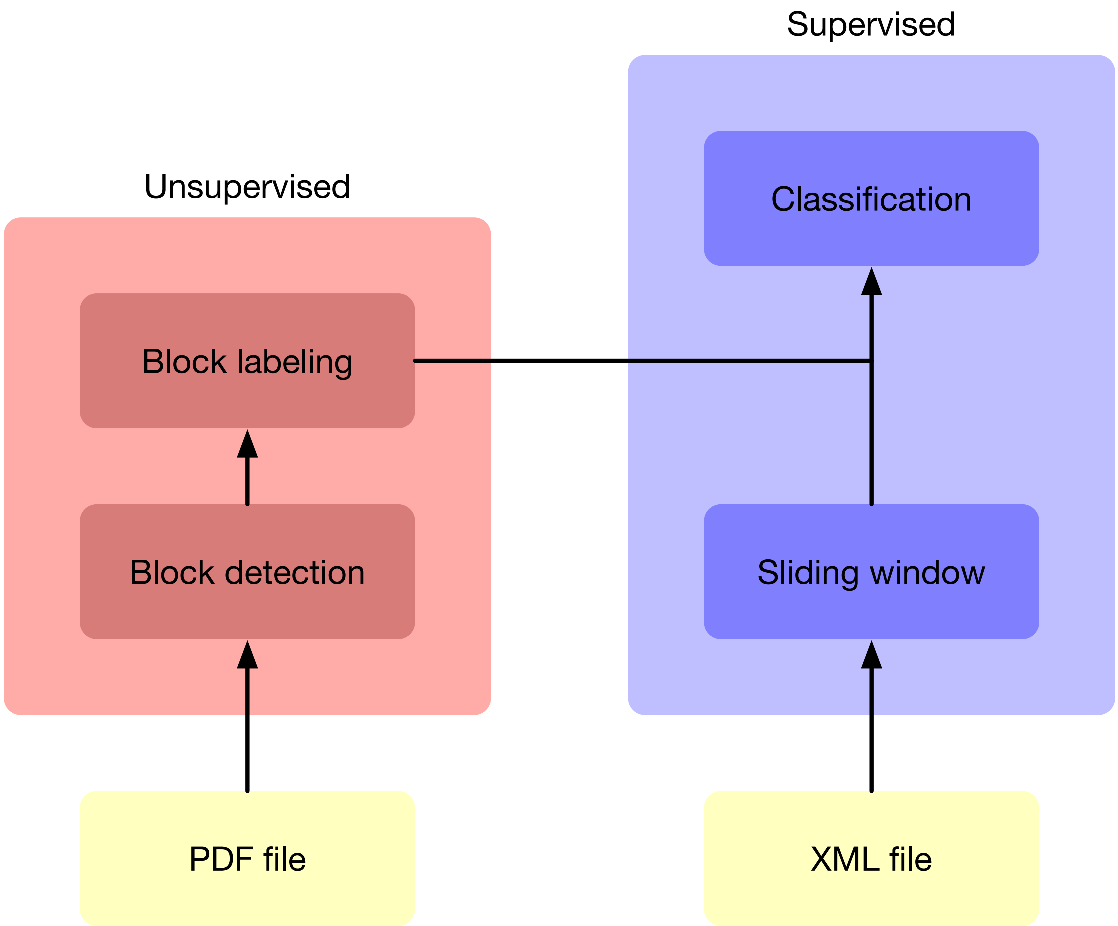


Figure 5 – A high-level overview of the system. The unsupervised block augments the input to the classifier.

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

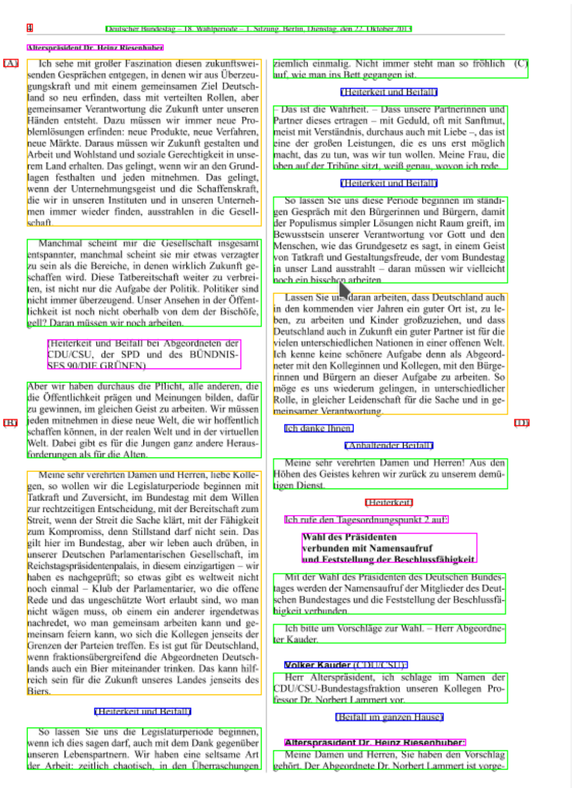


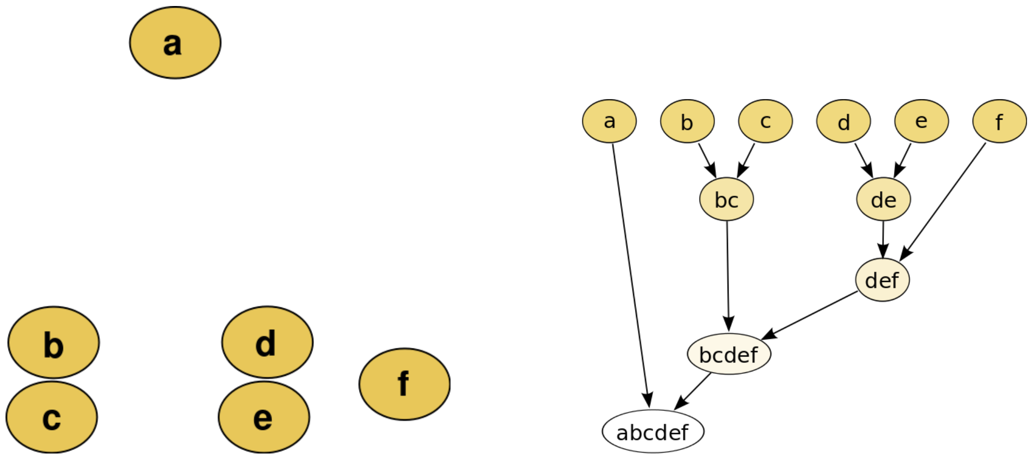
Figure 6 – An example of clustered blocks of text. Blocks with the same outline color belong to the same cluster.

- 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 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 $\mathcal{O}(n^2)$ [7], 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.



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

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

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 both the label (i.e. does or does not start a speech) and the cluster type. The text content of this window is then entered into a standard convolutional neural network architecture similar to the one proposed by Kim [1]. 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 pre-trained German models), followed by a number of Convolutional filters with max pooling. The output of the filters is then concatenated into a single feature vector, which is amended with the cluster type as an additional feature before feeding it 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 Figure 8 along with its parameters and their baseline value.

The network is trained for 100 epochs, stopping early once no improvement in training loss has been made for 10 epochs. The optimisation process is done using the Adadelta[8] algorithm, with binary cross-entropy as the loss function. The loss will exhibit some up and down fluctuations after the global 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 obtained.

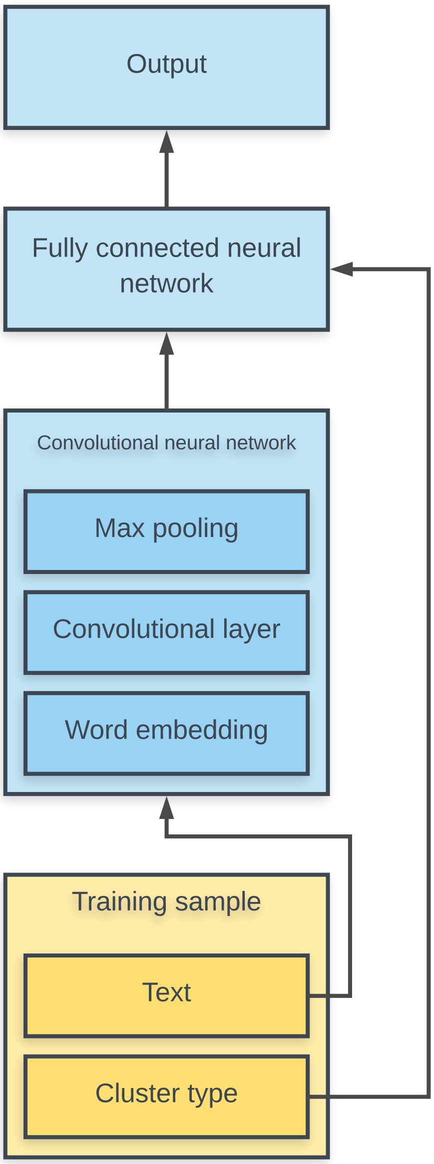
4.2.1 Convolutional Neural Networks

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



(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	3
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.

Figure 8 – The model and its parameters.

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 Figure 9. In this example, the input text is convolved with a filter with a width of 3 and a

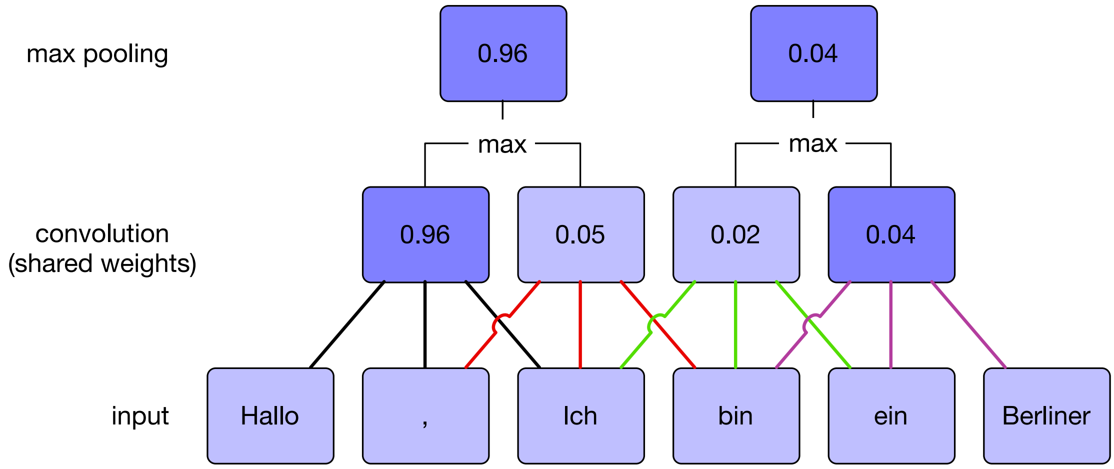


Figure 9 – A simplified convolutional neural network with one filter and max pooling.

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. [9] 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 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 Figure 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 [2]. Since this particular task at first glance appears to be fairly reliant on word position (e.g. a colon at the end of a sentence very often indicates the start of a speech, a colon in the middle almost certainly does not), the degree of pooling will be experimented with.

4.2.2 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.[10]

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.[11]

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 4, 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

6.1 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:

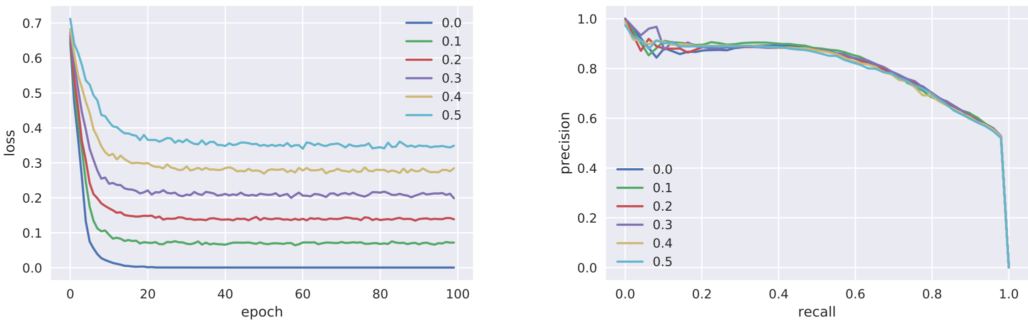
- Dropout, with values taken from the set $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.
- Weight decay, with values taken from the set

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

Further details on these methods are given in section 4.

6.1.1 Dropout

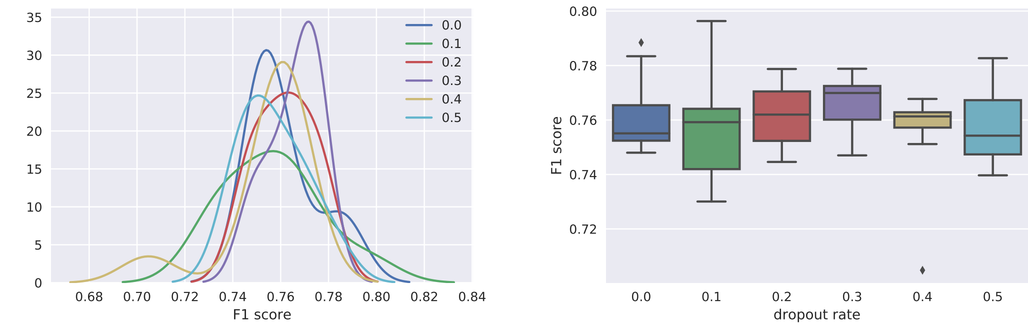
Figure 10 shows the training loss over time, as well as the resulting precision-recall curve on the training set. Other than the training loss being higher with higher dropout values (which is to be expected), there does not seem to be any difference in performance. This lack of effect is also shown in fig. 11. A dropout rate of 0.3 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) The average loss at each epoch.

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

Figure 10 – 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

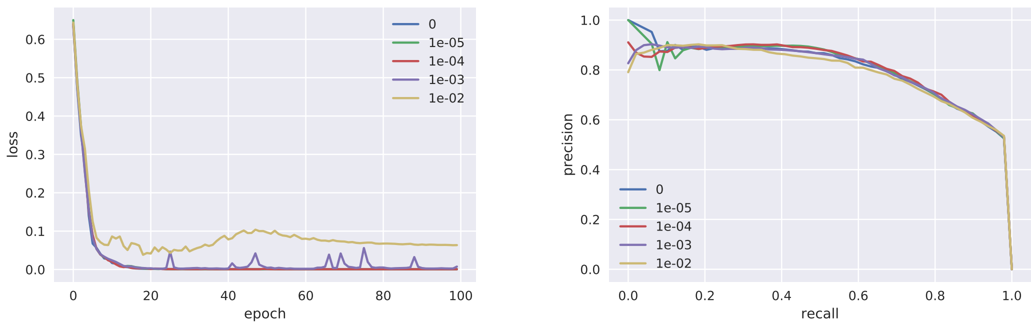
Figure 11 – A kernel density estimation and boxplot, based on the F1 score values over 10 repeated trials.

dropout rate	F1 mean	F1 stddev	AoC mean	AoC std	Area under averaged curve
0	0.76126	0.0133728	0.714461	0.0105124	0.799083
0.1	0.756681	0.0190385	0.714664	0.0120656	0.809108
0.2	0.761842	0.0115077	0.716461	0.010563	0.801978
0.3	0.765695	0.0101811	0.715871	0.00994032	0.810398
0.4	0.755077	0.0172803	0.716783	0.0105444	0.801529
0.5	0.756964	0.0130938	0.717907	0.0143531	0.79936

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

6.1.2 Weight decay

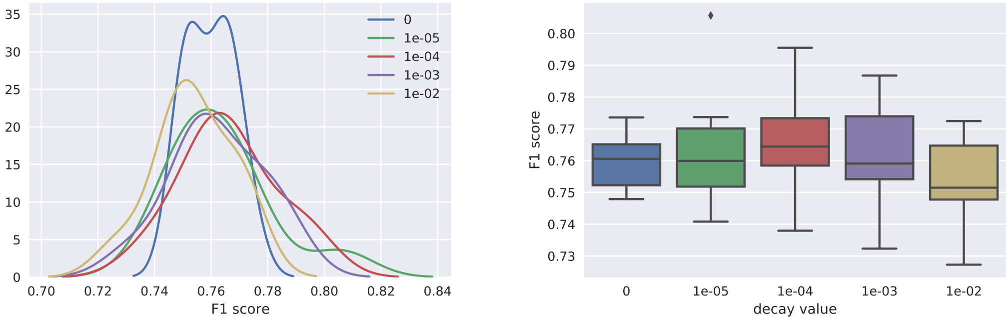
Similarly to the dropout plots, fig. 12 shows the training loss over time and the precision-recall curve at various weight decay values. There are again no significant differences, with a value of 1×10^{-4} performing just slightly better.



(a) The average loss at each epoch.

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

Figure 12 – 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 13 – A kernel density estimation and boxplot, based on the F1 score values over 10 repeated trials.

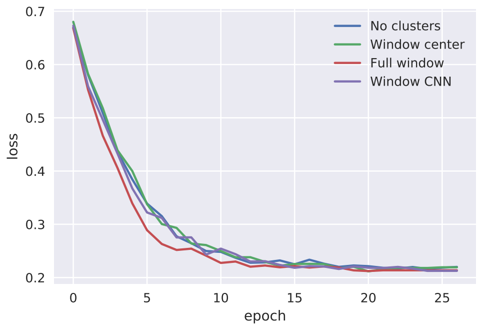
decay value	F1 mean	F1 stddev	AoC mean	AoC std	Area under averaged curve
0	0.759594	0.008119	0.710192	0.0078634	0.805901
1e-05	0.763445	0.0171368	0.715502	0.0113409	0.807963
0.0001	0.766299	0.0160939	0.732198	0.0102998	0.80583
0.001	0.762132	0.01537	0.736607	0.0131718	0.801772
0.01	0.753685	0.013023	0.737005	0.0150467	0.791353

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

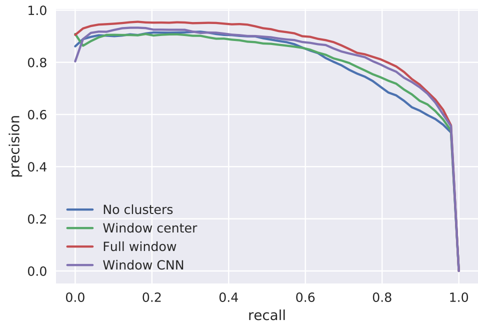
6.2 Models

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 0.3 and the decay to 1×10^{-4}. The plots in Figure 14 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.

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.

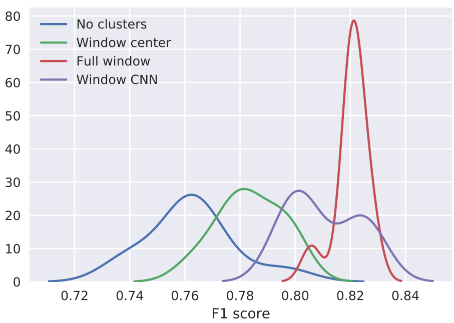


(a) The average loss at each epoch.

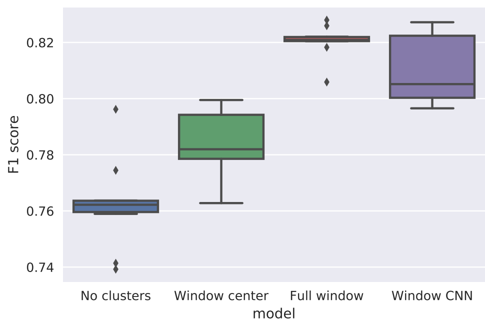


(b) The average precision-recall curves for the various models.

Figure 14 – 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 15 – A kernel density estimation and boxplot, based on the F1 score values over 10 repeated trials.

model	F1 mean	F1 stddev	AoC mean	AoC std	Area under averaged curve
No clusters	0.762332	0.0151038	0.748954	0.014515	0.814618
Window center	0.783569	0.0111505	0.755135	0.0127125	0.818773
Full window	0.82046	0.00554701	0.791899	0.0122418	0.871375
Window CNN	0.810462	0.0120246	0.773475	0.0158784	0.846052

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

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.

	Window center	Full window	Window CNN
No clusters	0.0193663	6.83171e-06	1.51942e-05
Window center		9.04756e-06	0.000636625
Full window			0.0356536

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).

6.3 Number of clusters

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

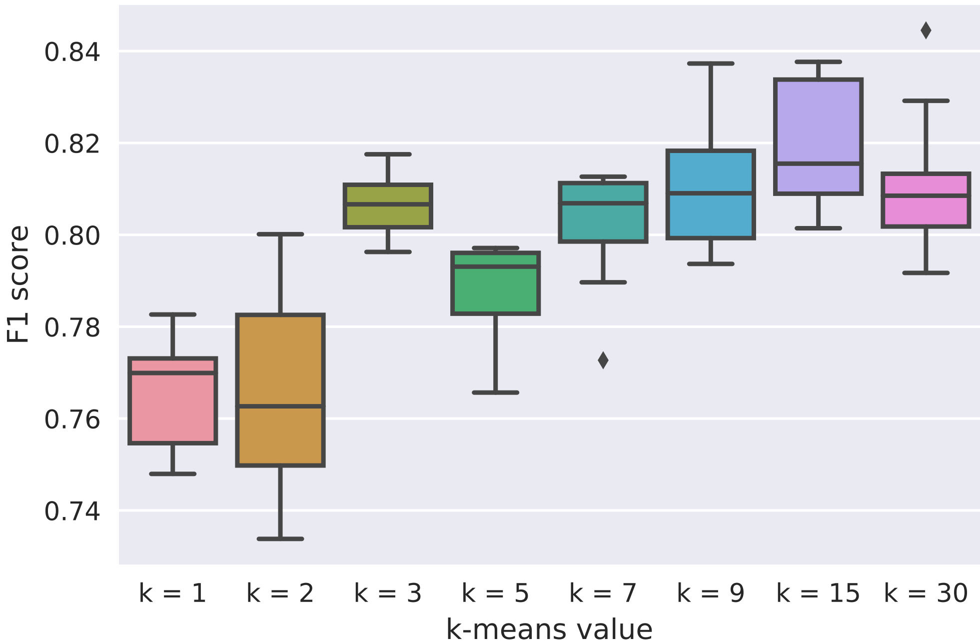


Figure 16 – A boxplot based on the F1 scores of 10 repeated trials for various amounts of clusters created by the k-means algorithm.

7 Conclusion

Todo: conclusion

References

1. Kim, Y. Convolutional Neural Networks for Sentence Classification. *CoRR* **abs/1408.5882.** arXiv: `1408.5882`. `http://arxiv.org/abs/1408.5882` (2014).
2. 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).
3. Zhang, X., Zhao, J. & LeCun, Y. *Character-level convolutional networks for text classification* in *Advances in neural information processing systems* (2015), 649–657.
4. 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).
5. 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).
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. Sibson, R. SLINK: an optimally efficient algorithm for the single-link cluster method. *The computer journal* **16,** 30–34 (1973).
8. Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *CoRR* **abs/1212.5701.** arXiv: `1212.5701`. `http://arxiv.org/abs/1212.5701` (2012).
9. 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).
10. 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).

11. 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).