Text Zoning for Job Advertisements with Bidirectional LSTMs

Ann-Sophie Gnehm
Institute of Computational Linguistics & Swiss Job Market Monitor
University of Zurich
gnehm@soziologie.uzh.ch

Abstract

We present an approach to text zoning for job advertisements with neural networks. Text zoning refers to segmenting texts into eight classes differing from each other regarding content. It aims at capturing text parts dedicated to particular subjects, e.g. the publishing company or qualifications wanted, and hence facilitates subsequent information extraction. We use Bi-LSTMs, a class of neural networks particularly suited for sequence labeling. Our best approach, with task-specific word embeddings and ensemble technique, reaches token-level accuracy of 89.8% and outperforms previous approaches with CRFs.

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. 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, descriptions of actual job tasks and much more. In this way, job openings are an excellent source for research on labor and labor market (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 of annotations on characteristics of the job, the person wanted, and the company offering the job, is manually added. The SJMM corpus hence provides a unique and rich database for labor market research. The aim of this study is to partially supersede manual annotation by automatic data processing with supervised machine learning, in order to lower costs of data collection and enlarge research opportunities.

Text zoning here is defined as segmenting the job advertisement text into zones (or classes), that differ from each other regarding their content. It intends at capturing text parts dedicated to particular entities or subjects such as the job task or the publishing company (see Table 1 for definitions and examples). In doing so, it simplifies information extraction. Firstly, text parts of interest can be localized more precisely. 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 CRM system" as a 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.

The goal of this work is to develop an automatic solution to text zoning for job ads. Firstly, we will test how well manual segmentation can be reproduced with supervised machine learning for the corpus as a whole. Since the main goal is to develop a classifier for future application, the automatic classification will then, secondly, be optimized regarding job ads from most recent years

zone	definition	examples
1	company description	“ein erfolgreiches Unternehmen der Baubranche”
2	reason of vacancy	“für unsere neu eröffnete Filiale”
3	administration & residual text	“wir suchen”, “Ihre Bewerbung senden Sie an”
4	job agency description	“Ihr Partner für die Vermittlung von Dauerstellen”
5	material incentives	“ansprechendes Salär”, “5 Wochen Ferien”
6	job description	“für den Kundenempfang”, “eine vielseitige Aufgabe”
7	required hard skills	“eine Ausbildung und Berufserfahrung als Sozialarbeiter”
8	required personality (soft skills)	“Sie sind belastbar und diskret”

Table 1: Definitions and examples of text zones

2 Related Work

A previous approach to automatize text zoning of the SJMM (Gnehm, 2016) with Conditional Random Fields (CRFs) and simple features, such as token uni- and bigrams, part-of-speech (PoS) bigrams, and the relative position tokens in text, achieved an accuracy on token level of 87.7%.

Though dealing with the same task in general, results of Hermes and Schandock (2017) are not directly comparable to results for text zoning on the SJMM corpus. Firstly, they classified whole paragraphs of job ads, not single tokens, and – as a consequence, to reach adequate description quality – they allowed multi-label classification. Furthermore, their classification taxonomy is only half of the size of the SJMM classification taxonomy (4 classes vs. 8 classes). Finally, they classified paragraphs independently from each other and did not model job ads as sequences of paragraphs. Their best approach with K-Nearest Neighbors Algorithms reached an accuracy of 97%.

The largest part of other research on text zoning deals with scientific papers or abstracts (also referred to as argumentative zoning), a smaller part focuses on more general texts, such as newspaper articles. Two conclusions for the task at hand may be drawn from these approaches: Firstly, modeling the task as sequence tagging, that is to consider context for the current classification decision, can improve performance considerably, as Merity et al. (2009) as well as Hirohata et al. (2008) showed. Secondly, if we 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. In many cases a single sentence refers to several different text zones.

3 Data & Methods

3.1 Data

The SJMM corpus comprises over 38,000 manually segmented job ads in German, covering the time period from 1950 to 2014. Ads are collected from corporate websites, online job portals and the press.

Wir/Z3 suchen/Z3 für/Z1 unser/Z1
attraktives/Z1 Gross-Sortiment/Z1
eine(n)/Z6 Dekorateur(in)/Z6 Wir/Z3
wünschen/Z3 :/Z3 -/Z7 Gute/Z7
Berufsbildung/Z7 und/Z7 Erfahrung/Z7
-/Z8 Kreativität/Z8 ,/Z8 Vielseitigkeit/Z8
-/Z8 Idealalter/Z8 25/Z8 -/Z8 40/Z8
Jahre/Z8 Wir/Z3 bieten/Z3 :/Z3 -/Z6
Weitgehende/Z6 Selbständigkeit/Z6 ,/Z6
-/Z6 grosses/Z6 Atelier/Z6 -/Z6
interessante/Z6 ,/Z6 vielseitige/Z6
Dauerstelle/Z6 ./Z6 Auf/Z3 Ihre/Z3
Bewerbung/Z3 [adr]/Z3 ,/Z3 8048/Z3
Zürich/Z3 ,/Z3 freut/Z3 sich/Z3 [kper]/Z3
./Z3 Foto/Z1 Hobby/Z1 AG/Z1

Figure 1: Example of segmented ad (each token with its zone tag)

In text zoning, job ads are segmented in eight different classes, distinguishable from each other with regard to their contents (see Table 1). The text is split up on token level, where each token is assigned exactly one class. Zone 3, which includes primarily information about the application proce-

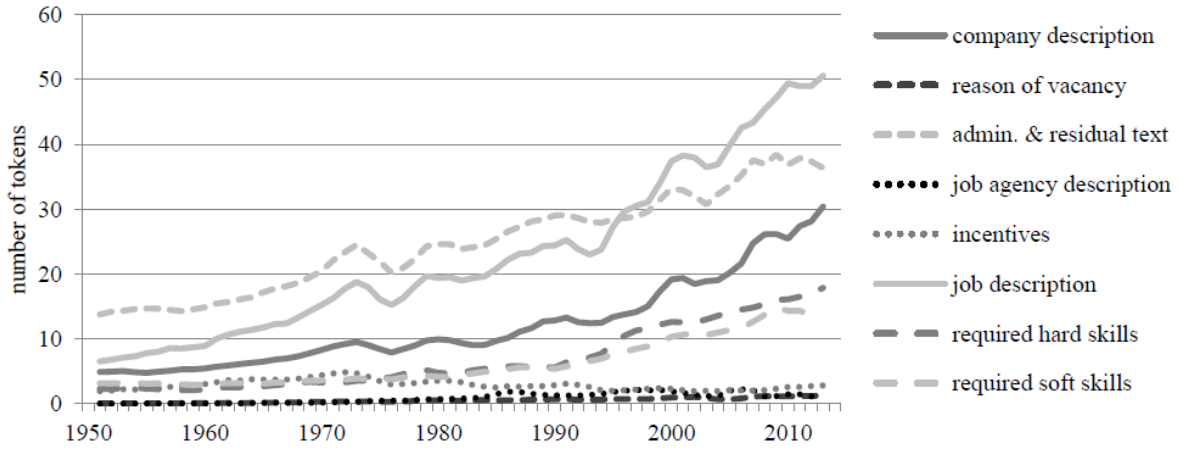

Figure 2: Number of tokens per zone 1950-2014 (press, moving averages over 3 years)

For example, most often the reason of the vacancy remains unknown. On the other hand, zones can show up several times in one ad, e.g. you will typically find information on the job description in several places (see Figure 1). In conclusion, text zoning for SJMM job ads is modeled as a one-label multiclass classification on the level of token sequences.

The text type of job ads has undergone noticeable change over the last decades. For instance, 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. Likewise, the amounts of text referring to required hard and soft skills rise strongly. Proportionally, all other zones become less important.

Figure 2 also reveals a strongly skewed class distribution: The two largest zones, Z6 and Z3, comprise around 30% of the tokens each, whereas Z2, Z4 and Z5 on the other hand have very little evidence with frequencies lower than 5%. A baseline classifier that assigns every token the majority class Z6 reaches an accuracy of 30.5%.

Furthermore, tokens show a high class ambiguity: More than 50% of the tokens appear in all eight classes. For the most part, these tokens are function words which by their nature usually adopt class affiliation from neighbored tokens. However, another 40% of the tokens have been seen in more than one class as well. This implies that the clas-

3.2 Methods

We model text zoning as a sequence labeling task (see Graves (2012) for further discussion): we tag sequences of input data (tokens) with sequences of labels (zone tags), where neither single tokens nor single tags represent individual data points, but instead the current token and the current zone tag strongly depend on preceding and subsequent tokens or tags, respectively.

As we have a large amount of labeled data at hand, we train text zoning models with supervised machine learning. We use BiLSTMs, a class of recurrent neural networks with two improvements that make them especially suited 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. Secondly, bidirectional LSTMs consider *context on both sides* for the classification of the actual token (Graves, 2012).

Feature engineering is rather simple, since the only input to the model are word embeddings. To use pretrained word embeddings can improve model performance, as initial values of parameters (i.e. values of pretrained vectors vs. values of randomly initialized vectors) influence model regularization. On the other hand, to train task-specific embeddings, where dimensionality reduction of input vectors is done within the model optimization process itself, seems to be promising in light of the relatively large amount of labeled data

Conceptnet Numberbatch embeddings (Speer and Chin, 2016), which achieve measurably higher performance in word-similarity evaluations than any previous known system. Conceptnet comprises 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. We conducted several processing steps to reach this objective, most important a semantic reduction of compound words: If there is no embedding for the (lemmatized) compound word, we search the embedding for the head, respectively the embedding for the largest part if there is more than one modifier ("vermittler" instead of "personalvermittler", "dienstleistungsunternehmen" instead of "mediendienstleistungsunternehmen"). Decompounding has been done with the GERTWOL tool for morphological analysis (Koskeniemmi and Haapalainen, 1996). After all, for three out of four tokens (respectively for 92% of tokens without numeral, punctuation and special characters) a pretrained embedding is available. This coverage seems to be sufficient for using the embeddings as input.

For rare words, that is, words that occur no more than five times in the training data or words not covered by a pretrained embedding, we use character-level representations, as in the approach by Neubig et al. (2017). In doing so, as there is much more data on character level, the model can generalize better. Furthermore, this approach also solves the unknown word problem: 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).

We use the network architecture of the BiLSTM PoS tagger by Huang et al. (2015) as starting point for our experiments[2]: Bidirectional token-level LSTMs encode the input, and on top of the states of the token-level BiLSTMs comes a Multilayer Perceptron (MLP) with one hidden layer. For rare tokens, the training process optimizes an embedding over the characters: BiLSTMs encode the input, and the concatenated output vectors serve as input for the token-level BiLSTMs. Network parameters are randomly initialized and optimized with Adam algorithm (Kingma and Ba, 2014).

[2]We [illegible]

4 Experiments & Results

4.1 Experiments

For all experiments, we split annotated data into a training set (80% of ads), a development set (10%) and a final test set (10%). We train models in 50 (or 30) iterations and keep the model that reaches highest accuracy on the development set. Accuracy, as well as precision and recall, are assessed on token level. To validate our most important results, we run specific model settings five times and report mean accuracy and standard deviation of model performance over the five runs.

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. Furthermore, a task specific tuning of embeddings might be promising considering the large amount of training data.

A second series of experiments deals with the question, whether changing the hyperparameters can increase model performance, as performance usually rises with model capacity (see e.g. Collobert et al. (2011)).

Thirdly, the main objective is to build a model that is optimized to segment present-day or future job ads. As mentioned in Section 3.2, job ads as a text type have undergone major changes in the past decades. Thus, it is probable that older training data lowers model performance. On the other hand, excluding older job ads results in a smaller training set. 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.

At the very end, automatic text zoning is optimized by an ensemble technique: Five models with the same setting, but different (random) initialization will be trained. As they start from different points in the hypothesis space, the classifiers will differ slightly from each other (Collobert et al., 2011; Rokach, 2010). The test set will be tagged by the five classifiers, keeping for every token the majority vote of the five models as final classification. Especially if agreement between the models is not very high, we can expect an increase in model performance.

4.2 Results

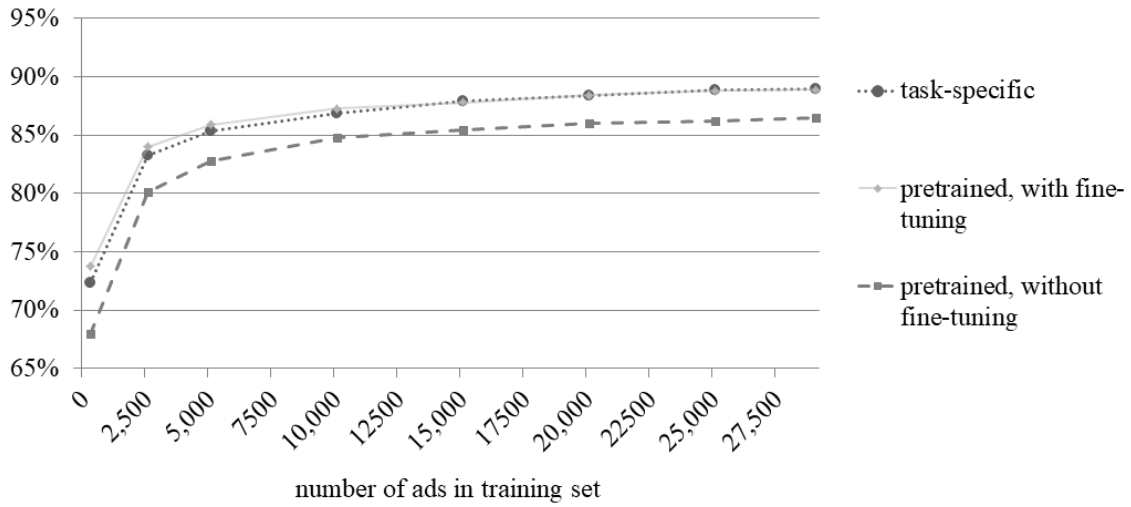

Figure 3: Effects of training set size on development set accuracy

(2015), as described in Section 3.2, 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 a previous approach with CRFs (Gnehm, 2016).

embeddings	accuracy	s.d.
task-specific	89.0%	0.03%
pretrained, not fine-tuned	86.5%	0.05%
pretrained , fine-tuned	88.9%	0.04%

Table 2: Development set accuracy of models with task-specific and pretrained embeddings (mean accuracy over 5 model runs)

curacy is clearly lower when using pretrained embeddings (86.5%) than when training task-specific embeddings (89.0%). Allowing the pretrained embeddings to be adapted during training brings a substantial gain in performance, but still accuracy is still slightly lower than with task-specific embeddings (88.9%). These results indicate that a task-specific training or at least adaption of embeddings is fruitful for job ad segmentation.

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.[3]

Experiments with the **size of the training set**

[3]The most positive effect of character-based embeddings (accuracy of 86.5% vs. 79.6%) is obviously observed in the

show that the amount of labeled data is large enough to learn a robust model. As can be seen from Figure 3, the learning curve levels off strongly as the training set size increases, for all three models. 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 randomly initialized – 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. 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 which results in poorer performance. In conclusion, these results suggest that learning task-specific representations here is successful given the large amount of labeled training data. Based on these results, continuing experiments without pretrained word embeddings (and time-consuming preprocessing) and instead training word embeddings as part of the model parametrization is recommended.

Experiments with **hyperparameter settings** show that adding a second hidden layer to the token-level BiLSTMs raises accuracy up to 89.2%. However, adding more layers does not improve classification, while increasing the training time considerably. Also, other configurations with increased hyperparameters – bringing more capacity to the model (additional layers on character-level BiLSTMs, or higher dimensional input vectors) – do not boost performance. Therefore, we continued experiments with only a second layer added on the token-level BiLSTMs.

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.[4] As can be seen from Table 3, adding more and older training data slightly improves performance, but this holds only for jobs ads published from 1970 onward. Going back further in time results in decreasing model performance, indicating that the effect of more data is partly weakened by some sort of out-of-domain effect. 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 onward.

time period	accuracy	s.d.	train set size	
			ads	tokens
as of 2010	88.9%	0.11%	5K	1,1M
as of 2000	89.1%	0.03%	10K	1,7M
as of 1990	89.2%	0.02%	15K	2,1M
as of 1980	89.2%	0.04%	20K	2,5M
as of 1970	89.3%	0.04%	25K	2,8M
as of 1960	89.2%	0.06%	30K	3,1M
as of 1950	89.2%	0.05%	32K	3,2M

Table 3: Development set accuracy with training data covering different time periods (mean accuracy over 5 model runs)

However, using older training data does not lower model performance substantially. Thus, long term change of job ads as a text type over time 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. The model using training data from 1970 onward 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.

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, we don't need training data older than 1970. And fourth, additional detailed analyzes showed that it is advisable to use one single model for all job ads, regardless of text length or publication chan-

[4]Previous experiments have shown that highest accuracy

By using an **ensemble of classifiers**, model performance further increases. Given the best model specification according to the experiments presented above, we trained five classifiers with different (random) initial parameters. As can be seen from Table 4, the classifiers vary only slightly in accuracy. Also, the tiny differences in results on development and test set indicate that models do not suffer from a development set overfit. Furthermore, Inter-Annotator Agreement between the five models is very high, as Fleiss' kappa values over 0.92 indicate. 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.

	dev set	test set
accuracy model 1	89.4%	89.3%
accuracy model 2	89.1%	89.2%
accuracy model 3	89.1%	89.3%
accuracy model 4	89.2%	89.2%
accuracy model 5	89.2%	89.2%
mean acc. model 1-5	89.2%	89.2%
s.d.	0.12%	0.06%
accuracy with ensembling	89.9%	89.8%
Fleiss' kappa	0.924	0.923

Table 4: Evaluation of single models and ensembling on development set and test set

Besides, ensembling brings an additional benefit for real-world applications, as disagreement between the models can support **human quality control**: We can identify tokens without a majority vote (this applies to 0.5% of the tokens in the test set) and correct them manually. For the present case, this post-processing would raise accuracy above 90%. Manual inspection further reveals that not all deviations from gold standard are problematic: Table 5 shows an example for a text passage where model predictions differ from gold standard tags only regarding the question of how detailed to split text into residual text and other zones. The predicted tags make sense as well and will not negatively affect subsequent information extraction. Results of manual inspection suggest that a considerable part of predictions differ

token	gold standard	model prediction
einer	soft skills	soft skills
exakten	soft skills	soft skills
Person	soft skills	soft skills
bietet	job descr.	admin./resid. text
sich	job descr.	admin./resid. text
hier	job descr.	admin./resid. text
eine	job descr.	job descr.
spannende	job descr.	job descr.
Aufgabe	job descr.	job descr.

Table 5: Text passage with acceptable differences between predicted and true labels

show that regarding the most interesting text zones for labor market research – description of the company and the job, as well as required hard skills and soft skills – evaluation measures vary roughly around 90% (see Table 6). Just for required soft skills, recall is somewhat lower (84%). The reason for this latter finding is a question for further investigation.

Clearly, the skewed class distribution is a limiting factor to model performance. Evaluation measures, notably recall, are relatively low for text zones that are not so frequent, i.e. reason of the vacancy, description of job agencies and incentives. In terms of content, it can be justified to merge reason of the vacancy (Z2) and incentives (Z5) after labeling to the description of the job (Z6). By merging Z2 and Z6, recall for the resulting new class would rise up to 89.7%, precision up to 92.1%, and accuracy over all classes would increase up to 90.1%. If we merge additionally Z5, 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 most important labor market research interests.

zone	precision	recall
1 company description	89.1%	89.4%
2 reason of vacancy	87.0%	72.2%
3 admin. and residual text	93.5%	91.3%
4 job agency description	80.1%	65.4%
5 incentives	85.4%	75.1%
6 job description	89.5%	92.1%
7 required hard skills	90.7%	91.8%
8 required soft skills	90.2%	83.9%

As an additional experiment, we tried out **Viterbi algorithm for global decoding** in order to find the optimal sequence of classification labels. Global decoding is supposed to smooth out label sequences and adjust tags of ambiguous tokens to tags of their neighbors. However, Viterbi decoding does not bring any improvement (accuracy = 89.1%, s.d. = 0.12%). This implies that BiLSTMs by themselves succeed in using context information in order to find appropriate label sequences.

5 Conclusion

5.1 Summary

Overall, the results of supervised machine learning with BiLSTMs for 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 results of an earlier approach with CRFs (Gnehm, 2016) this is an improvement of more than 2 percentage points, or an error rate reduction on the token level by 16%. These results suggest that we achieve satisfying results given the task at hand.

The large amount of labeled training material proves to be highly beneficial. Experiments with the size and the time period of the training set show that the total amount of labeled data is sufficient. When applying the model to most recently published job ads, adding job ads published before 1970 results in decreasing performance, though only slightly. Here, the positive effect of more data might be partially surpassed by some kind of out-of-domain effect.

Particularly in combination with this large amount of 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 needed when using pretrained embeddings. In conclusion, due to the large amount of training data, we are able to train a model with simple feature engineering that generalizes well.

Results of more in-depth error analysis are en- 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. By merging classes after tagging, i.e. adding reason of the vacancy and incentives to the description of the job, we can further increase performance and reach an overall accuracy over 90%, while incurring only a minor information loss.

Applying an ensemble technique proves to be beneficial. Firstly, it brings the largest performance improvement of all optimization steps per se, raising accuracy by 0.5 percentage points. Secondly, ensembling allows us to identify text zones with low Inter-Model Agreement and to pass on these text zones to manual revision. Applying this kind of post-processing to the 0.5% in the test set without a majority vote is a second option for raising accuracy over 90%.

5.2 Future Work

While results so far do not suggest that more capacity for the current network architecture will boost performance, the fact that we have not observed overfitting might indicate that we have not yet reached the upper limit of model capacity. Future experiments could test the effect of altering several model parameters at the same time.

Options for model improvements include to pretrain domain-specific word embeddings on a very large unlabeled dataset of job ads, or to add local features by using gazetteers on job titles, locations, company and job agency names.

Instead of merging certain text zones after classifying, we could train a model with fewer classes. This would reduce the skewed class distribution problem considerably and imply only a minor information loss for downstream applications.

Finally, we could also compare BiLSTMs with other sequence tagging algorithms and use more sophisticated ensemble methods to improve our text zoning solution.

Acknowledgments

I would like to thank my supervisor Dr. Simon Clematide for his guidance, helpful advice and for

References

Thorsten Brants, Francince Chen, and Ioannis Tsochantaridis. 2002. Topic-based document segmentation with probabilistic latent semantic analysis. In *Proceedings of the 11th international conference on information and knowledge management*. ACM, pages 211–218. https://doi.org/10.1145/584792.584829.

Marlis Buchmann, Helen Buchs, Ann-Sophie Gnehm, Debra Hevenstone, Urs Klarer, Marianne Mueller, Stefan Sacchi, and Alexander Salvisberg. 2016. *Swiss Job Market Monitor: Scientific Use File Documentation (Release 2016)*. University of Zurich: Swiss Job Market Monitor. Distributed by FORS, Lausanne (http://forsbase.unil.ch).

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537. https://dl.acm.org/citation.cfm?id=2078186.

Ann-Sophie Gnehm. 2016. *Erschliessung des Korpus Stellenmarkt-Monitor Schweiz und Automatisierte Textsegmentation mit Supervised Machine Learning. (Unveroeffentlichte Dokumentation Programmierprojekt)*. Universitaet Zuerich: Institut fuer Computerlinguistik.

Yoav Goldberg. 2017. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, San Rafael.

Alex Graves. 2012. *Supervised sequence labelling with recurrent neural networks*. Springer, Heidelberg.

Juergen Hermes and Manuel Schandock. 2017. *Stellenanzeigenanalyse in der Qualifikationsentwicklungsforschung: Die Nutzung maschineller Lernverfahren zur Klassifikation von Textabschnitten*. Bundesinstitut fuer Berufsbildung, Bonn.

Kenjii Hirohata, Naoaki Okazaki, Sophia Ananiadou, and Mitsuru Ishizuka. 2008. Identifying sections in scientific abstracts using conditional random fields. In *Proceedings of the Third International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing (AFNLP), volume 1, pages 381–388. http://www.aclweb.org/anthology/I08-1050.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. *Bidirectional LSTM-CRF models for sequence tagging*. arXiv. https://arxiv.org/abs/1508.01991.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Kimmo Koskeniemmi and Mariikka Haapalainen Niemeyer, Tbingen, number 34 in Sprache und Information, pages 121–140.

Stephen Merity, Tara Murphy, and James R. Curran. 2009. Accurate argumentative zoning with maximum entropy models. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*. Association for Computational Linguistics, pages 19–26. http://aclweb.org/anthology/W09-3603.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Callesteros, David Chiang, Daniel Clothiaux, and Trevor Cohn. 2017. *DyNet: The Dynamic Neural Network Toolkit*. arXiv. https://arxiv.org/abs/1701.03980.

Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review* 33(1):1–39. https://doi.org/10.1007/s10462-009-9124-7.

Robert Speer and Joshua Chin. 2016. *An ensemble method to produce high-quality word embeddings*. arXiv. https://arxiv.org/abs/1604.01692.

Qi Sun, Runxin Li, Dingsheng Luo, and Xihong Wu. 2008. Text segmentation with lda-based fisher kernel. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (Short Papers)*. Association for Computational Linguistics, pages 269–272. https://doi.org/10.3115/1557690.1557768.