UNIVERSITY OF AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Document Structure Analysis By Means Of Sentence Classification

by

Maarten de Jonge

10002109

April 11, 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

Chapter 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

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

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.

Chapter 2

Related Works

Todo: Expand section

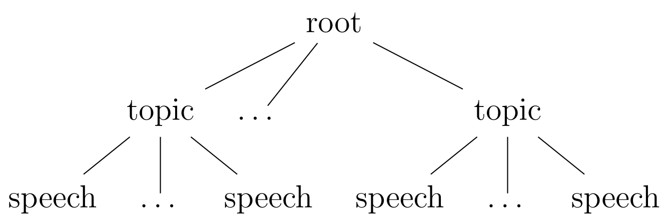
Various forms of convolutional neural networks are commonly used for text classification. The most classical architecture is described by Kim [1], where the input words are tokenized and embedded before passing them to the convolutional neural network. Zhang et al. [4] show that comparable results can be achieved by operating on the character-level rather than the word-level.

In terms of analyzing document structure, Klampfl et al. [2] 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.

Chapter 3

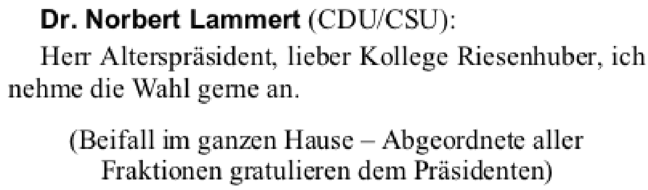
Problem Statement

The thesis will focus on parliamentary proceedings from the German *Bundestag*. These proceedings are available online as PDF files dating back to 1949, and have used essentially the same document layout and conventions since the start. Semantically, the layout of these documents forms a shallow tree:



Each document consists of a series of topics to discuss, where each topic contains a series of speeches made by the present politicians. The task to be solved is retrieving these speeches as accurately and robustly as possible. Luckily, in this case a rule-based system for segmenting the files already exists and was used to create a fairly large training set. Since in general such a system will not exist and annotating the data by hand is expensive, 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.

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 realising 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[1]. 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 3.1 shows an example of a portion of a PDF file and the corresponding XML produced by `pdftohtml`.


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) 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

Figure 3.1: The data representations

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

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

training samples, taking up 503 MiB. This is a rather lopsided distribution (there are roughly 60 negative samples for each positive sample), which might have to be accounted for by, for instance, subsampling negative samples before training. Figure 3.2 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.

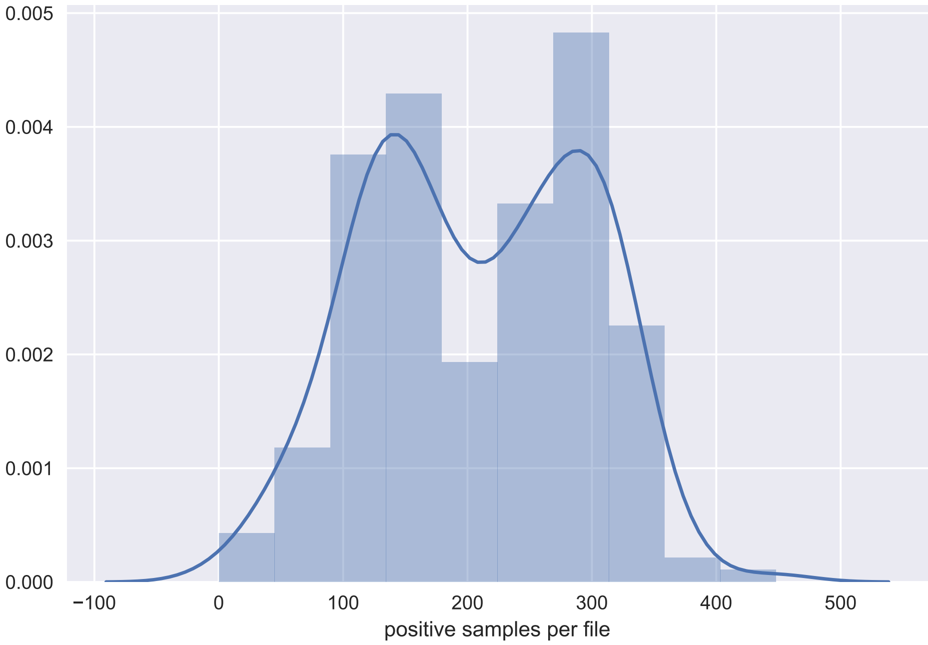


Figure 3.2: Distribution of the number of positive samples per file

Chapter 4

Approach

The system consists of two separate parts; an unsupervised algorithm for augmenting data, and a supervised algorithm for classifying the data (Figure 4.1). The unsupervised portion attempts to augment the data with additional structural information. It could be considered a preprocessing step, with the choice of parameters acting as a way to inject some amount of domain knowledge into the data. The supervised portion consists of a regular classification algorithm.

Todo: Rewrite this and add all the details about how the data is used (sliding window, etc)

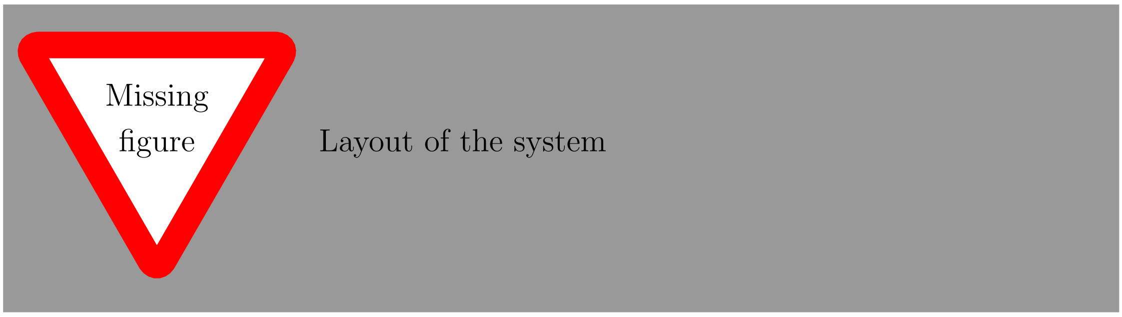


Figure 4.1: A high-level overview of the system

4.1 Unsupervised

The unsupervised algorithm attempts to detect and label blocks of text in the PDF file, as shown in Figure 4.2 for an example). This approach is based on work by Klampfl et al. [2], and consists of two clustering steps:

1. Individual letters are clustered together into blocks of semantically relevant text (e.g. a full paragraph, or a section header).
2. These blocks are labeled by a different clustering algorithm.

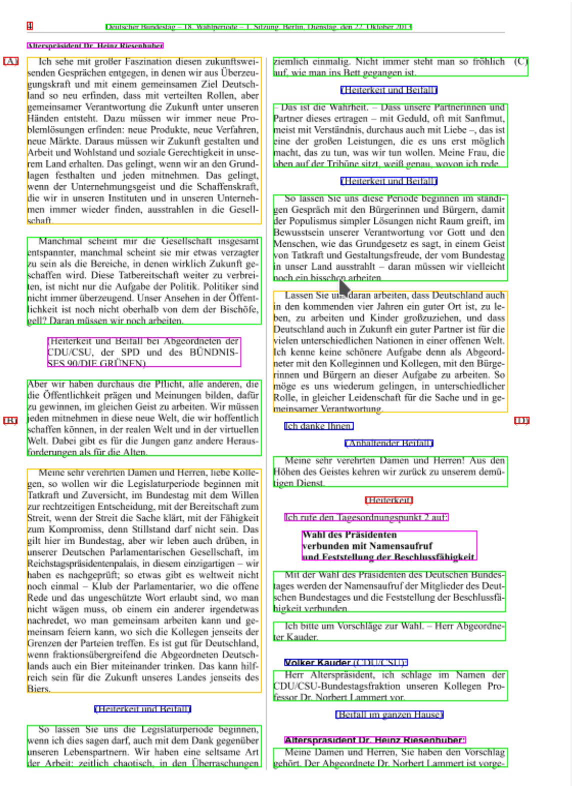


Figure 4.2: An example of clustered blocks of text, blocks with the same outline color belonging to the same cluster.

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 4.3. The algorithm gets fed the individual characters present in the PDF files, then iteratively groups the two closests clusters (the initial inputs being regarded as clusters of one element) together until only a single cluster remains. This process involves two parameters:

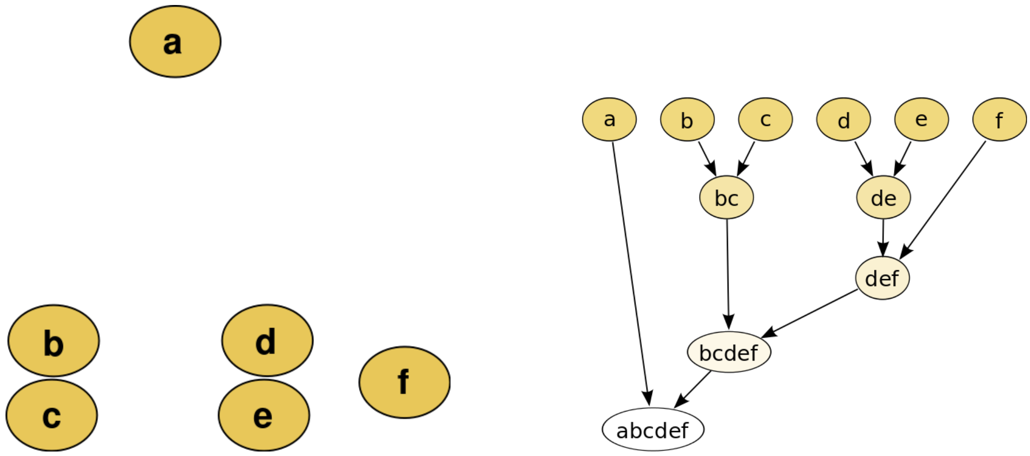
1. The distance function between two characters.

2. The distance function between two groups 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 groups 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 groups 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 groups 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. [2], single-linkage clustering performs best for this task due to its tendency to form long thin clusters, mimicking the structure of 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 $\mathcal{O}(n^2)$ [3], making it far more usable on realistic datasets.



(a) Before (b) After

Figure 4.3: An example of hierarical agglomerative clustering.

After the dendrogram is concstructed, the only choice left is at which level to cut the tree to obtain the desired blocks of text. This is left as a parameter to be manually finetuned.

4.1.2 Classical Clustering

The extracted blocks from the previous step are then clustered according to the similarity of their shapes (width and height). This is done using K-means clustering for some chosen K, or with the DBSCAN algorithm.

Todo: Either expand this section or just integrate it with the previous subsection.

4.2 Supervised

After the data is augmented by the previously described clustering algorithm, it's fed into a supervised classifier, more specifically a convolutional neural network (CNN). The decision to use a CNN was motivated by both its performanceKim [1] and, for practical reasons, its superior training speed compared to recurrent neural networks.

4.2.1 Convolutional Neural Networks

Todo: Explain CNNs

Chapter 5

Results

Todo:

- Difference between running with/without clustering data, vary number of clusters
- CNN vs RNN, probably conclude that RNNs offer no benefit to compensate for the added computational cost
- vary stuff like the embedding size and number of filters
- add tests on older data

Chapter 6

Conclusion

Todo: conclusion

Bibliography

[1] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *CoRR* abs/1408.5882 (2014). arXiv: `1408.5882`. URL: `http://arxiv.org/abs/1408.5882`.

[2] Stefan Klampfl et al. "Unsupervised document structure analysis of digital scientific articles". In: *International Journal on Digital Libraries* 14.3-4 (2014), pp. 83–99.

[3] Robin Sibson. "SLINK: an optimally efficient algorithm for the single-link cluster method". In: *The computer journal* 16.1 (1973), pp. 30–34.

[4] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification". In: *Advances in neural information processing systems*. 2015, pp. 649–657.