Bachelor’s thesis

for the degree of

Bachelor of Arts

presented to the Faculty of Arts and Social Science

of the University of Zurich

Text Zoning for Job Advertisements with Bidirectional LSTMs

Ann-Sophie Gnehm

Examiner: Prof. Dr. Martin Volk

Supervisor: Dr. Simon Clematide

Institute for Computational Linguistics

Submission date: January 12, 2018
Abstract

The thesis at hand presents an approach to text zoning for job advertisements with bidirectional LSTMs (BiLSTMs). Text Zoning refers to segmenting job advertisement into eight zones that differ from each other regarding content. It aims at capturing text parts dedicated to particular subjects (e.g. the publishing company, qualifications of the person wanted, or the application procedure) and hence facilitates subsequent information extraction. As we have 38,000 job advertisements in German from 1950 to 2014 available (Swiss Job Market Monitor corpus), each labeled with text zones, we benefit from a large amount of training data for supervised machine learning. We use BiLSTMs, a class of recurrent neural networks particularly suited for sequence labeling, as they integrate well information over long sequences and consider context on both sides of the actual label for classification decisions. Our best model reaches a token-level accuracy of 89.8%, which is 2 percentage points above results from previous approaches with CRFs and implies an error rate reduction by 16%. Models with task-specific embeddings perform better than models with pretrained word embeddings, which is probably due to the large amount of labeled training data. When optimizing the model for future application on recently published job advertisements, the inclusion of older training data lowers performance, as some sort of out-of-domain effect counteracts the effect of more training data. Ensembling, i.e. to aggregate classification decisions of five models, brings the largest improvement of all optimization steps, raising accuracy by 0.5 percentage points. In conclusion, we succeeded in building a high performing solution to automatic text zoning for job ads based on neural networks.
Zusammenfassung

# Contents

1 Introduction ........................................................................................................................................ 1

2 Related Work ................................................................................................................................... 4

3 Data & Methods .............................................................................................................................. 7

   3.1 Data ........................................................................................................................................... 7

      3.1.1 SJMM Corpus: Size, Time Period Covered, Sources ................................................................. 7
      3.1.2 Text Zoning .......................................................................................................................... 7
      3.1.3 Distribution of Text Zones: Variation over Time and Publication Channels ....................... 10
      3.1.4 Distribution of Vocabulary over Text Zones .......................................................................... 11
      3.1.5 Vocabulary in Text Zones over Time ..................................................................................... 12
      3.1.6 Preprocessing and Encoding of Job Ads .............................................................................. 15

   3.2 Methods ...................................................................................................................................... 15

      3.2.1 Sequence Labeling with Bidirectional Long Short-Term Memory Networks (BiLSTMs) .... 15
      3.2.2 Feature Engineering ............................................................................................................. 16
      3.2.3 Pretrained Word Embeddings as Features .......................................................................... 16
      3.2.4 Task-Specific Embeddings ................................................................................................... 18
      3.2.5 BiLSTMs Architecture and Toolkit ...................................................................................... 19

4 Experiments & Results .................................................................................................................. 21

   4.1 Experiments .............................................................................................................................. 21

   4.2 Results ...................................................................................................................................... 23

      4.2.1 Initial Neural Network Architecture .................................................................................... 23
      4.2.2 Task-Specific versus Pretrained Embeddings ..................................................................... 23
List of Figures

Figure 1: Example of transcribed and manually segmented job ad .......................................................... 9
Figure 2: Number of tokens per text zone 1950-2014 (press, moving averages over 3 years) ........ 10
Figure 3: Share of text zones per publication channel (2010-2011) .................................................... 11
Figure 4: Top 25 words for company & job description, hard & soft skills (up to and after 1989)... 14
Figure 5: Token-level BiLSTMs ........................................................................................................... 20
Figure 6: Character-level BiLSTMs ....................................................................................................... 20
Figure 7: Effects of training size on development set accuracy ............................................................... 25
Figure 8: Frequency distribution of error rates per job ad ................................................................. 28
Figure 9: Example of a job ad showing a high error rate ....................................................................... 29
Figure 10: Relationship between error rate and text length ................................................................. 30
Figure 11: Text passage with acceptable differences between predicted and true labels ............... 32
Figure 12: Precision and recall of classification on the test set ......................................................... 33
List of Tables

Table 1: Definitions and examples of text zones ................................................................. 8
Table 2: Distribution of tokens and types over text zones .................................................. 12
Table 3: Class ambiguity of tokens and types ...................................................................... 12
Table 4: Coverage of SJMM vocabulary by Conceptnet embeddings .................................. 17
Table 5: Development set accuracy of models with task-specific and pretrained embeddings .... 23
Table 6: Development set accuracy for model with different hyperparameter settings ........... 26
Table 7: Development set accuracy with training data covering different time periods .......... 27
Table 8: Confusion matrix of classification on the development set ....................................... 28
Table 9: Evaluation of single models and ensembling on development set and test set .......... 31
Table 10: Confusion matrix of classification on the test set .................................................. 34
Table 11: The 50 most frequent wrongly classified tokens, frequencies in brackets ............. 35
1 Introduction

Job openings are a highly interesting text type in several respects: While they show certain standardization in structure, form and style, their content is rather diverse and varied. Furthermore, they do not only depict labor market demand in terms of occupations and qualifications needed, but also provide information on hiring strategies, self-representation of companies, forms of division of labor, descriptions of actual job tasks, the employer-employee relationship and the like. In this way, job openings are an excellent source for economical, psychological and sociological research on labor and labor market (see Buchmann et al. 2016).

The Swiss Job Market Monitor (SJMM)¹ is devoted to a systematic monitoring and analysis of the Swiss labor market. For this purpose, SJMM has set up a monitor corpus of job openings, reaching back to 1950. The corpus is based on representative samples of job ads published in the three most important media channels – corporate websites, online job portals and press. Based on the information in the job opening texts, a whole range of annotations on characteristics of the job, the person wanted, and the company offering the job, are manually added. The SJMM corpus hence provides a unique and rich database for qualitative or quantitative (labor market) research, enabling to analyze both long-term developments as well as current trends.

Labor market research, as well as social science at large, profit from methods of computational linguistics (CL) or natural language processing (NLP) in two respects: Firstly, NLP allows replacing manual annotation of texts by semi- or fully automatic data processing. Both procedures typically lower costs of data collection, permitting to increase sampling size. Hence, this allows performing more detailed analyses and thus addressing more complex or interesting research questions. Secondly, NLP brings new methods to social science, as for instance, co-occurrence analysis or topic modeling. In this way, it also opens up completely new research prospects, for the SJMM as well as for labor market research and other social science disciplines at large. The aim of this study is to partially supersede manual annotation by automatic data processing with supervised machine learning, in order to enlarge research opportunities.

Text zoning is a very important part of the manual annotation of the SJMM corpus, and this procedure is also supposed to be one of the first steps in the workflow of automatic information extraction. Text zoning refers to segmenting the whole job advertisement (job ad) text into zones (or classes), that differ from each other with regard to their content. Text zoning aims at specifically capturing text parts dedicated to particular entities (or subjects) as for instance the company, qualifications of the person wanted, or the application procedure. In the SJMM corpus, to cover everything that appears in a job ad

¹ http://www.stellenmarktmonitor.uzh.ch
with a certain frequency and that is expressed by a certain amount of text, eight different text zones are
distinguished²:

Z10: description of the company publishing the ad,
Z20: reason of the vacancy (e.g. corporate growth),
Z30: administration and residual text,
Z40: description of a job agency,
Z50: material incentives,
Z60: description of the job task,
Z70: required qualification in terms of education, training and skills (hard skills),
Z80: required personality and traits (soft skills).

In this way, text zoning allows analyzing which quantities or shares of text are dedicated to certain
entities or subjects. To give a simple example, this allows analyzing how the relation between required
qualifications and offered incentives evolves over time, or differs between job ads. But more
importantly, this procedure facilitates subsequent automatic information extraction. Firstly, text parts of
interest can be localized more precisely and the amount of text to be analyzed is reduced. Secondly, it
allows disambiguating words with more than one sense, or to decide if text passages actually refer to the
subject of interest. Imagine for instance, that we want to extract required soft skills for a vacant position:
To know if the keyword "dynamic" refers to the personality wanted indeed, or to a "dynamic working
atmosphere" in the publishing company, or to a "dynamic CRM system" as working tool instead,
simplifies the information extraction task considerably. In conclusion, text zoning provides structure to
job ad texts and therefore, represents a substantial information gain. By this means, it is useful to all
methods of NLP typically applied in social science, such as frequency analysis, gazetteer-based content
analysis, topic modeling, co-occurrence analysis, as well as supervised or unsupervised machine learning
with various target variables (see for NLP in social science e.g. Lemke & Wiedemann 2016, p. 96). For
these reasons, text zoning seems to be the most appropriate starting point for superseding manual
annotation by automatic data processing.

² Detailed information on the SJMM corpus and text zoning guidelines are provided by the SJMM upon request.
The goal of this work is therefore to develop an automatic solution to text zoning for job ads. As a first step, we will test how well results of manual segmentation can be reproduced with supervised machine learning for the SJMM corpus as a whole. The automatic text zone classifier is intended to supersede manual segmentation and hence will mainly be used to segment current job ads or ads published in the near future. Consequently, automatic classification will then, as a second step, be optimized regarding job ads from most recent years (i.e. from 2010 to 2014). In the course of this, we will also test, if results of previous experiments with classic algorithms (Conditional Random Fields) can be enhanced. The task will be tackled with Recurrent Neural Networks (RNNs), which recently have proven to be very effective for a very broad range of NLP tasks, some of them also very similar to text zoning. Keeping in mind that in the present case text zoning is mainly supposed to facilitate information extraction for the SJMM research interests, not every single class is of equal importance. Good classification results are especially valuable regarding the description of the organization and the open position, as well as for text parts referring to qualifications and personality traits of the person wanted. In developing an automatic text zoning solution, we will focus on meeting these requirements.

The rest of this thesis is structured as follows: Section 2 presents goals, methods and important insights of related work on text zoning for job ads and other text types. Section 3.1 describes the SJMM corpus and gives further insight into difficulties of the text zoning task at hand. The methodological approach towards the task is presented in Section 3.2. Section 4.1 describes the experiments conducted and Section 4.2 provides a thorough discussion of the results, points out limitations and suggests future work. Section 5 summarizes the most important findings of the thesis.
2 Related Work

A previous approach to automatize existing manual text zoning of the SJMM (Gnehm 2016), with Conditional Random Fields (CRFs) and relatively simple features, such as token unigrams and bigrams, part-of-speech (PoS) bigrams, and information on the relative position of the token in the job ad text\(^3\) achieved an accuracy on token level of 87.7%. CRFs allowed considering classification of previous elements for the classification of the actual token and in this way optimizing the prediction of global tag sequence.

Hermes & Schandock (2016) conducted experiments in text zoning for job ads as well. In their approach, to generate training data for supervised machine learning, they manually segmented 280 job ads in 1,500 paragraphs in total. They distinguished only four classes, namely information on the employer, information on the job, required skills and administration (or residual) text. Since they intended to classify paragraphs as a whole, they allowed assigning multiple labels to one paragraph, as paragraphs often contain information on different text zones. To reach high precision, in a first step, a rule-based classifier, which made use of 65 predefined expressions, assigned labels to paragraphs. Subsequently, they experimented with several statistical classifiers – Naive Bayes, Rocchio, Support Vector Machines and K-Nearest Neighbors algorithm – in order to improve recall. Their best approach, K-Nearest Neighbors (with K=4) outperformed all other algorithms by far with an accuracy of 97%.

Though dealing with the same task in general, results of Hermes & Schandock (2016) are not directly comparable to results for text zoning on the SJMM corpus. Firstly, Hermes & Schandock (2016) classified whole paragraphs, not single tokens, and – as a consequence, to reach adequate description quality – they allowed multi-label classification. Furthermore, their classification system is only half of the size of the SJMM classification system. Finally, they classified paragraphs independently from each other and did not model job ads as sequences of paragraphs.

The largest part of other research on text zoning, especially in recent years, deals with scientific papers or abstracts. In this context, the task is often also referred to as argumentative zoning. A smaller part of research focuses on longer and more general texts, such as newspaper or magazine articles. The goals behind text zoning are in both fields of application very similar to the goals in the present thesis: Approaches for general texts as well as for scientific text try to facilitate and enhance automatic summarization, information retrieval and question answering systems. All of these NLP applications gain considerably from access to small and coherent text segments (e.g. Sun et al. 2008, Merity et al. 2009). For instance, an automatic summarization system for scientific texts should treat the same sentence very differently depending on its rhetorical context (Moens & Teufel 2002): The same information, e.g. about a protein interaction, may express an already known result, a conjecture or a new result. Since

\(^3\) The job ad text is segmented in ten parts of equal length, and for every token the information to which part it belongs is stored.
information extraction often heavily relies on surface information, such as lexical or syntactical patterns, to know the argumentative zone of the information, facilitates and improves information extraction considerably (Mizuta et al. 2006).

Most of these approaches (e.g. Brants et al. 2002, Guo et al. 2010, Guo et al. 2011, Merity et al. 2009) classified on the level of sentences, and assigned every sentence one single class (or text zone). The basic assumption behind this might not always be true, but it simplifies the task considerably (Merity et al. 2009). Mizuta et al. (2006), however, classified on the level of constituents, as they observed many sentences which consisted of clauses relating to different text zones. Moreover, they allowed multilabel classifications, as a constituent in a scientific text can refer to different text zones at the same time, e.g. mention previous approaches and link to the contribution of the paper itself.

The previous studies on text zoning for scientific texts used a predefined classification system and distinguished six to seven zones in full papers (Guo et al. 2011, Merity et al. 2009, Teufel & Moens 2002) and three to four classes in abstracts (Guo et al. 2010, Hirohata et al. 2008; Mizuta et al. 2006). For more general texts, such as newspaper or magazine texts, with a broad range of topics, Brants et al. (2002) and Sun et al. (2008) by contrast developed the classification system inductively, by Latent Dirichlet Allocation (LDA) or probabilistic Latent Semantic Analysis (pLSA).

As mentioned above, topic modeling is of use for (domain independent) text zoning for newspaper or magazine articles. Brants et al. (2002) detected topics in block candidates by pLSA, where block candidates were sentences or longer units of variable size (e.g. paragraphs). They chose optimal segmentation boundaries by comparing similarity values between block candidates. Very similar, Sun et al. (2008), detected topics with LDA and use a LDA-based Fisher Kernel to calculate similarities between adjacent blocks. They calculated optimal segmentation boundaries dynamic, that is, block size was variable in this approach as well. In both approaches, the choice of how to compute similarity (which similarity measure to choose), has proven to be essential for performance.

In the first approach on text zoning for scientific texts, Teufel & Moens (2002) used a Naive Bayes Classifier with rather complex, task-specific features, such as verb syntax or different types of formulaic expressions. Merity et al. (2009) improved F-Score for the same task on the same corpus from 76% up to 96.88% using a Maximum Entropy Classifier and modeling the task as sequence labeling: The classifications of up to four prior sequence elements were used as history features and the prediction for the present element hence relies on previous decisions. They used the Viterbi algorithm for global decoding, that is, to find the optimal sequence of classification labels. With this modeling, rather simple word features like unigrams and bigrams were sufficient, more complex and task-specific features didn’t bring considerable additional benefit. Results from Hirohata et al. (2008) showed too that modeling as a sequence labeling task is fruitful and allows going with rather simple features (n-grams, relative sentence location and features from previous or next sentences). While their results are not directly comparable to previous approaches due to different corpora and classification systems, they showed that algorithms with global optimizations outperform algorithms without global optimization as well: In their case,
Conditional Random Fields reached around 5% higher per-abstract accuracy than Support Vector Machines.

From the related work presented above, two important conclusions for the task at hand may be drawn. Firstly, modeling the task as sequence tagging, that is to consider preceding classification decisions for the current classification decision, can improve performance considerably, as Merity et al. (2009) as well as Hirohata et al. (2008) showed. Secondly, if approaches make use of distributional semantics, it is important to find an appropriate modeling: Sun et al. (2008) and Brants et al. (2002) both obtained rather different results depending on the similarity measure chosen. Furthermore, it can be stated, that text zoning for job ads serves a similar purpose as argumentative zoning for scientific texts: It provides access to smaller and more coherent text parts, which facilitates subsequent information extraction. In addition, since information extraction often relies on linguistic surface structure, zoning can provide the context needed to disambiguate keywords or phrases of interest. In contrast to most argumentative zoning approaches, sentences are considered too coarse-grained for a reliable segmentation for our application, as in many cases a single sentence refers to several different text zones (see Figure 1 in Section 3.1.2 for an example).
3 Data & Methods

3.1 Data

3.1.1 SJMM Corpus: Size, Time Period Covered, Sources

While the SJMM corpus as a whole consists of job ads in German, French, Italian and English, labels for text zones are only available for job ads in German language. As the SJMM corpus comprises over 38,000 manually segmented job ads from 1950 to 2014, the amount of training material for supervised machine learning is sufficiently large. For the whole period from 1950 onwards, job ads from the press are recorded by transcription, with certain rules ensuring some text standardization. Most importantly, information that is not of primary interest to the SJMM such as phone numbers or email addresses is replaced with the abbreviations [tel] and [email], respectively (see e.g. Figure 1).¹ In 2010 and 2011, job ads from corporate websites and online job portals were gathered and recorded additionally, normally by copy-pasting without any further editing, or, if not possible by transcription.

The sampling size for the press is around 500 ads per year. In 2010 and 2011 in total around 2,600 ads from corporate websites and 1,600 ads from online job portals were collected. While the largest part of the training material thus stems from the press, for the most recent 5 years, more job ads from online publication channels are available. This reflects changes in the practice of publishing job openings and permits at the same time to develop a classifier suitable for present or future application.

3.1.2 Text Zoning

In text zone, job ad texts are split up in eight different text zones, distinguishable from each other with regard to their contents. All information provided in the job ad is assigned to a particular entity (or subject), like the open position, the labor-seeking company, required qualifications and personality for the position, the application process and so forth. This procedure facilitates subsequent information extraction, as it is for instance essential to know if the keyword "dynamic" refers to the personality wanted or to a "dynamic control system" as part of the company's working tools. Table 1 shows definitions and examples for all eight text zones.

These eight text zones cover all information regularly provided in job ads. The zone Z30 (administration and residual text) is assigned in case no other text zone seems appropriate. However, not every job ad contains information on every single text zone. For instance, most of the job ads are published by the company itself, which implies there is no description of an employment agency (Z40). Likewise, the reason of the vacancy (Z20) remains often unknown. It should furthermore be noted, that every text

¹ Detailed information on editing rules and classification guidelines for text zoning is provided by the SJMM upon request.
passage is assigned only one text zone. Thus, in case a passage contains information on several text zones, a decision has to be made. In conclusion, text zoning for SJMM job ads is modeled as a multiclass classification on the level of token sequences.

<table>
<thead>
<tr>
<th>Z10</th>
<th>description of the company</th>
<th>„ein erfolgreiches Unternehmen der Baubranche“</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z20</td>
<td>reason of the vacancy</td>
<td>„für unsere neu eröffnete Filiale“</td>
</tr>
<tr>
<td>Z30</td>
<td>administration and residual text</td>
<td>„Ihre Bewerbung senden Sie an“, „wir suchen“, „wir bieten“</td>
</tr>
<tr>
<td>Z40</td>
<td>description of a job agency</td>
<td>„Ihr kompetenter Partner für die Vermittlung von Temporär- und Dauerstellen“</td>
</tr>
<tr>
<td>Z50</td>
<td>material incentives</td>
<td>„den hohen Anforderungen entsprechendes Salär“, „5 Wochen Ferien“</td>
</tr>
<tr>
<td>Z60</td>
<td>description of the job</td>
<td>„eine vielseitige Tätigkeit“, „für den Empfang unserer Kunden“, „teilweise Wochenendarbeit“</td>
</tr>
<tr>
<td>Z70</td>
<td>required education, experience, knowledge (hard skills)</td>
<td>„Sie haben eine Ausbildung und Berufserfahrung als Sozialarbeiter“</td>
</tr>
<tr>
<td>Z80</td>
<td>required personality and traits (soft skills)</td>
<td>„Sie sind belastbar, zuverlässig und diskret“</td>
</tr>
</tbody>
</table>

Table 1: Definitions and examples of text zones

Job ads often show typical sequences of these text zones. A common pattern is to start with a description of the company, followed by a description of the job task. Then, required qualifications regarding hard skills and soft skills are listed. In the end, information is provided about the application process and about how to file in an application. But certainly, deviations from this pattern are frequent, and likewise there is variation concerning how often text zones alternate with one another. In manual segmentation, the text is split up in accordance with syntactical units, with the lowest level being a single token. Figure 1 shows a typical example of a manually segmented job ad. As can be seen from the example, the first sentence-like text part refers to three different text zones, indicating that modeling the task on a token level is an appropriate solution.
Wir suchen für unser attraktives Gross-Sortiment eine(n) Dekorateur(in) Wir wünschen:

- Gute Berufsbildung und Erfahrung
- Kreativität, Vielseitigkeit und Erfahrung
- Idealalter 25 – 40 Jahre

Wir bieten:

- Weitgehende Selbständigkeit,
- grosses Atelier
- interessante, vielseitige Dauerstelle.

Auf Ihre Bewerbung [adr], 8048 Zürich, freut sich [kper].

Foto Hobby AG

Figure 1: Example of transcribed and manually segmented job ad
3.1.3 Distribution of Text Zones: Variation over Time and Publication Channels

The text type of job ads has undergone noticeable change over the last 65 years. As Figure 2 illustrates, job ad texts have grown substantially, from around 30 tokens on average in 1950 up to over 150 tokens in 2014. In particular, the job description and the description of the company increase. This reflects a functional change of the job ad, which has become over time more and more an instrument of corporate communications (see e.g. Salvisberg 2010a). Secondly, job task descriptions grows, as job tasks differentiate over time (see e.g. Salvisberg 2010b), and at the same time gain in importance for attracting employees (see e.g. Gnehm 2012). In addition, the amount of text referring to required skills (Z70 and 780) increases strongly, which correlates with an increase in required qualifications in the labor market during this time period (see e.g. Salvisberg 2010b). Proportionally, the reason of a vacancy (Z20), incentives (Z50), description of a job agency (Z40), and residual text (Z30), become less important.

![Number of tokens per text zone 1950-2014 (press, moving averages over 3 years)](image)

Whereas ads from different publication channels differ strongly in overall text length, the distribution of text over zones is quite similar. Online job ads with 240 tokens on average are clearly longer than job ads in the press with 150 tokens (in 2010/2011). This difference might be explained by higher costs for job ads published in print media. Figure 3 illustrates that the distribution of tokens over text zones by contrast differs only slightly between the channels: Text zone Z70, which refers to required hard skills, is longer in online publication channels, especially in online job portals. This corresponds with the fact that vacancies with high skill requirements get published more likely in online channels (Buchs & Sacchi, 2011). Furthermore, ads from online portals contain on average somewhat less information on the job description and on the company itself, but more often information on employment agencies. This is due to the fact that the share of ads published by employment agencies – with the company and details of the job task often remaining unknown – is the highest in online job portals (whereas this kind of job ad
by its very nature hardly ever appears on corporate websites) (ibid.). These findings might indicate that, while there are some minor media-specific differences in job ads, the general structure and function of job ads is the same in all publication channels.

Figure 3: Share of text zones per publication channel (2010-2011)

3.1.4 Distribution of Vocabulary over Text Zones

The distribution of types and tokens over the different text zones provides insights into the difficulties of the classification task as a whole as well as for particular text zones. Strikingly, in zone Z10 (description of the company) the share of types exceeds the share of tokens by far (see Table 2). On average, every type in Z10 appears no more than twice in job ad texts. This relates to the fact that in Z10 a large part of the types are company (or product) names, which occur infrequently as a matter of course. In contrast, for zone Z30 (administration & residual text) the share of tokens is much higher than the share of types. A large part of the information provided in this zone is not of primary interest to the SJMM, and is therefore replaced by substitute elements like [tel] for phone numbers or [kper] for contact person. This procedure reduces the number of different types substantially. The variation of the vocabulary differs thus considerably among the classes.

In addition, as mentioned above, the distribution of tokens over the text zone classes is quite uneven (see Table 2): The two largest zones, Z60 and Z30 comprise around 30% of the tokens each, followed by Z10 (roughly 15%), and Z70 und Z80 (around 10% each). The three classes Z20, Z40 and Z50 on the other hand have very little evidence with frequencies lower than 5%. A baseline classifier that assigns every token the majority class Z60 reaches an accuracy of 30.5%. In other words, with this baseline accuracy, every third token would be classified correctly.
Table 2: Distribution of tokens and types over text zones

Table 3 provides further insights into class ambiguity of tokens and types. In fact, 77% of the types occur only in one single class. But these unambiguous types represent only 7% of all tokens, i.e. these types occur only twice on average. As mentioned above, a large part of these unambiguous types are names. At the other end of the spectrum are 0.2% of the types that occur in all eight text zones, and which sum up to more than half of the tokens. The largest parts of these types are function words, which (by definition) can appear in all text zones and usually adopt class affiliation from neighbored tokens. However, the remaining 40% of the tokens have been seen in more than one class as well. This implies that the classifier cannot rely on token information only. Instead, it is necessary to consider context information to assign zones to tokens.

Table 3: Class ambiguity of tokens and types

3.1.5 Vocabulary in Text Zones over Time

Figure 4 shows word clouds for the most interesting text zones, description of the company (Z10) and the job (Z60), required hard skills (Z70) and soft skills (Z80). As the figure shows the 25 most frequent
words per zone from 1950-1989 (on the left), and from 1989-2014 (on the right), we can identify some changes in vocabulary over time, too. For instance, "international" and "weltweit" (worldwide) pop up in the word cloud for Z10 in the second period, signaling increasing globalization of markets over time. Furthermore, "familie" (family) is no longer under top 25 words for the more recent time period, instead we find the term "gruppe" (holding), which points to the fact that companies grow over time and show more professional corporate structures. In word clouds for job description (Z60), we can further observe developments regarding working conditions: In the more recent time period, jobs are less often referred to as permanent ("dauerstelle" is no longer part of the top 25 words), and mentions of regulated working hours ("geregelt", "arbeitszeit", "freizeit") are less frequent, too.

Over time, level and type of required qualifications change as well: the frequency of "fh" (university of applied sciences) and "weiterbildung" (advanced training) increases, whereas "lehre" (apprenticeship) occurs less frequent. Also, "maschinenschreiben" (typewriting) is no longer under top 25 words during the second period, which is clearly linked to technological change. When it comes to soft skills, employers most often looked for friendly ("freundliche"), reliable ("zuverlässig") and capable ("tüchtig", "tüchtige") employees in the first period and for flexible ("flexibilität"), stress-resistant ("belastbar") and team-minded ("teamfähigkeit") personalities in the second period. While naming just a few examples here, it becomes clear that, during the time period covered, profound changes of the labor market took place, and these developments are reflected in noticeably change in job advertisement vocabulary (for a more detailed discussion of labor market developments, see e.g. Salvisberg 2010b).
Figure 4: Top 25 words for company & job description, hard & soft skills (up to and after 1989). Word clouds constructed with Voyant Tools (Sinclair & Rockwell 2016).
3.1.6 Preprocessing and Encoding of Job Ads

The SJMM corpus is stored in XML format. Each job ad represents one XML element, with the year of publication, the publication channel and the language of the job ad text stored as attributes. Job ad texts are tokenized by the TreeTagger (Schmid 1995), and every single token is represented as a subelement of the job ad. The PoS tag and the lemma - both annotated by the TreeTagger as well - and information on the text zone of the token are added as attributes to the element. To serve as input for the machine learning task, every job ad has to be represented as a sequence of elements. To this end, every job ad is converted to a single line in a text file, where each line is of the form "word1/zonetag1 word2/zonetag2 word3/zonetag3 ...".

3.2 Methods

3.2.1 Sequence Labeling with Bidirectional Long Short-Term Memory Networks (BiLSTMs)

As the previous chapters pointed out, we model automatic text zoning as a sequence labeling task (see Graves 2012, p. 1 for further discussion). In other words, we tag sequences of input data (tokens) with sequences of labels (zone tags). The term sequence labeling (instead of pattern classification) refers to the basic assumption that neither single tokens nor single tags represent independent individual data points. On the contrary, the current token and the current zone tag strongly depend on preceding and subsequent tokens or tags, respectively. Since we have a large amount of labeled data at hand (i.e. sequences of tokens, each labeled with its zone tag), we tackle the task by supervised machine learning, and that is, we use our labeled data to train a model.

In Section 2, we presented several approaches with classic algorithms to solve sequence labeling tasks (as CRFs). In more recent research on similar sequence labeling NLP tasks, recurrent neural networks (RNNs) have proven to be very effective. RNNs are a subclass of artificial neural networks that imitate the cyclical connectivity of neurons in the brain and use iterative functions to store information. The main advantage of RNNs for sequence labeling is that they are flexible in the use of context information: They recognize well which information to store and which to forget. This enables RNNs to detect patterns reliably even when data is noisy (Graves 2012, p. 1).

Standard RNNs have further been improved by two redesigns for Sequence Labeling. Firstly, so-called Long Short-Term Memories (LSTMs) are better in integrating information over long sequences, by using so-called gates (input, output, and forget gates) and "memory cell" units. A second improvement are bidirectional LSTMs (BiLSTMs): Standard RNNs process context information only in one direction (typically forward), which makes sense if it is time data. For all other sequence labeling tasks, it is usually helpful to consider context on both sides of the actual label, as it is done by bidirectional RNNs. Bidirectional LSTMs unify these two advantages, and seem therefore particularly suitable for the task at hand. In BiLSTMs, the left and the right context of the current item will be used as evidence for the decision.
3.2.2 Feature Engineering

In (deep) learning with neural networks, application-specific model engineering with rather simple features often replaces traditional application-specific feature engineering (Graves 2012, p.12; Collobert et al. 2011, p. 2499). Features are represented as vectors, and these feature vectors (or combinations of feature vectors) are input to the model (Collobert et al. 2011, p. 2501). In this study, words – or more precisely tokens – represented as dense numeric vectors are the only input to the model. Fundamental assumption behind this is the distributional hypothesis: Similar words tend to occur in similar context, that is, with similar neighbors (see e.g. Goldberg 2017, p. 116; Potts 2013). Starting point is basically a vector, with the number of dimensions corresponding to the size of the vocabulary, for every word. Then, by reducing the vector dimensionality, word embeddings (lower dimensional vectors) are learned. The basic assumption still holds true, hence similar words tend to show similar vectors. Word embeddings as input to neural networks can either be pretrained, that is, dimensionality reduction takes place outside of the model as unsupervised (with respect to the task) pretraining, or, dimensionality reduction is done within the model optimization process itself with regard to the task, leading to task-specific embeddings. Both approaches will be tested in the experiments.

3.2.3 Pretrained Word Embeddings as Features

To use pretrained word embeddings corresponds to learning a good representation of words on a large unlabeled dataset (i.e. unsupervised with respect to the actual labeling task) as a first step, and subsequently using the resulting representation for a supervised task, usually with a smaller set of labeled data. Unsupervised pretraining of embeddings is supposed to help in two different ways. Firstly, initial values of parameters (i.e. values of pretrained vectors vs. values of randomly initialized vectors), can influence model regularization. Secondly, learning about the input distribution – and thereby revealing factors that cause variation in input data – facilitates learning about mapping from input to output data. This approach is supposed to be particularly efficient when the initial representation is not very informative, as it is the case for words represented by one-hot vectors, where every two distinct words show exactly the same distance from each other (Goodfellow et al. 2016, p. 524 ff.).

In this study, we use Conceptnet Numberbatch embeddings (Speer & Chin 2016; abbreviated with Conceptnet embeddings in the following) as pretrained word embeddings. Conceptnet embeddings take as input embeddings produced by word2vec (Mikolov et al. 2013) and GloVe (Pennington et al. 2014), two of the most popular word embedding so far. By combining them with structured knowledge from two semantic networks (ConceptNet and PPDB, the paraphrase database), they achieve measurably higher performance in word-similarity evaluations than any previous known system. Conceptnet comprise almost 130,000 German word embeddings, each represented as a vector with 300 dimensions. A reasonable coverage of the vocabulary of the SJMM corpus by the pretrained embeddings is crucial for using them as input in neural networks. We conducted the following processing steps to reach this objective:
1. Starting point: looking up German Conceptnet embeddings for (lowercased) lemmas (TreeTagger (Schmid 1995) was used for lemmatization)

2. Improving preprocessing: improving tokenization by stripping trailing or leading special characters; flexible regular expression lookup (for vowel mutations, sharp S, hyphens etc.) for Conceptnet embeddings

3. Improving lemmatization: using GERTWOL lemmatizer (Haapalainen & Majorin 1994) for words without a TreeTagger lemma

4. Semantic reduction of compound words: Searching the Conceptnet embedding for the head, if there is no embedding for the whole compound word, respectively the embedding for the largest part, if there is more than one modifier ("vermittler" instead of "personalvermittler", "dienstleistungsunternehmen" instead of "mediendienstleistungsunternehmen")

5. Replacing abbreviations and words with spelling variants with the corresponding Conceptnet embedding (e.g. "techn." replaced with "technisch", "cirka" replaced with "circa")

Table 4 presents coverage for lemma tokens and types of the SJMM corpus vocabulary. The first two columns refer to all lemmas, the last two columns show figures for cleaned vocabulary, i.e. lemmas without numeral, punctuation and special characters. Without any preprocessing, a Conceptnet embedding is available only for 16% of all lemma types, in other words, coverage is rather low. The processing steps described above increase coverage quite strongly, and in the end we find a corresponding Conceptnet embedding for more than half of the lemma types. Semantic reduction of compounds gives the most significant improvement, raising coverage of types by approximately 30 percentage points and coverage of tokens by approximately 5 percentage points. Compared to this, all other measures are rather ineffective. After all, for three out of four tokens (respectively for 92% of cleaned tokens) a pretrained embedding is available. This coverage seems to be sufficient for using the embeddings as input to the neural network.

<table>
<thead>
<tr>
<th>processing steps</th>
<th>share of all types</th>
<th>share of all tokens</th>
<th>share of cleaned types</th>
<th>share of cleaned tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. initial lookup</td>
<td>16.0%</td>
<td>68.4%</td>
<td>16.0%</td>
<td>83.6%</td>
</tr>
<tr>
<td>2. improved preprocessing</td>
<td>19.2%</td>
<td>70.2%</td>
<td>19.3%</td>
<td>85.8%</td>
</tr>
<tr>
<td>3. GERTWOL lemmatization</td>
<td>20.3%</td>
<td>70.4%</td>
<td>20.4%</td>
<td>86.1%</td>
</tr>
<tr>
<td>4. semantic reduction of compounds</td>
<td>51.7%</td>
<td>75.3%</td>
<td>51.9%</td>
<td>92.1%</td>
</tr>
<tr>
<td>5. replacements</td>
<td>51.9%</td>
<td>75.6%</td>
<td>52.1%</td>
<td>92.6%</td>
</tr>
</tbody>
</table>

Table 4: Coverage of SJMM vocabulary by Conceptnet embeddings
When using pretrained word embeddings as input to a neural network, we can choose if we want to fine-tune them for the task. That is, we keep the pretrained embedding vectors fixed. By doing so, we keep the learned generalization as it is and forego adapting them for the given task. During training, the model then basically learns how to weight embedding vectors optimally. Or, we treat the embedding vectors as model parameters and optimize them with the rest of the model in order to solve the task. A disadvantage of this approach is that only vectors of words that appear in the training set change, while embeddings for unseen words stay the same. This might result in a poorer generalization and a larger test set error (see Goldberg 2017, p. 117). In our experiments, we also test to what extent this choice influences model performance.

3.2.4 Task-Specific Embeddings

An ideal representation of the input data for modeling unveils the underlying factors that generated variation in the data, but especially those factors that are relevant for solving the task. When embeddings are learned task-specifically, that is, as part of the model parametrization, we have a direct clue to find an ideal representation, since we know for each token x the label y that directly specifies the value of the variation of interest. In that sense, to learn task-specific embeddings is part of the supervised learning. In deep neural networks, usually the last layer is a classifier, and the rest of the network learns to provide a helpful representation to this classifier (Goodfellow et al. 2016, p. 525 and p. 552). This approach is especially promising with a relatively large amount of labeled data available (Goldberg 2017, p. 115). Since the SJMM corpus comprises around 38,000 labeled job ads, we train task-specific embeddings in our experiments and compare model performance to the approach with pretrained word embeddings.

Low-frequency words are an issue when learning task-specific representations, since the model cannot generalize well when examples for learning are sparse. One way to deal with this problem is to model one and the same embedding for all words with very low frequency (and subsequently use this embedding for words in the test set that were not part of the training material as well). Another way to tackle this is to use character-level representations for low-frequency words, as in the approach by Neubig et al. (2017): For rare words, i.e. words which occur five times at most in the training material, the training process learns an embedding for its characters. In doing so, as there is much more data on this level, the model can generalize better. Likewise, when using pretrained word embeddings, we can learn character-level representations for words in the training material that are not covered by pretrained word embeddings. In conclusion, character-based embeddings seem to provide reasonable representations for low-frequency words, as they neither rely on very few examples nor lump together all rare words. Furthermore, this approach also solves the unknown word problem: With this representation, we are able to provide an embedding for every word in the test set, regardless of whether the word was part of the training data or not (Goldberg 2017, p. 131).
3.2.5 BiLSTMs Architecture and Toolkit

Bidirectional LSTMs are implemented with the DyNet 2.0 toolkit (Neubig et al. 2017). As a first step, the structure for the parameter collection is set up, a trainer is defined and parameters are added to the collection. Then, in order to train the model, DyNet performs the following steps for each training example:

a. create a computation graph,

b. run forward to compute the model hypothesis and compute the loss, which represents the deviation between true labels and the one predicted by the current model,

c. run backward to compute the gradients,

d. use the trainer to optimize the parameters of the model with respect to the latest gradients.

The DyNet toolkit works with dynamically changing network structures in two ways: Firstly, it creates a new computation graph for each training example (so, training is not done in batches of training examples), and secondly, the concrete form of the computation graph depends on the input that is the features of each specific training example.

As our task is similar to PoS tagging, we use the BiLSTM PoS tagger by Huang et al. (2015) as starting point for our experiments. The classifier uses bidirectional token-level LSTMs (as described above) to encode the input. On top of the states of the bidirectional word-level LSTMs comes a Multilayer Perceptron (MLP) with one hidden layer. That is, for every token, the resulting output vectors from the forward LSTM and the backward LSTM are concatenated and then fed as input to an MLP (see Figure 5).

As described above, the use of character-based representation for low-frequency tokens potentially improves model performance (Neubig et al. 2017). Hence, for every token that occurs not more than five times in the training data (or, for which no pretrained embedding is available), the training process optimizes an embedding over the characters. As with the token-level LSTMs, one forward and one backward LSTM encode the input. The concatenated output vectors of the character-level LSTMs serve then as input for the token-level BiLSTMs (see Figure 6).

The number of layers for word-level and char-level LSTMs, the dimensionality of vectors to represent token embeddings and character embeddings, as well as the number of nodes in the hidden layers are the hyperparameters in this architecture that can be varied to alter model capacity. For a formalization of this architecture, see Huang et al. (2015).

---

5 We based our experiments on a Python implementation for this PoS tagger, which can be found here: https://github.com/clab/dynet_tutorial_examples/blob/master/tutorial_bilstm_tagger.py
Figure 5: Token-level BiLSTMs. Reprinted from Dyer et al. (2016, p. 4).

Figure 6: Character-level BiLSTMs. Reprinted from Dyer et al. (2016, p. 29).
4 Experiments & Results

4.1 Experiments

For all experiments, the machine learning setup is to split the annotated data into a training set (80%), a development set (10%), and a final test set (10%). Models are trained in 50 iterations (30 iterations in some experiments, respectively). Thus, we do not apply an early stopping criterion, but keep the model that reaches highest accuracy on the development set over all iterations. More thorough evaluation (such as precision and recall, or evaluation on final test set) is in each case based on this best performing model.

A first series of experiments analyzes whether word embeddings that are learned with the training material perform better than pretrained word embeddings. This relates to the question of how domain-specific the vocabulary of the SJMM corpus is. That is, firstly, which share of the vocabulary is covered by pretrained general word embeddings, and secondly, are there domain-specific differences in the semantic meaning of words? Furthermore, to train embeddings with the model itself implies a task-specific tuning of the word embeddings. This might be promising, especially considering the large amount of available training data. In this light, the impact of an increasing number of training instances on the learning curves will be examined as well. Besides, we will evaluate the effect of using character-based representation on model performance.

A second series of experiments deals with the question, whether changing the hyperparameters can increase model performance. Performance usually rises with model capacity (see e.g. Collobert et al. 2011). Since this leads to even more computationally expensive and time-consuming training, the hyperparameter settings are not tested exhaustively. Our experiments focus on augmenting the number of layers and the number of input vector dimensions to character-level and word-level BiLSTMs.

Thirdly, the main objective is to build a model that is optimized for the present-day or future application, in other words, to segment job ads that are published from 2015 onwards. As Section 3.1 demonstrated, job ads as a text type have undergone major changes in the past 65 years. This holds for length and vocabulary of the ad texts as a whole as well as for the share of specific text zones. Thus, it is probable that older training data lower the performance of a model aimed at segmenting present-day job ads. On the other hand, excluding older job ads results in a smaller amount of training data. The trade-off between these two factors, size and up-to-datedness of the training data, will be tested on a subset of current job ads.
The best model derived from these three series of experiments is then evaluated in more detail. Model performance is in all experiments assessed on the level of tokens. We evaluate precision and recall for each text zone class, and report overall accuracy as defined below (see e.g. Carstensen et al. 2010, p. 154f.):

\[
\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}
\]

\[
\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}
\]

\[
\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Positives} + \text{Negatives}}
\]

Additionally, confusion matrices as well as qualitative analysis of errors might point to further model improvements. Specific evaluations will show if errors are related to text length and publication channels of the job ads. As these two variables are available prior to automatic segmentation in the information extraction workflow, it would be easy to build specifically adapted models.

At the very end, the automatic text zoning of job ads is optimized by an ensembling technique (see Rokach 2010 for an overview): Given the finally chosen model setting, five classifiers with differently initialized parameters will be trained. As they start from different points in the hypothesis space, the classifiers will differ slightly from each other (Collobert 2011, p. 2521; Rokach 2010, p. 21). The test set will be tagged by all five models, keeping for every token the majority vote of the five modes as final output. This procedure allows calculating variance in model performance as well as Inter-Annotator Agreement (IAA, e.g. Fleiss’ kappa) of the models. Moreover, especially if IAA is not very high, we can expect an increase in model performance. This will be evaluated by usual measures, precision, recall and overall accuracy. Regarding the downstream processing steps in real-world application, ensembling brings an additional benefit: Low Inter-Model Agreement in classification indicates a difficult task and an uncertain classification. Text zones with low agreement can be identified and passed on to manual verification, in order to further increase accuracy.
4.2 Results

4.2.1 Initial Neural Network Architecture

The initial model with the same neural network architecture previously used by Huang et al. (2015), as described in Section 3.2.5, reaches an accuracy of 89.0% on token level. The setting of hyperparameters in this first experiment is as following: We use one hidden layer for word-level and character-level BiLSTMs each, where tokens are represented with an input vector of 128 dimensions and output vector of 100 dimension (output vector of forward and backwards LSTMs with 50 dimensions each get concatenated). Character-level BiLSTMs are applied when a token appears less than six times in the training material. Characters are represented with 20-dimensional input vectors. Output vectors of forwards and backwards LSTMs comprise 64 dimensions each, the concatenation of 128 dimensions then is handed over as input to token-level BiLSTMs. MLPs on top of the BiLSTMs reduce 100-dimensional output vectors of BiLSTMs to 32 dimensions and subsequently to 8 final dimensions representing the classification tags. With this first model, we achieve already higher accuracy than in previous approaches with CRFs (Gnehm 2016).

4.2.2 Task-Specific versus Pretrained Embeddings

The use of Conceptnet embeddings does not improve model performance, as can be seen from the results in Table 5. Keeping the setting the same as in the initial model (i.e. applying the char-level LSTMs for lemmas without a Conceptnet embedding), accuracy is clearly lower when using pretrained embeddings (86.5% vs. 89.0%). Allowing the pretrained embeddings to be adapted during training brings a substantial gain in performance, though accuracy is still slightly lower than in the initial model (88.9% vs. 89.0%). These results indicate that a task-specific training or adaption of embeddings is fruitful for job ad segmentation given the amount of available training data.

<table>
<thead>
<tr>
<th>model</th>
<th>maximum accuracy</th>
<th>mean accuracy (it. 6-50)</th>
</tr>
</thead>
<tbody>
<tr>
<td>task-specific embeddings</td>
<td>89.0%</td>
<td>88.8%</td>
</tr>
<tr>
<td>pretrained embeddings, without fine-tuning</td>
<td>86.5%</td>
<td>85.5%</td>
</tr>
<tr>
<td>pretrained embeddings, with fine-tuning</td>
<td>88.9%</td>
<td>88.7%</td>
</tr>
</tbody>
</table>

Table 5: Development set accuracy of models with task-specific and pretrained embeddings

Overall, these finding suggest we can learn a good representation for the vocabulary in the SJMM corpus with the amount of available in-domain training material. The experiments reveal that task-specific embeddings help slightly more than pretrained general embeddings, even though the latter are built from much more data. Maybe the use of more comprehensive or language-specific pretrained general

---

6 Results of the first five training iterations are not included in mean accuracy, since random initialization of parameters leads to big gaps in model performance at the beginning of training iterations.
word embeddings would have led to better results. In addition, improving lemmatization might improve coverage of the SMM vocabulary with Conceptnet embeddings as well, and thus raise performance of pretrained embeddings. Another direction for future work could be to pretrain domain-specific word embeddings with more non-labeled data, i.e. on a very large corpus of job openings without manual segmentation annotation. However, based on the results presented, continuing experiments without pretrained word embeddings (and time-consuming preprocessing) and instead training word embeddings as part of the model parametrization is recommended.

4.2.3 Character-Level Representation

Character-level embeddings are helpful, as for all the three models performance is lower, if we use one and the same embedding for all unknown or rare words instead. With task-specific embeddings, we reach a slightly lower accuracy of 88.7% (vs. 89.0%), if we go without character-level representations. With pretrained (fine-tuned during training) embeddings, we reach at most an accuracy of 84.8% without the character-based BiLSTMs (vs. 88.9%). The effect is much stronger for the model with pretrained embeddings, as the character-level BiLSTM in this case is applied for the quarter of tokens without a pretrained embeddings, whereas in the model with task-specific embeddings, only around 6% of the tokens appear no more than five times in the training material and hence are represented at character-level. In conclusion, to use representations on character-level for rare or unknown words is beneficial.

4.2.4 Effects of Training Set Size

Furthermore, experiments with the size of the training set show that the amount of labeled data is large enough to learn a robust model. Keeping the model setting as described above, we start the experiment with a training set of 250 job ads, and then increase the amount of training data in several steps, finally using the complete training set with more than 29,000 job ads. As can be seen from Figure 7, the learning curve levels off strongly as the training set size increases. While the first step improves accuracy by more than 10 percentage points, the last step raises accuracy only by 0.1 percentage point. These findings imply that the model would not benefit considerably from a larger amount of training data in general.

---

7 The most positive effect of character-based embeddings (accuracy of 86.5% vs. 79.6%) is obviously observed in the model with pretrained word embeddings that are not adapted during training process: In this model setting, character-based embeddings are the only way to fine-tune the representation of input data to the task.
Figure 7: Effects of training size on development set accuracy

The effect of the training set size is somewhat smaller for pretrained embeddings than for task-specific representations. Precisely because vectors of Conceptnet embeddings are pretrained – instead of a randomly initialized at the beginning of the training process – they perform better when using comparatively little amounts of training data. But subsequently, task-specific embeddings are able to profit more from additional training material than pretrained embeddings. For pretrained embeddings without adaption, the learning curve shows a similar shape, but on a remarkably lower level. Since the representations themselves are fixed, training material can only be used to adjust the weighting of the representations in the model. Therefore, even with a huge amount of training data, this model does not reach the accuracy of models that adapt semantic representations of the input data to the task. In conclusion, these results suggest that learning task-specific representations here is mainly successful in combination with the large amount of labeled training data.
4.2.5 Exploring Hyperparameter Settings

Experiments with varied hyperparameter settings show slight improvements for some models. Adding a second hidden layer to the token-level BiLSTMs raises the maximum accuracy up to 89.2% and increases mean accuracy over 50 training iterations slightly as well (see model 2, Table 6.) However, adding more layers barely increases performance, as a model with three hidden layers reaches the same maximum accuracy and only a slightly higher mean accuracy (see model 3). Higher numbers of layers (models with 4 or 5 hidden layers) even result in declining performance, while increasing the training time considerably. Other configurations with increased hyperparameters – bringing more capacity to the model – do not boost performance: Increasing the number of layers for character-level BiLSTMs has no positive effect on accuracy. Also, increasing dimensionality of input vectors does not enhance accuracy; this applies for token-level LSTMs (model 6) as for the character-level LSTMs (model 7). In further experiments, we will therefore continue with model 2 (2 hidden layers on the token-level BiLSTMs), as this is the smallest model architecture yielding noticeable performance improvements.

<table>
<thead>
<tr>
<th>model setting</th>
<th>maximum accuracy</th>
<th>mean accuracy (it. 6-50)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. initial setting</td>
<td>89.0%</td>
<td>88.8%</td>
</tr>
<tr>
<td>2. 2-layer token-level BiLSTM</td>
<td>89.2%</td>
<td>88.9%</td>
</tr>
<tr>
<td>3. 3-layer token-level BiLSTM</td>
<td>89.2%</td>
<td>89.0%</td>
</tr>
<tr>
<td>4. 4-layer token-level BiLSTM</td>
<td>89.2%</td>
<td>88.9%</td>
</tr>
<tr>
<td>5. 5-layer token-level BiLSTM</td>
<td>89.0%</td>
<td>88.6%</td>
</tr>
<tr>
<td>6. 2-layer character-level BiLSTM</td>
<td>89.0%</td>
<td>88.7%</td>
</tr>
<tr>
<td>7. input vector for token embeddings with 300 dimensions</td>
<td>89.0%</td>
<td>88.6%</td>
</tr>
<tr>
<td>8. input vector for character embeddings with 40 dimensions</td>
<td>89.0%</td>
<td>88.7%</td>
</tr>
</tbody>
</table>

Table 6: Development set accuracy for model with different hyperparameter settings

4.2.6 Reaching back to 1950: Is Older Data Still Useful Data?

Are older job ads useful to train a model for present-day and future application? To answer this question, several models using training data from different time periods are evaluated on a development set consisting of 10% of the data from 2010 to 2014. As can be seen from table 7, adding more and older training data slightly improves performance, but this holds only for jobs ads published from 1970 onwards. Going back further in time results in decreasing maximum and mean model performance. These findings indicate that the effect of more data is partly weakened by some sort of out-of-domain effect when including very old job ads. This out-of-domain-effect can be observed although the amount of training data from 1950 to 1970 is considerably smaller than the amount of training data from 1970 onwards. However, using older training data does not lower model performance substantially. Thus, long term change over time in the size, vocabulary and functions of job ads as described in Chapter 3.1 is not a major issue when it comes to text zoning of job ad texts. Moreover, change seems to be slow, suggesting that for a future application, we might not need to manually annotate vast amounts of training material often. This is clearly an encouraging observation. Considering both maximum and mean
accuracy, the model using training data from 1970 onwards seems to find the optimal trade-off between a training set that is up to date and large enough. Hence, this model is selected for present-day and future application.

<table>
<thead>
<tr>
<th>time period</th>
<th>max. accuracy</th>
<th>mean accuracy (it. 6-30)</th>
<th>job ads in training set (rounded to 1,000)</th>
<th>tokens in training set (rounded to 1,000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>as of 2010</td>
<td>89.0%</td>
<td>88.5%</td>
<td>5,000</td>
<td>1,105,000</td>
</tr>
<tr>
<td>as of 2000</td>
<td>89.1%</td>
<td>88.9%</td>
<td>10,000</td>
<td>1,701,000</td>
</tr>
<tr>
<td>as of 1990</td>
<td>89.2%</td>
<td>88.9%</td>
<td>15,000</td>
<td>2,137,000</td>
</tr>
<tr>
<td>as of 1980</td>
<td>89.2%</td>
<td>88.9%</td>
<td>20,000</td>
<td>2,514,000</td>
</tr>
<tr>
<td>as of 1970</td>
<td>89.3%</td>
<td>89.0%</td>
<td>25,000</td>
<td>2,830,000</td>
</tr>
<tr>
<td>as of 1960</td>
<td>89.2%</td>
<td>88.8%</td>
<td>30,000</td>
<td>3,065,000</td>
</tr>
<tr>
<td>as of 1950</td>
<td>89.2%</td>
<td>88.7%</td>
<td>32,000</td>
<td>3,238,000</td>
</tr>
</tbody>
</table>

Table 7: Development set accuracy with training data covering different time periods

4.2.7 Error Analysis on the Development Set

According to the confusion matrix presented in Table 8, the model experiences more or less the same difficulties human annotators reported in personal communication as well. One main problem is to differentiate between description of the company as a whole (Z60) and description of the job itself (Z10). Especially for text parts providing information on the department of the open position, description of infrastructure, or the working environment, human annotators find it difficult to decide. This is also reflected in the fact that the word "team", which can refer to the company as whole as well as to a small working group, is among the most frequent errors. A second problem is to distinguish between job description (Z60) and requirements (Z70 & Z80), as tasks and requirements are often wrapped up in one and the same text passage (e.g. "Sie gehen offen auf unsere Lernenden zu und unterstützen diese mit Ihrem Fachwissen jederzeit kompetent."). Furthermore, it is striking that a comparably large portion of confusion is caused by tokens belonging to residual text (Z30) that are not recognized. This is due to the fact that human annotators did not split up sentences at the same level of granularity very consistently, (e.g. "Z70 Sie bringen mit: Eine Ausbildung als..." vs. "Z30 Sie bringen mit: Z70 Eine Ausbildung als..."). An analysis of the most frequent misclassified words further reveals that especially function words (e.g. prepositions, pronouns or conjunctions) are often wrongly classified. This comes to no surprise as they occur (according to their very nature) in every single text zone. Regarding these last two points, classification errors are not a major issue for subsequent information extraction, as the affected text parts are not of principal interest. To sum up, these findings on the classification on the level of tokens

---

*Previous experiments have shown, that highest accuracy is typically reached around iteration 20, and model performance does barely change after iteration 30. Therefore, for all experiments described below, the number of training iterations is set to 30. By doing so, we can also reduce the risk of overfitting the model to the development set.*
and specific text zones make clear that automatic text segmentation does not suffer unexpected or inexplicable problems.

<table>
<thead>
<tr>
<th></th>
<th>Z10</th>
<th>Z20</th>
<th>Z30</th>
<th>Z40</th>
<th>Z50</th>
<th>A60</th>
<th>Z70</th>
<th>Z80</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>truth</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z10</td>
<td>15.2%</td>
<td>0.0%</td>
<td>0.5%</td>
<td>0.1%</td>
<td>0.0%</td>
<td>1.0%</td>
<td>0.0%</td>
<td>0.1%</td>
</tr>
<tr>
<td>Z20</td>
<td>0.1%</td>
<td>0.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Z30</td>
<td>0.6%</td>
<td>0.0%</td>
<td>23.4%</td>
<td>0.1%</td>
<td>0.0%</td>
<td>1.2%</td>
<td>0.6%</td>
<td>0.3%</td>
</tr>
<tr>
<td>Z40</td>
<td>0.1%</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.5%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Z50</td>
<td>0.1%</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.0%</td>
<td>1.0%</td>
<td>0.2%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Z60</td>
<td>1.5%</td>
<td>0.0%</td>
<td>0.6%</td>
<td>0.0%</td>
<td>0.1%</td>
<td>29.4%</td>
<td>0.3%</td>
<td>0.6%</td>
</tr>
<tr>
<td>Z70</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.6%</td>
<td>11.1%</td>
<td>0.4%</td>
</tr>
<tr>
<td>Z80</td>
<td>0.1%</td>
<td>0.0%</td>
<td>0.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.8%</td>
<td>0.3%</td>
<td>8.1%</td>
</tr>
</tbody>
</table>

Cell values: shares of tokens (total of all cells = 100.0%)

Table 8: Confusion matrix of classification on the development set

More detailed examinations on the level of job ads do not reveal high potential for optimization as well. As Figure 8 shows, the frequency distribution of the error rate per ad is positively skewed. This is good news, since the largest part of job ads shows error rates lower or around 10%, and only a small proportion shows error rates high above average. A closer look at some ads with high error rates reveals that at least a part of them are rather untypical job ads. Figure 9 shows an example for a job ad with rather unusual task, resulting in poor classification results particularly for Z60, the description of the job, and a high overall error rate.

![Frequency distribution of error rates per job ad](image)
<table>
<thead>
<tr>
<th>token</th>
<th>gold standard</th>
<th>model prediction</th>
<th>classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gesucht</td>
<td>Z30</td>
<td>Z30</td>
<td>✓</td>
</tr>
<tr>
<td>in</td>
<td>Z10</td>
<td>Z60</td>
<td>×</td>
</tr>
<tr>
<td>Küttigen</td>
<td>Z10</td>
<td>Z60</td>
<td>×</td>
</tr>
<tr>
<td>:</td>
<td>Z10</td>
<td>Z10</td>
<td>✓</td>
</tr>
<tr>
<td>Zuverlässigen</td>
<td>Z80</td>
<td>Z60</td>
<td>×</td>
</tr>
<tr>
<td>,</td>
<td>Z80</td>
<td>Z80</td>
<td>✓</td>
</tr>
<tr>
<td>liebevollen</td>
<td>Z80</td>
<td>Z60</td>
<td>×</td>
</tr>
<tr>
<td>Hütedienst</td>
<td>Z60</td>
<td>Z60</td>
<td>✓</td>
</tr>
<tr>
<td>,</td>
<td>Z60</td>
<td>Z60</td>
<td>✓</td>
</tr>
<tr>
<td>stundenweise</td>
<td>Z60</td>
<td>Z60</td>
<td>✓</td>
</tr>
<tr>
<td>oder</td>
<td>Z60</td>
<td>Z60</td>
<td>✓</td>
</tr>
<tr>
<td>tageweise</td>
<td>Z60</td>
<td>Z60</td>
<td>✓</td>
</tr>
<tr>
<td>,</td>
<td>Z60</td>
<td>Z60</td>
<td>✓</td>
</tr>
<tr>
<td>Für</td>
<td>Z60</td>
<td>Z10</td>
<td>×</td>
</tr>
<tr>
<td>Unseren</td>
<td>Z60</td>
<td>Z10</td>
<td>×</td>
</tr>
<tr>
<td>gut</td>
<td>Z60</td>
<td>Z10</td>
<td>×</td>
</tr>
<tr>
<td>erzogenen</td>
<td>Z60</td>
<td>Z10</td>
<td>×</td>
</tr>
<tr>
<td>,</td>
<td>Z60</td>
<td>Z10</td>
<td>×</td>
</tr>
<tr>
<td>Dreijährigen</td>
<td>Z60</td>
<td>Z10</td>
<td>×</td>
</tr>
<tr>
<td>Labradorrüden</td>
<td>Z60</td>
<td>Z10</td>
<td>×</td>
</tr>
<tr>
<td>.</td>
<td>Z60</td>
<td>Z10</td>
<td>×</td>
</tr>
<tr>
<td>Entlöhnung</td>
<td>Z50</td>
<td>Z50</td>
<td>✓</td>
</tr>
<tr>
<td>nach</td>
<td>Z50</td>
<td>Z50</td>
<td>✓</td>
</tr>
<tr>
<td>Absprache</td>
<td>Z50</td>
<td>Z50</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Figure 9:** Example of a job ad showing a high error rate
The error rate is not clearly related to text length, as can be seen from the scatterplot in Figure 10. Automatic segmentation works rather well for the largest part of ads, be it for very short ads or be it for exceptionally long ads. Hence, there is no reason to exclude very short or very long ads from automatic segmentation or to build specific classifiers for ads of different text length.

![Scatterplot of error rate vs. text length](image)

**Figure 10: Relationship between error rate and text length**

An analysis of error rates per publication channel reveals that classification works somewhat better for job ads published in the press and on company websites (mean error rate per ad 10.5% each) than for ads published on online job portals (mean error rate per ad 12.5%). This might be explained by the composition of the training material: Ads from the press make up for the largest share of the training set by far (over 85%), whereas ads from online portals are underrepresented (around 5%) compared to the development set. Ads from company websites are more frequent than ads from online portals (around 8%). Furthermore, there is probably relatively little variation in data from company websites: As there is often a number of job ads sampled from the same company website, different job ads can show similar or nearly identical text parts (e.g. the description of the company itself). Therefore, since for some channels only little – or little variational – training material is available, building channel-specific classifiers might lead to overfitting to training data. Considering the rather small differences in performance between publication channels as well, we can go with one and the same general and robust model for ads of all different publication channels.

---

9 This finding holds true in channel-specific analysis as well.
4.2.8 Ensembling and Evaluation on the Final Test Set

The findings presented above lead to several important insights for building an optimal model: First, task-specific embeddings perform better than the pretrained Conceptnet word embeddings, given the amount of training data available. Second, hyperparameters are best set to two layers for word-level LSTMs, keeping all other hyperparameters as in the initial setting. Third, for a future real-world application on current job ads, it is best to go without training data older than 1970. Fourth, detailed analyzes show that it is advisable to use one single model for all job ads, regardless of length or publication channel of job ads. Considering all this, a single model achieves 89.3% accuracy on the development set containing job ads from 2000 to 2014.

<table>
<thead>
<tr>
<th></th>
<th>development set</th>
<th>test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>accuracy model 1</td>
<td>89.4%</td>
<td>89.3%</td>
</tr>
<tr>
<td>accuracy model 2</td>
<td>89.1%</td>
<td>89.2%</td>
</tr>
<tr>
<td>accuracy model 3</td>
<td>89.1%</td>
<td>89.3%</td>
</tr>
<tr>
<td>accuracy model 4</td>
<td>89.2%</td>
<td>89.2%</td>
</tr>
<tr>
<td>accuracy model 5</td>
<td>89.2%</td>
<td>89.2%</td>
</tr>
<tr>
<td>accuracy with ensembling</td>
<td>89.9%</td>
<td>89.8%</td>
</tr>
<tr>
<td>variance model 1-5</td>
<td>0.016</td>
<td>0.004</td>
</tr>
<tr>
<td>Fleiss’ kappa</td>
<td>0.924</td>
<td>0.923</td>
</tr>
</tbody>
</table>

Table 9: Evaluation of single models and ensembling on development set and test set

By using an ensemble of classifiers, model performance further increases. Given the model specification described above, we trained five classifiers with different (random) initial parameters. As can be seen from Table 9, the classifiers vary only slightly in accuracy (variance = 0.016 on the development set; 0.004 on the test set). Also, the tiny differences in results on development and test set make clear that models do not suffer from a development set overfit. All in all, the models seem to generalize well and converge consistently to a solution. Furthermore, Inter-Annotator Agreement between the five models is very high, as Fleiss' kappa values over 0.92 for both development and test set indicate. According to Landis & Koch (1977), values between 0.6 and 0.8 represent substantial agreement; values above 0.8 speak for almost perfect agreement. In other words, for the vast majority of classifications, the five models find the same solution. Nonetheless, to combine the classifiers improves model performance: Using the majority vote of the five models as the final classification increases accuracy by nearly 0.5 percentage points compared to the best single model, for the development set and for the test set.

Besides, ensembling brings an additional benefit for real-world applications, as disagreement between the models can support human quality control. One way to do so is to identify tokens without a majority vote (this applies to around 0.5% of the tokens in the development set and in the test set) and to correct them manually. For the present case, this kind of post-processing would raise accuracy above 90%. A manual inspection of errors reveals that not all deviations from gold standard are problematic. We observe even some cases where the model predicts correct solutions, while the gold standard tags are
incorrect. Figure 11 shows an example for a text passage with wrong label predictions that do not pose a problem. Model prediction differs only from gold standard tags regarding the question, how detailed to split text into residual text and other text zones. The predicted tags make sense as well and this labeling will not negatively affect subsequent information extraction. Results of the manual inspection all in all suggest that a considerable part of model predictions that differ from gold standard tags are not serious errors (this holds for the majority of the 20 cases revised). Nonetheless, ensembling allows us to perform quality checks, particularly for text passages that seem to be difficult for automatic text zoning.

<table>
<thead>
<tr>
<th>token</th>
<th>gold standard</th>
<th>model prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Einer</td>
<td>soft skills</td>
<td>soft skills</td>
</tr>
<tr>
<td>Zuverlässig</td>
<td>soft skills</td>
<td>soft skills</td>
</tr>
<tr>
<td>und</td>
<td>soft skills</td>
<td>soft skills</td>
</tr>
<tr>
<td>exakten</td>
<td>soft skills</td>
<td>soft skills</td>
</tr>
<tr>
<td>Persönlichkeit</td>
<td>soft skills</td>
<td>soft skills</td>
</tr>
<tr>
<td>bietet</td>
<td>job description</td>
<td>admin. &amp; residual text</td>
</tr>
<tr>
<td>sich</td>
<td>job description</td>
<td>admin. &amp; residual text</td>
</tr>
<tr>
<td>hier</td>
<td>job description</td>
<td>admin. &amp; residual text</td>
</tr>
<tr>
<td>eine</td>
<td>job description</td>
<td>job description</td>
</tr>
<tr>
<td>spannende</td>
<td>job description</td>
<td>job description</td>
</tr>
<tr>
<td>Aufgabe</td>
<td>job description</td>
<td>job description</td>
</tr>
</tbody>
</table>

Figure 11: Text passage with acceptable differences between predicted and true labels

Classification results for single text zones are encouraging (see Figure 12). Evaluation measures, notably recall, are relatively low for text zones that are not so frequent, i.e. reason of the vacancy (Z20), description of job agencies (Z40) and incentives (Z50). Regarding the most interesting text zones, description of the organization (Z10) and the job (Z60), as well as required hard skills (Z70) and soft skills (Z80), evaluation measures vary roughly around 90%. Just for required soft skills (Z80), recall is somewhat lower (84%). The reason for this latter finding is a question for further investigation. The two low-frequency classes, reason of the vacancy (Z20) and incentives (Z50), show comparatively poor results, firstly, and are not of principal interest to SJMM labor market research, secondly. Therefore, we could merge them (after labeling) with the description of the job (Z60) – which can be justified in terms of content. By merging Z20 and Z60, recall for the resulting new class would rise up to 89.7% and precision up to 92.1%, and accuracy over all classes would increase up to 90.1%. If we merge additionally Z50, we even reach recall of 90.6%, precision of 92.4% and a global accuracy of 90.5%. This post-labeling merging would imply a rather minor information loss, also acceptable considering our main research interests.
Future experiments could also try to increase performance by adding local features to the model. For instance, gazetteers with information on names of employment agencies could help to raise recall for the respective text zone (Z40). Likewise, using gazetteers on job titles, company names, personal names, or locations, we could possibly further improve classification results for other zones as well. But by and large, the results presented suggest that automatic text zoning works sufficiently well, particularly for the specific zones most relevant to SJMM labor market research.

The confusion matrix for the final test (see Table 10) does not reveal major problems, very similar to the confusion matrix for the development presented previously. Compared to the results on the development set, per label accuracy is slightly lower for residual text (Z30) and soft skills (Z80), but higher for all other text zones. The sources of confusion are quite the same as reported for the development set. An analysis of the most frequent misclassified tokens further reveals that function words make up for a considerable part of errors (see e.g. most frequent misclassified tokens in Table 11). Additionally, false negatives regarding the classification of residual text (Z30) are relatively frequent.
As an additional experiment, we tried out Viterbi algorithm for global decoding in order to find the optimal sequence of classification labels. This seemed to be promising, since more than 50% of the tokens can show up in all eight different text zones (see Section 3.1.4), and the two main sources of confusion are closely related to this fact. Global decoding is supposed to smooth out label sequences and adjust tags of ambiguous token to tags of their neighbors. However, using Viterbi decoding does not bring any improvement. This implies that the BiLSTMs by themselves succeed in using context information in order to find appropriate label sequences. In any case, these latter two sources of confusion most likely will not strongly harm further processing steps. All in all, the findings presented suggest that automatic text zoning works well enough to facilitate subsequent information extraction.
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.</td>
<td>(640)</td>
<td>26</td>
<td>Team</td>
<td>(49)</td>
</tr>
<tr>
<td>2</td>
<td>,</td>
<td>(527)</td>
<td>27</td>
<td>!</td>
<td>(46)</td>
</tr>
<tr>
<td>3</td>
<td>Sie</td>
<td>(441)</td>
<td>28</td>
<td>des</td>
<td>(44)</td>
</tr>
<tr>
<td>4</td>
<td>in</td>
<td>(225)</td>
<td>29</td>
<td>haben</td>
<td>(44)</td>
</tr>
<tr>
<td>5</td>
<td>für</td>
<td>(172)</td>
<td>30</td>
<td>uns</td>
<td>(41)</td>
</tr>
<tr>
<td>6</td>
<td>der</td>
<td>(167)</td>
<td>31</td>
<td>Die</td>
<td>(38)</td>
</tr>
<tr>
<td>7</td>
<td>die</td>
<td>(163)</td>
<td>32</td>
<td>einer</td>
<td>(38)</td>
</tr>
<tr>
<td>8</td>
<td>/</td>
<td>(143)</td>
<td>33</td>
<td>arbeiten</td>
<td>(35)</td>
</tr>
<tr>
<td>9</td>
<td>im</td>
<td>(96)</td>
<td>34</td>
<td>Zürich</td>
<td>(34)</td>
</tr>
<tr>
<td>10</td>
<td>über</td>
<td>(94)</td>
<td>35</td>
<td>bieten</td>
<td>(33)</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>(93)</td>
<td>36</td>
<td>Teams</td>
<td>(33)</td>
</tr>
<tr>
<td>12</td>
<td>an</td>
<td>(93)</td>
<td>37</td>
<td>zur</td>
<td>(33)</td>
</tr>
<tr>
<td>13</td>
<td>Ihr</td>
<td>(81)</td>
<td>38</td>
<td>können</td>
<td>(32)</td>
</tr>
<tr>
<td>14</td>
<td>ist</td>
<td>(80)</td>
<td>39</td>
<td>auch</td>
<td>(31)</td>
</tr>
<tr>
<td>15</td>
<td>ein</td>
<td>(79)</td>
<td>40</td>
<td>aus</td>
<td>(31)</td>
</tr>
<tr>
<td>16</td>
<td>Aufgaben</td>
<td>(75)</td>
<td>41</td>
<td>Ihren</td>
<td>(30)</td>
</tr>
<tr>
<td>17</td>
<td>den</td>
<td>(68)</td>
<td>42</td>
<td>am</td>
<td>(26)</td>
</tr>
<tr>
<td>18</td>
<td>einen</td>
<td>(68)</td>
<td>43</td>
<td>bringen</td>
<td>(25)</td>
</tr>
<tr>
<td>19</td>
<td>Vorteil</td>
<td>(63)</td>
<td>44</td>
<td>dem</td>
<td>(25)</td>
</tr>
<tr>
<td>20</td>
<td>Wir</td>
<td>(55)</td>
<td>45</td>
<td>dieser</td>
<td>(25)</td>
</tr>
<tr>
<td>21</td>
<td>als</td>
<td>(54)</td>
<td>46</td>
<td>Kunden</td>
<td>(25)</td>
</tr>
<tr>
<td>22</td>
<td>diese</td>
<td>(53)</td>
<td>47</td>
<td>Tätigkeit</td>
<td>(25)</td>
</tr>
<tr>
<td>23</td>
<td>Ihnen</td>
<td>(53)</td>
<td>48</td>
<td>Arbeiten</td>
<td>(24)</td>
</tr>
<tr>
<td>24</td>
<td>oder</td>
<td>(50)</td>
<td>49</td>
<td>Ihrer</td>
<td>(24)</td>
</tr>
<tr>
<td>25</td>
<td>auf</td>
<td>(49)</td>
<td>50</td>
<td>Aufgabengebiet</td>
<td>(23)</td>
</tr>
</tbody>
</table>

Table 11: The 50 most frequent wrongly classified tokens, frequencies in brackets
5 Conclusion

5.1 Summary

Overall, the results of supervised machine learning with BiLSTMs for the task of text zoning of job ads are convincing. For the SJMM corpus as a whole, accuracy reaches 89.0% with the original hyperparameter setting and 89.2% with a model with an additional hidden layer. Optimizing the model in several steps regarding segmenting job ads from more recent years (2010-2014), we raise accuracy up to 89.8% on the final test set. Compared to the results achieved with the model at the starting point, this is an improvement of almost 1 percentage point, and compared to results of an earlier approach with CRFs (Gnehm 2016) an improvement of more than 2 percentage points. Using a neural approach, we are able to reduce error rate on the token level by 16% compared to the CRF approach. To keep in mind, assigning every token the majority class, we achieve a baseline accuracy of 30.5%. Overall these results suggest that we achieve satisfying results given the task at hand. In these experiments, we explored a substantial part of possible hyperparameter settings for the chosen neuronal net architecture. This suggests that chances to reach substantial improvement in model performance with simple changes in the hyperparameter setting are rather low. Unfortunately, we do not have exact numbers on the Inter-Annotator Agreement in manual text zoning; hence the upper bound of human performance is unknown. It is possible that this upper bound is already reached with an accuracy of almost 90%.

The large amount of labeled training material proves to be highly beneficial for building an automatic solution for text zoning. Experiments with the size of the training set, as well as with the time period covered by the training set, both show that the total amount of labeled data is sufficient: The learning curve flattens out strongly in both experiments after 25,000 training examples. Adding more older job ads (i.e. job ads published before 1970) results in slightly decreasing model performance, when applying the model to most recently published job ads. Here, the positive effect of more data is partially surpassed by some kind of out-of-domain effect. However, the large amount of training data allows building a model that generalizes well: Overfitting seems not to be an issue, as there is only a minor difference in performance between development set and final test set (accuracy of 89.9% vs 89.8%). Admittedly, we did not run cross-validations, since the training process is computationally highly intensive, but results for development set and final test set are very similar, which fosters trust in our results. Particularly in combination with this large amount of available training data, task-specific embeddings prove to be at least as effective as pretrained word embeddings. For a future application, one main advantage of task-specific embeddings is that we can do without the (time consuming) preprocessing otherwise needed when using pretrained embeddings. In comparison with previous approaches to text zoning, feature engineering in this thesis is kept simple. But, similar to approaches like Sun et al. (2008) or Brants et al. (2002), choosing a good representation for distributional semantics turns out to be a decisive factor: Using pretrained word embeddings with no fine-tuning to the task results in considerably poorer performance. In conclusion, due to the large amount of training data, we are able to train a model with task-specific embeddings that generalizes well.
Results of more in-depth error analysis are encouraging as well. For the most interesting text zones regarding (our own) labor market research – description of the company and the job, required hard and soft skills – recall and precision are both around 90%. One limiting factor for model performance seems to be the skewed class distribution: The classes with the lowest frequencies (reason of the vacancy, description of a job agency, and incentives) show also poorest classification results. Merging classes after tagging, i.e. adding reason of the vacancy (Z20) and incentives (Z50) to the description of the job (Z60), we can further increase precision and recall, and reach an overall accuracy over 90%, while incurring only a minor information loss.

Even though Inter-Annotator Agreement between the different models is very high (Fleiss’ kappa over 0.9), ensembling proves to be beneficial. Firstly, it brings the largest performance improvement of all optimization steps per se, raising accuracy by 0.5 percentage points on the development set as well as on the test set. Secondly, ensembling allows us to identify text zones with low Inter-Model Agreement and to pass on these text zones to manual revision. This kind of post-processing is a second option for raising accuracy over 90%. The analysis of the confusion matrices shows, firstly, that a considerable share of errors deals with residual or administration text, which might not negatively influence downstream application too much, and secondly, the model has difficulties with the same decisions that are also difficult for human annotators. This can be explained mainly by the fact that certain text passages can refer to different text zones at the same time, and hence in manual segmentation not all decisions are made consistently. In other words, confusion is mainly due to modeling the task as a one-label classification, which seems to be the most appropriate modeling though, all things considered. All in all, these results of detailed error analysis underpin confidence in our model performance.

Error analysis shows not only satisfying results at large, but also that it is appropriate to use one and the same model for job ads of all publication channels and of different text length. Over the time period covered by our training data, job ads have undergone substantial change regarding the function of the text type, its vocabulary and its text length. However, our results suggest that these developments do not negatively affect quality of automatic text zoning. Therefore, we can be optimistic that our solution for automatic text zoning will work well for currently published job ads, and that we will not be forced to manually annotate large amounts of additional training material very often for future application. In conclusion, it seems that we succeeded in building a high performing solution based on neural networks for automatic text zoning for SJMM job ads.
5.2 Future Work

In order to better assess the quality of our model, we need to know the upper bound of the model performance. This upper bound depends strongly on the quality of labeled training data, or in other words, on how well and consistently human annotators can solve the task of text zoning. Therefore, we will evaluate Inter-Annotator Agreement for manual text zoning. To this end, we will apply our automatic text zoning solution to non-labeled job ads published from 2015 onwards and subsequently revise predicted labels manually. At the same time, this step will shed light on how well automatic text zoning works for most recently published job ads, and possibly allow us to identify further potential for model improvement.

Furthermore, we can perform additional experiments with the setting of hyperparameters. Empirical results so far do not strongly suggest that more capacity for the current neural network architecture will boost model performance. But the fact that we have not observed overfitting, might indicate that we have not yet reached the upper limit of model capacity. One option not tested so far, is to alter several model parameters at the same time, instead of increasing a single parameter while keeping the rest the same. If doing so, we probably also need to apply regularization strategies (e.g. word drop, see Goodfellow et al. 2016, p. 224ff.) to prevent the model from overfitting to the training data.

Additional options for model improvements include to pretrain domain-specific word embeddings on a very large unlabeled dataset of job ads, or, to add additional (local) features. For instance, using gazetteers with names of job agencies could help to improve recall for this zone (Z40). Likewise, we can provide the model potentially helpful information by using gazetteers on job titles, locations or company names. Finally, instead of merging certain text zones after classifying, we could train a model with fewer classes (six or seven instead of eight text zones). This would reduce the skewed class distribution considerably and imply only a minor information loss for downstream applications.
References


