Algorithmic Segmentation of Job Ads Using Textual Analysis

Michael Tschuggnall[1], Benjamin Murauer[1], Günther Specht[1], and Julia Brandl[2]

[1]Department of Computer Science, University of Innsbruck, Austria
[2] Department of Organization and Learning, University of Innsbruck, Austria
{firstname.lastname}@uibk.ac.at

Abstract. As job ads are getting more prevalent online, an automated analysis is becoming increasingly important, especially in the field of human resource management. In this paper, we propose an approach to automatically segment job ads by predefined categories like the description of a job or the offering company, which is needed to categorize and quantify different aspects of job ads. Using a manually annotated data set, textual features are extracted for each segment type in a first step and utilized to train state-of-the-art machine learning classification methods. Subsequently, these models are used by iterative algorithms to detect the individual segments. Using several optimization techniques like detecting typical segment start phrases, comprehensive evaluations show promising results.

1 Introduction

With the rise of the interconnected world wide web, the amount of data digitally transferred through various channels is increasing steadily. Especially businesses have adjusted and optimized their workflows by utilizing new possibilities of data exchange in many areas like allowing location-separated global teams or performing targeted social media marketing campaigns. Consequently, also the publication of job ads has been expanded or even completely shifted from print media to online services. With massive amounts of digitally available job ads, a crucial task in the field of human resource management (HRM) is to systematically analyze them to be able to answer questions like "What does the market want?", "Which companies search for which person types?" or "In which domains are social skills more important?" [4, 21]

Typical study examples include attempts to detect trending key requirements demanded by employers in a specific domain, or the extraction of what is offered by companies in return (e.g., [1, 3]). Most of those studies are done by manual inspection of advertisements, which has the drawback that only a small subset of potential sources can be examined due to limitations of human resources. To conduct quantitative research, an automated textual analysis is needed. A fundamental prerequisite for making meaningful statements is thereby an automated identification of well-known segments like the description of the job tasks or requirements within ads. For example, it makes a crucial difference if the phrase "work experience" is stated within the job requirements or appears in a sentence like "You will be integrated in a team with long-term work experience", as different conclusions can be drawn depending on the position of the phrase.

Given that a job ad depicts the employer's offer and the future employee's duties, the job ad content can be divided into four categories [6], whereby not necessarily all of them have to be present: the (i) **company description** introduces the offering company, who they are, what they do etc., the (ii) **job description** describes the job itself with all its components, in the (iii) **requirements** it is stated which education, technical/social skills etc. are demanded from the applicant, and finally the (iv) **offer** segment describes what the company offers, including salary and other benefits. Examples of those four categories are shown in Table 1.[1]

Category	**Example 1**	**Example 2**
company description	In 2017, *CompanyX* not only launched a completely new product, it created and has led ever since a whole new product category. Nowadays *CompanyX* employs more than 12000 people in over 165 countries, selling over 5 billion products a year. The World of *CompanyX* provides the forum for you to use your talent and passion, to develop yourself and make an impact.	About us: we look for innovation everywhere. For 130 years, we have been at the forefront of innovation, but finding solutions to the world's biggest problems has never been more important than right now. Join us today and become an essential part of the solution!
job description	The Controller role is responsible for acting as business partner to the Sr. VP or VP of the defined area and the respective management team. Project lead and participation in the respective functional area is also part of the role. In addition, steering support processes within his/her defined area and contributing to core financial processes is expected from the position holder as well.	Your responsibilities – Work on the software verification (component test, system test, building verification environment) – Develop automated test cases and work closely with the development team to ensure testability – Take part in the analysis of problems that occur in live operations – Travel in order to directly support our clients in the verification phase of the project
requirements	Because our business is dynamic and advances in science and technology require new methods of production we are looking for individuals who can do the following: (i) collaborate with team members in the identification and implementation of continuous improvement initiatives and action plans. (ii) Support activities in the areas of cost containment, efficiency, productivity, energy conservation, waste minimization, operational excellence and lean practices.	Your profile – Experience with Scrum or other iterative methodologies and experience with C# Development – Minimum of 4 years of experience designing and developing systems in a .Net environment – Experience with JavaScript, not only jQuery – Prior experience with a source control system (SVN, GIT, TFS, . . .) – Good verbal and written communication skills in English and German (at least level B2)
offer	The minimum annual salary for the position is 50,000$ gross, whereby the effective salary is based on your qualification and experience.	We offer: – an interesting job within the packaging & paper industry – to be part of a successful multicultural company – an empowering environment – new office building in the city center – several attractive social benefits

Table 1: Examples for the Different Segment Types (Anonymized).

[1] Note that while the example is provided in English for better understandability, the data used throughout this paper is in German.

This paper presents an approach to automatically detect those categories, if present, by applying textual analysis based on state-of-the-art machine learning techniques. It can thereby be seen as a hybrid approach that incorporates both machine learning techniques as well as specifically developed algorithms operating on top. Respectively, the two major steps are: (i) training a model in order to be able to classify each segment, and (ii) calculating the final segmentation utilizing this model. In the first step, a data set with manually assigned ground truth is used to train a model which is able to identify each segment type with high accuracy when it is given correctly separated segments as input, i.e., one of the four categories and not overlapping structures. Using this model, algorithms are designed to enable an automatic classification even if the borders are not previously known, and thus segmenting and clustering the job ad by the defined categories.

Summarizing the main contribution of this paper, we present an algorithmic text segmentation approach that is specifically tailored for job ads. Utilizing job-ad-specific knowledge, experiments reveal that algorithmic segmentation represents a valid and comparable alternative to Conditional Random Fields (CRF) [17], a state-of-the-art technique used in text segmentation and tagging.

The rest of this paper is organized as follows: Section 2 briefly outlines related work, and Section 3 explains the generation of the model in detail, which is needed as a prerequisite for the actual job ad segmentation presented in Section 4. The resulting basic algorithm is evaluated in Section 5, which is improved and reevaluated in Section 6. Finally, a conclusion and possible future work is discussed in Section 7.

2 Related Work

Job advertisements have become more prevalent online, and they have shown to improve the chances of being employed [16]. Therefore, research on job ads is becoming more important in many fields, with classic examples ranging from optimizing skill sets for specific positions [1] to optimizing strategies for HR departments [11]. The information to be gained from the text differs greatly. For example, for some applications detailed features that are required by potential applicants are extracted [3], while others try to extract information to fill specific templates [7]. Also, complex frameworks have been constructed to automatically use job ads and distribute them over different channels [15], where detailed information is extracted by rule-based models, including regular expressions and other manually tailored methods. In order to improve the extraction procedures, one goal is to detect specific topics within the text, whereby there exists no general rule of how many parts a job should be divided into. For example, [18] extract 13 different categories from the text of French job ads, including, e.g., contact information or mobility requirements, whereas [34] divide their documents into 3 more general sections (education, job title and sector). Both approaches rely on common textual features like word, character or POS-tag frequencies.

To cope with the general text and/or topic segmentation problem, a wide range of different methods is in use, often based on the research by Hearst [14], in which the lexical cohesion of terms is analyzed. Generally, approaches utilizing the lexical cohesion of segments follow either a *local* or *global* strategy. Local methods thereby process documents using fixed-sized fragments, aiming to locally detect significant changes in cohesion (e.g., [14, 12]). On the other hand, global methods try to maximize the lexical cohesion for each segment from a global perspective (e.g. [8, 32]). Due to the fact that both strategies have their pros and cons, i.e., leading to different accuracies in different situations, e.g., Simon et al. propose a graph-based hybrid approach [29]. With respect to this terminology, the presented approach in this paper can also be seen as a text segmentation problem that falls into the hybrid category, as it locally traverses text blocks using sliding windows while considering the global view. Nevertheless, in contrast to typical approaches in this field, the ad segmentation does not aim to draw borders based on topic or genre changes, but by predefined job ad categories. Moreover, segments must be attributed to those categories, i.e., only finding borders is insufficient in this case.

In terms of style markers used, approaches utilize different features and methods, e.g., Bayesian models [10], Hidden Markov Models [5], vocabulary analysis in various forms like word stem repetitions [24] or word frequency models [25]. While there exist some recent papers (e.g., [26, 20]) that include a comparison between some of the segmentation approaches on the same data sets, in general it is difficult to compare performances due to the heterogeneous problem and data types being approached. To counter this problem, the PAN workshop series[2] recently proposed tasks to create stylistic clusters within documents, revealing that accurate stylistic segmentation is still a difficult problem [30, 31].

With respect to automatically labelling tokens from a text document (e.g., POS tagging or Named Entity Recognition), Conditional Random Fields (CRF) have proven to be effective [28, 19]. Moreover, they have successfully been used for other sequential labeling problems, such as detecting common fields like authors, title, year or publisher from the headers and citations of research papers [22].

3 Preliminary Experiments: Training a Textual Model

To tackle the problem of automatically detecting the four predefined segment types (i) *company description*, (ii) *job description*, (iii) *requirements* and (iv) *offer* within an unseen job ad, at first a textual model is needed that can differentiate between those segments. I.e., arbitrarily given one of those segments, the model should automatically be able to predict the correct one based on textual characteristics. In this work, two variants, i.e., models based on dictionaries and machine-learned models have been built and tested. Both variants are presented in the following and are created on the basis of a manually tagged data set

[2] http://pan.webis.de, visited Jan. 2018

consisting of approximately 1,200 manually tagged Austrian job ads, written in German language[3].

3.1 Dictionary-Based Models

Due to the fact that job ads are formulated very heterogeneously, e.g., containing full sentences describing the job in detail, or consisting basically only of a job title accompanied by bullet lists, the main unit taken into account are words (including stemming and n-grams). As a first attempt, dictionaries for each category are created using the tagged data set. The prediction is then made by comparing the given segment with the dictionaries and computing similarities. Concretely, the following methods have been tested:

(1.) *Simple Stems*: In this variant the dictionary of a segment consists of all occurring stems and their normalized weights. Given an unseen segment S, the similarity to a dictionary D is then calculated by summing up the weights in D for every occurring stem in S. By using this computation, important stems that frequently appear in a specific segment are also of higher importance with respect to the final similarity score.

(2.) *N-Grams*: This approach is very similar to the previous one and only replaces stems by character n-grams (using $n = \{2, 3, 4, 5, 6\}$). Thus, a dictionary contains all occurring n-grams including their normalized weights, and the similarity is computed by summing up the weights of all matching n-grams.

(3.) *Lucene*: Finally, the query model of the Java library *Lucene*[4] has been utilized, which uses the tf-idf metric and vector space models. Thereby a document is created for each segment type by concatenating all samples, which is subsequently given to Lucene to create an appropriate index. To be specific, four different indices are created, each consisting of only one document containing all samples of the respective job ad category. Finally, the similarity of an unseen segment S to a dictionary D is then computed as follows: For each word in S, a query is formed that consists of this word only, and the similarity score for the dictionary document is retrieved from Lucene. The overall score is then computed by summing up the delivered scores of all queries (words).

3.2 Machine-Learned Models

As an alternative to the dictionary-based prediction, common machine learning techniques have been applied to build a model which can predict the segment type. The list of provided features is as follows: (i) single word stems *(1s)*, stemmed word 2-grams *(2s)* and stemmed word 3-grams *(3s)*, each of them by removing or keeping stop words (*nostop / stop)*, (ii) all possible combinations of the previous features (e.g., *1s-3s-nostop* for using single stems and stemmed word 3-grams, by eliminating stop words), and (iii) character n-grams using $n = \{2, 3, 4, 5, 6\}$.

[3] original data provided by *textkernel*, https://www.textkernel.com, visited Jan. 2018
[4] Lucene, https://lucene.apache.org, visited Jan. 2018

To determine the best working algorithm for this approach, several commonly used methods have been tested. Using the WEKA toolkit as a general framework [13], the following classifiers have been utilized: Naive Bayes a Bayes Network using the K2 classifier, Large Linear Classification using LibLinear, a support vector machine using LibSVM with nu-SVC classification, a k-Nearest-Neighbours classifier (kNN)using $k = 1$, PART, and a pruned C4.5 decision tree (J48). Thereby all classifiers have been used with their respective standard settings.

3.3 Model Evaluation

Both the dictionary-based as well as the machine-learned models have been evaluated using a manually tagged data set, which has been created from a subset of 1,200 samples of all Austrian job ads in 2015. All considered ads are formulated in German language, and manually segmented into the four segment types by an expert group of three persons with high experience in the human resource management field. The content of the samples thereby varies from mainly just bullet lists to fully formulated, grammatically correct sentences. Moreover, segment types may also be spread over multiple positions in the text, e.g., stating a job requirement at the beginning as well as at the end of an ad.

The data set has been divided into two halves, whereby one half has been used as the training/test corpus for the evaluation of the models, and the other half for the evaluation of the segmentation algorithm (see Section 5). From the 600 job ads serving for the model evaluation, 300 have been used to train the models, i.e., to build the dictionaries or to train the classifiers, respectively, and the remainder has been used for testing. Finally, the size of the dictionaries (d_s) has been parameterized (except for Lucene), choosing the dictionary to contain only the 50, 75, 100, 150 or 200 most frequent stems/n-grams, or to contain all of them. In case of the machine-learned models, the number of features has also been parameterized by the same criteria.

method	**d$_s$**	**cmp**	**job**	**offer**	**req**	**avg**
Lucene	*all*	85.7	84.8	89.2	93.1	**88.2**
5-grams	75	73.2	76.6	81.4	88.3	**79.9**
6-grams	75	72.3	76.6	81.8	84.0	**78.7**
stems	200	75.3	72.7	77.9	87.0	**78.2**
4-grams	75	69.3	75.3	74.9	81.4	**75.2**
3-grams	100	61.0	69.3	66.7	75.8	**68.2**

(a) Dictionary-Based Models

classifier	**d$_s$**	**features**	**cmp**	**job**	**offer**	**req**	**avg**
LibSVM	100	1s-2s-stop	94.8	94.7	97.0	97.9	**96.1**
BayesNet	150	4-grams	92.5	92.7	97.2	96.1	**94.6**
Naive Bayes	75	1s-3s-stop	93.0	92.1	95.5	96.4	**94.2**
kNN	150	5-grams	88.2	92.1	93.2	96.8	**92.6**
LibLinear	150	4-grams	88.0	93.1	96.3	90.8	**92.0**
PART	100	1s-3s-stop	81.0	84.8	90.4	89.4	**86.4**
J48	100	1s-3s-stop	79.1	82.2	91.2	89.6	**85.5**

(b) Machine-Learned Models

Table 2: Best Accuracies of Dictionary-Based and Machine-Learned Models in Percent, Given Already Segmented Categories.

The best results for both dictionary-based and machine-learned models are shown in Table 2, where it can be that – while Lucene performs best for the former – most machine learning algorithms outperform it substantially. By utilizing frequencies of single stems and stemmed 2-grams, LibSVM could achieve an accuracy of 96%. In general, good accuracies are achieved for all four segment types by restricting the number of features, with the *offer* and *requirement* type having slightly better values.

4 Ad Segmentation

According to the nature of most job ads, which are given as unstructured full texts, the previously discussed classification cannot be used as is, because the borders of the different segments are not known. That is, a classifier can only predict the segment type with high accuracy if the complete and correctly separated segment is given as input. To tackle this limitation, an algorithm has been developed that is based on the constrained classifier, but allows to estimate segment borders. In the first step, a model-based probability is calculated for each text position to belong to each segment type. Subsequently, these values serve as input for the final ad segmentation, basically by locating probability peaks and applying thresholds to widen a segment.

4.1 Applying the Model

Given a full text job ad, at first a probability for each segment type and text position (token) is estimated. This is done by iteratively traversing the text using sliding windows of length w_l and step w_s, using the model to predict a vector $[p_C, p_J, p_O, p_R]$ for each window. The values in this vector correspond to the probability for the window to be of the type *company*, *job*, *offer* and *requirement*, respectively. For example, Fig. 1 shows three windows of length seven, where for each of the windows a corresponding vector is computed using the model. The window step w_s in the example is three tokens.

After processing all sliding windows, the probability vector for each token is computed by calculating the average of all predicted window vectors where the respective token appears. In the example, the first three words have been predicted by only one window, and thus the probability vectors for those words correspond to the vector of the corresponding window. On the other side, the tokens *'of', 'our', 'team'* have been predicted by two windows, and thus the token vectors represent the average of those. Finally, *'you'* has been predicted by three windows and is consequently calculated as the average of all three involved windows.

The overall procedure can be formalized as is stated in Algorithm 1. Given the consecutive list of tokens T of the job ad, it computes the average probability vector P_i for each token T_i by applying the model function $prob(t)$ on the respective tokens t of each window. According to the results of Section 3.3, machine-learned models are used for calculating the respective probabilities. Thereby, as

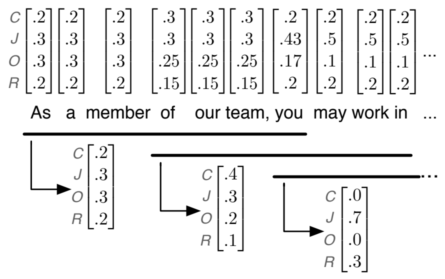

Fig. 1: Example of Computing Probability Vectors for the Categories Company Description (C), Job Description (J), Offer (O) and Requirements (R) .

not all classifiers deliver an estimation for each class (segment type) but only the predicted class (e.g., LibSVM would only produce $[0, 0, 1, 0]$, whereas BayesNet would produce a more informative vector like $[.1, .1, .5, .3]$), the model function $prob(t)$ utilizes BayesNet instead of the slightly more accurate LibSVM classifier.

A visual example of the probability calculation for a sample job ad is depicted in Fig. 2, which shows the probabilities for each token and segment type. The text document spans on the x-axis from left to right, and the ground truth for each type is included in colored rectangles. It can be seen that each segment type is matched by the probability distribution, having peaks in the correct corresponding segment.[5]

4.2 Calculating the Final Segmentation

On the basis of the probability vectors for each token and segment type, the aim of the last step is to compute the final segmentation. As can be seen visually, probabilities like in Fig. 2 already match their corresponding segment type, i.e., having their peaks within the correct frame. To determine the whole fragments algorithmically, the basic idea is to use thresholds that widen a peak to the left and the right, respectively, until the probability falls below the thresholds. According to Algorithm 2, the procedure is as follows: (1.) Out of the unattributed segment types, choose the one with the highest peak. (2.) Starting from the peak, go to the left and attribute every token to the chosen type, until the threshold is reached or the token is already attributed to another type. (3.) Starting from the peak, do the same to the right.

During the execution, it may happen that peaks for a segment type appear within an already attributed segment. In this case, this segment type has no attribution, which also reflects the possibility that a job ad may not contain a certain segment type. In a similar way, segment types have no final attribution

[5] What can be observed in addition is that not all tokens correspond to a segment type, i.e., the text between company and job description and at the end of the ad is general and thus not attributed to a specific type.

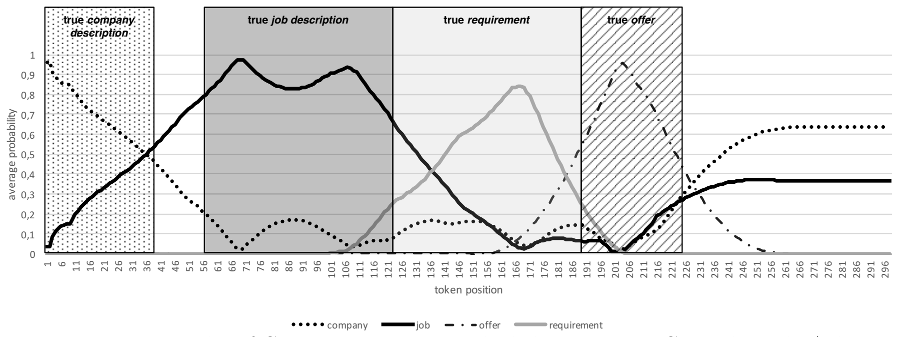

Fig. 2: Example of Segment Type Probabilities For a Sample Job Ad.

if an unattributed peak is found, but left and right positions are already below the given thresholds.

Algorithm 1 Probability Vector Calculation

```
input:
    $T$              list of tokens
    $w_l$, $w_s$     window length, window step
    $prob(t)$        computes the probability vector
                     for the tokens $t$
output:
    $P$              list of average probability vectors

 1: for $i$ from 1 to length($T$) do $v_i \leftarrow []$
 2: end for

 3: for $i$ from 1 step $w_s$ to length($T$) do
 4:     $t = [T_i, ..., T_{i+w_l}]$
 5:     $[p_C, p_J, p_O, p_R] \leftarrow prob(t)$
 6:     for $j$ from $i$ to $i + w_l$ do
 7:         append $[p_C, p_J, p_O, p_R]$ to $v_j$
 8:     end for
 9: end for

10: for $i$ from 1 to length($T$) do
11:     $a_C \leftarrow$ average of all $p_C$ in $v_j$
12:     $a_J \leftarrow$ average of all $p_J$ in $v_j$
13:     $a_O \leftarrow$ average of all $p_O$ in $v_j$
14:     $a_R \leftarrow$ average of all $p_R$ in $v_j$
15:     $P_i \leftarrow [a_C, a_J, a_O, a_R]$
16: end for
```

Algorithm 2 Final Segmentation

```
input:
    $P_t$              token probabilities for type $t$
    $\delta_l$, $\delta_r$   left and right thresholds
    $maxP(T)$          returns the segment type
                       with the highest peak
output:
    $S$                segment type attribution
                       per token

 1: for $i$ from 1 to length($P$) do $S_i \leftarrow unknown$
 2: end for
 3: $types \leftarrow \{C, J, O, R\}$

 4: while $|types| > 0$ do
 5:     $t \leftarrow maxP(types)$
 6:     $p \leftarrow$ index of peak within $P_t$,    $i \leftarrow p$
 7:     while $P_{t\,i} \geq \delta_l \wedge S_i = unknown$ do
 8:         $S_i \leftarrow t$,    $i \leftarrow i - 1$
 9:     end while
10:     $i \leftarrow p + 1$
11:     while $P_{t\,i} \geq \delta_r \wedge S_i = unknown$ do
12:         $S_i \leftarrow t$,    $i \leftarrow i + 1$
13:     end while
14:     $types \leftarrow types - \{t\}$
15: end while
```

5 Evaluation

The presented approach has been evaluated using the remainder of the data set described in Section 3.3, i.e., using 600 manually tagged Austrian job ads. According to the input of the algorithms, the following parameter ranges have

				segments				overall						
$\mathbf{w}_l$	$\mathbf{w}_s$	δ_l	δ_r	$\mathbf{F}_{(C)}$	$\mathbf{F}_{(J)}$	$\mathbf{F}_{(O)}$	$\mathbf{F}_{(R)}$	**recall**	**prec**	**F**	**winDiff**	**winP**	**winR**	**winF**
25	1	0.65	0.35	65.7	73.8	54.0	71.5	65.2	70.2	**67.6**	35.0	52.8	53.9	53.3
15	1	0.6	0.3	62.7	75.8	46.6	67.5	62.2	70.2	**66.0**	35.0	50.4	50.7	50.5
50	1	0.65	0.35	67.2	61.6	62.7	70.1	70.1	61.8	**65.7**	37.1	50.8	47.0	48.8
75	1	0.65	0.35	66.0	56.3	64.3	65.6	70.6	57.1	**63.1**	38.4	48.7	43.4	45.9
CJRO-BASELINE				62.1	56.1	30.5	41.0	61.7	41.6	49.1	44.2	39.4	19.7	26.1
RND-BASELINE				25.3	21.9	21.0	22.4	29.6	20.0	23.4	44.2	39.4	19.7	26.1

Table 3: Evaluation Results of the Basic Segmentation Algorithm.

been evaluated: window length in tokens $w_l = \{15, 25, 50, 75\}$, window step in tokens $w_s = \{1, 2, ..., 10\}$, left and right thresholds $\delta_l = \delta_r = \{0.3, 0.4, ..., 0.8\}$. Typically, studies on job ads focus on analyzing the appearances of individual tokens within different segment types[6] rather than requiring exact border positions (e.g., [21, 2]). Therefore, recall, precision and F_1-score have been used as the main metrics, which are calculated based on the comparison between the correct tokens of a segment and the predicted tokens[7]. With this method, the exact locations of segment borders are only implicitly evaluated, as only tokens - regardless of their text position - are taken into account. To also get evaluation results with respect to border positions, the commonly used text segmentation metric *WindowDiff* [23] as well as its slightly altered variant *WinPR* [27] have been utilized in addition. WindowDiff yields a normalized value between 0 and 1, where 0 indicates a perfect result, whereas WinPR computes standard recall and precision values for computed segmentations. Note that WindowDiff as well as WinPR only measure the accuracy of border positions and do not reflect the correct attribution of segments.

To put the results in perspective, two baselines have been computed: (i) the *RND-BASELINE* simply divides the job ad into four equally long segments and assigns the respective categories in a random order, and (ii) *CJRO-BASELINE* also divides the text equally into four segments, but always uses the most prominent category order for attribution, which is: *company description – job description – requirements – offer.*

Table 3 shows the best results per window length. Using a length of 25, i.e., 25 tokens per window, a token-based F-score of 67% could be gained, significantly outperforming the baselines. The segmentation-based F-score of WinPR is at 53% only, which indicates that the accuracy of finding exact border positions is substantially worse than determining relevant tokens within segments.

6 Improvements

Based on the previous results from Section 5, several techniques have been applied in order to further improve the accuracy of the approach.

[6] e.g., if 'abroad' appears within the offer and/or requirement segment

[7] thereby, if a segment type does not exist in the job ad but at least one token has been predicted for it, the recall/precision is set to 0, and also vice versa.

1. An intrinsic characteristic of job ads is that bullet lists are used frequently, e.g., to express tasks or required skills. Moreover, it seems reasonable that segment types do not switch within bullet lists[8]. Therefore the algorithm is extended to be able to also detect bullet lists using regular expressions. If a bullet list is detected during the spanning of segments to the left and right from the probability peak, the algorithm steps over the whole bullet list and includes it in the current segment. By doing so, it is ensured that bullet lists are never split, regardless if the probability is below the left or right threshold or not.
2. When manually inspecting job ads, it can be observed that individual segment types often start with similar phrases. For example, the description of a company may start with *'About us:'*, *'Who we are:'* or *'[COMPANY_NAME] is a ...'*. In order to detect such typical start sequences, additional dictionaries have been created using the training data, which store all stemmed token sequences of segment beginnings. If a newline character is found within the first three tokens, only this line is used, otherwise the first three tokens are used. Having computed a dictionary for every segment type, the segmentation algorithm from Section 4.2 is then altered to utilize segment beginnings. This means, that before searching for the probability peak within a segment type, it is first assessed whether a typical start sequence could be found. If yes, the matching position is used as the beginning of the segment, from where the algorithm spans now only to the right (using δ_r) and not to the left. To decide if a start position is found, common regular expressions on stems are used which match exactly, i.e., with no tolerance. On the other hand, if also the company name is involved, the extracted name of the offering company given by the data set is utilized. Because the extracted name often does not correspond exactly to the name written in the job ad, fuzzy string matching is applied[9].

Both previously mentioned modifications have been evaluated and reveal that both can enhance the performance when applied individually[10]. By finally combining both improvement methods, an accuracy of 77% could be gained, as can be seen in Table 4. To compare the performance to existing segmentation approaches that do not only draw borders but are also capable of assigning classes to segments, a conditional random field (CRF) with standard parameter configuration has been trained. It was then utilized to label every token with one of the four categories, plus an artificial *"other"* label for tokens belonging to no category. It can be seen that the presented algorithmic approach outperforms the CRF-baseline[11]. Note that while in principle it would be possible to incorporate at least some of the algorithmic improvements also to the CRF-baseline, this task is not trivial as the CRF provides no probabilities but only labels tokens.

[8] although there are rare cases where this is the case
[9] using the Jaro-Winkler distance [33, 9] with a threshold of 0.85
[10] Due to space constraints, the individual results are omited.
[11] Note that the segmentation metrics winDiff and winPR are not applicable for the CRF-baseline, as tokens are only labeled and no actual segmentation is performed.

				segments				overall						
$\mathbf{w}_l$	$\mathbf{w}_s$	δ_l	δ_r	$\mathbf{F}_{(C)}$	$\mathbf{F}_{(J)}$	$\mathbf{F}_{(O)}$	$\mathbf{F}_{(R)}$	**recall**	**prec**	**F**	**winDiff**	**winP**	**winR**	**winF**
50	1	0.5	0.35	69.9	79.0	72.8	86.6	79.6	75.3	**77.4**	28.5	63.9	57.4	60.5
75	1	0.55	0.45	69.4	78.6	72.9	88.1	79.4	75.6	**77.4**	29.2	62.6	57.2	59.8
25	1	0.4	0.35	69.3	73.5	68.3	73.6	70.2	74.1	**72.1**	29.9	61.9	59.4	60.6
CRF-BASELINE				64.1	71.0	73.1	72.1	71.9	71.6	**71.5**	–	–	–	–
15	1	0.4	0.3	67.8	76.5	60.3	69.7	67.3	74.0	**70.5**	30.2	61.0	56.3	58.5

Table 4: Evaluation Results Including Bullet List and Start Sequence Detection.

Therefore, a different and adapted approach would have to be developed which is not addressed in this paper and left for future work.

In a final set of experiments using the hybrid approach, it has additionally been evaluated whether introducing a minimum segment length l_{min} increases the overall accuracy. By defining $l_{min} = 28$ tokens, the overall F-score could be improved slightly to 78%. On the other side, an experiment was conducted that allowed a segment to be distributed over several positions within the document[12]. This was done by spanning a segment not only from one probability peak, but from two or three peaks if an individual predefined threshold was exceeded for the latter. Optimizing this threshold it has been found that the overall accuracy could not be improved, but instead is reduced by 10%. In a final experiment, it has been evaluated whether allowing different left and right thresholds for different segment types influences the segmentation accuracy, but the results indicated that this is not the case.

7 Conclusion and Future Work

In this paper, we presented an approach to automatically segment job ads into four predefined segment types. Using machine-learning based on extracted textual features, iterative algorithms operate to detect segments. Comprehensive evaluations including several optimizations using domain specific characteristics reveal that the segment types can be identified with an accuracy of 78% in average, outperforming an out-of-the-box Conditional Random Field implementation as a baseline. The proposed method may also be applied to other languages than German and evaluated, how the accuracy changes with different data sets. Moreover, it should be evaluated whether training a model which includes not only the predefined segment types, but also a specific general type (*'other'*) can enhance the accuracy. Finally, the classifier that holds the segment type model should be tuned for an application in real-life business scenarios.

[12] which is the case for the *offer* segment, for example, if the salary is stated within the *job description*, while the other "offer" is formulated at a different position

Acknowledgments

We thank textkernel for providing the job ads, as well as Gabriela Gateva and Bianca Schönherr for coding assistance.

References

1. A. Aken, C. Litecky, A. Ahmad, and J. Nelson. Mining for computing jobs. *IEEE Software*, 27(1):78–85, Jan 2010.
2. Anonymous Authors. Anonymized Title. Working Paper, 2017.
3. E. Aureli and D. F. Iezzi. Recruitment via web and information technology: a model for ranking the competences in job market. *Proceedings of JADT 2006*, pages 79–88, 2006.
4. B. J. Jansen et al. Using the web to look for work: Implications for online job seeking and recruiting. *Internet research*, 15(1):49–66, 2005.
5. D. M. Blei and P. J. Moreno. Topic segmentation with an aspect hidden markov model. In *Proceedings of the 24th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 343–348, New York, NY, USA, 2001.
6. J. W. Budd and D. Bhave. The employment relationship. *The SAGE handbook of human resource management*, pages 51–70, 2010.
7. M. E. Califf and R. J. Mooney. Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction. *Journal of Machine Learning Research*, 4:177–210, Jun 2003.
8. F. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference (NAACL)*, pages 26–33, Seattle, Washington, USA, April 2000. Association for Computational Linguistics.
9. W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.
10. J. Eisenstein and R. Barzilay. Bayesian Unsupervised Topic Segmentation. In *Proceedings of EMNLP 2008*, pages 334–343, 2008.
11. W. J. L. Elving, J. J. C. Westhoff, K. Meeusen, and J.-W. Schoonderbeek. The war for talent? the relevance of employer branding in job advertisements for becoming an employer of choice. *Journal of Brand Management*, 20(5):355–373, 2013.
12. O. Ferret, B. Grau, and N. Masson. Thematic segmentation of texts: two methods for two kinds of texts. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 392–396. Association for Computational Linguistics, 1998.
13. Hall, Mark et al. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
14. M. A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, Mar. 1997.
15. R. Kessler, J. M. Torres-Moreno, and M. El-Bèze. E-Gen: Automatic Job Offer Processing System for Human Resources. In *Proceedings of the 6th Mexican International Conference on Artificial Intelligence*, pages 985–995, 2007.
16. P. Kuhn and H. Mansour. Is internet job search still ineffective? *The Economic Journal*, 124(581):1213–1233, 2014.

17. J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the 18th Int. Conf. on Machine Learning*, pages 282–289, San Francisco, USA, 2001.
18. R. Loth, D. Battistelli, F.-R. Chaumartin, H. de Mazancourt, J.-L. Minel, and A. Vinckx. Linguistic Information Extraction for Job Ads (SIRE Project). In *Adaptivity, Personalization and Fusion of Heterogeneous Information*, pages 222–224, Paris, France, 2010.
19. A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proc. of HLT-NAACL-Volume 4*, pages 188–191. ACM, 2003.
20. H. Misra, F. Yvon, J. M. Jose, and O. Cappe. Text segmentation via topic modeling: an analytical study. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1553–1556. ACM, 2009.
21. O. Müller et al. Towards a typology of business process management professionals: identifying patterns of competences through latent semantic analysis. *Enterprise Information Systems*, 10(1):50–80, 2016.
22. F. Peng and A. McCallum. Information extraction from research papers using conditional random fields. *Inform. processing & management*, 42(4):963–979, 2006.
23. L. Pevzner and M. A. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002.
24. J. M. Ponte and W. B. Croft. Text segmentation by topic. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 113–125, Pisa, Italy, 1997. Springer.
25. J. C. Reynar. Statistical models for topic segmentation. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 357–364, Maryland, USA, 1999. ACM.
26. M. Riedl and C. Biemann. Topictiling: a text segmentation algorithm based on lda. In *Proc. of ACL 2012 Student Research Workshop*, pages 37–42. ACM, 2012.
27. M. Scaiano and D. Inkpen. Getting more from segmentation evaluation. In *Proceedings of HLT-NAACL*, pages 362–366, Montreal, Canada, 2012. ACM.
28. F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*, pages 134–141, Edmonton, Canada, 2003. ACM.
29. A.-R. Simon, G. Gravier, and P. Sébillot. Leveraging lexical cohesion and disruption for topic segmentation. In *International Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 1314–1324, 2013.
30. E. Stamatatos, M. Tschuggnall, B. Verhoeven, W. Daelemans, G. Specht, B. Stein, and M. Potthast. Clustering by Authorship Within and Across Documents. In *Working Notes Papers of the CLEF 2016 Evaluation Labs*, Sept. 2016.
31. M. Tschuggnall, E. Stamatatos, B. Verhoeven, W. Daelemans, G. Specht, B. Stein, and M. Potthast. Overview of the Author Identification Task at PAN-2017: Style Breach Detection and Author Clustering. In *Working Notes Papers of the CLEF 2017 Evaluation Labs.* CLEF and CEUR-WS.org, Sept. 2017.
32. M. Utiyama and H. Isahara. A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 499–506, 2001.
33. W. E. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau.* Citeseer, 1999.
34. J. Zavrel, P. Berck, and W. Lavrijssen. Information Extraction by Text Classification: Corpus Mining for Features. In *Proceedings of the workshop Information Extraction meets Corpus Linguistics*, 2000.