AIResume: Automated generation of Resume Work History

Janani Balaji*, Madhav Sigdel*, Phuong Hoang, Mengshu Liu and Mohammed Korayem
CareerBuilder LLC
Greater Atlanta Area, Georgia, US
{janani.balaji,madhav.sigdel,phuong.hoang,mengshu.liu,mohammed.korayem}@careerbuilder.com

ABSTRACT

Automatic text generation has redefined the act of content generation in multiple fields such as article/news summarization, chatbots, and virtual assistants. For a person on the job market, a resume is an important piece of document that determines his rate of success in landing a job. In this paper, we introduce AIResume (AIR) - a tool that utilizes a vast knowledge base comprising of resume entries and job descriptions to generate the work history portion of a person's resume with minimal input from the user. The system starts with suggesting personalized job titles based on the employer and then goes on to mine and present the user with relevant work activities for the selected employer and job title.

1 INTRODUCTION

The advances in computing capability and connectivity have increased the usage of smart phones as mainstream operational units. However, the smaller footprint of a mobile device limits its usability as a primary text input/output device. Research has been made towards optimizing the data intake experience by automating text generation and recommendation process [11][17]. Tools like voice-based input and automatic text suggestions help improve the user experience while interacting with hand held devices.

In the recruitment space, smart phones are increasingly being used as the key devices in searching and applying for jobs. A successful job search starts with a well written and concise resume that highlights the key accomplishments - education, work history with job functions, skill set, certifications etc.,. Nevertheless, at CareerBuilder, we have observed that more than 30% of job seekers do not have a resume to start with. This, combined with the increased popularity of smart phones as job search tools, prompted us to put Natural Language Generation (NLG) and Information Extraction (IE) technology and the huge number of resumes and job descriptions we have collected over the years to use, to help the user construct his/her resume.

In this paper, we introduce AI-Resume (AIR), our resume generation tool that helps construct the work history of a resume, with minimal input from the user. The system comprises of two parts - a personalized job title suggestion and personalized work activity suggestion. The user starts the process by entering the name of the company s/he wishes to enter in the resume. The system, in response, recommends the user with job titles that are personalized to that company. These personalized job titles are generated by collecting all the job titles associated with the chosen employer and applying extensive cleaning and grouping strategies. Once the user selects a job title, the user is presented with a set of work activities that are common for the chosen company and job title. The work activities are mined from the several million user resumes and job postings using language parsing techniques and ranking. A screen shot of the tool is shown in Figure 1. The key contributions of this paper can be summarized as:

- Present AIR, an automated tool that helps build the work history section of a user resume.
- Define the challenges in recommending personalized titles per employer and explain the methodology used to extract relevant titles.
- Provide a framework to mine relevant work activities for a given employer and job title from the database of resumes and job postings.

The rest of the paper is organized as follows. We describe the overview of our system in Section 2, and Section 3 explains our personalized job title generation strategy, while Section 4 elaborates our model to extract relevant work activities for a given company and job title. Section 5 discusses the evaluation results, Section 6 briefs over the related research and Section 7 concludes the paper with directions for future research.

2 SYSTEM OVERVIEW

Figure 2 provides the overview of AIR system that consists of three main components - data sources, titles and activities extraction, and user interaction.

Data sources: Having been in the forefront of the recruitment domain, at CareerBuilder, we have hundreds of millions of resume and job posting data collected over the years. We use both the

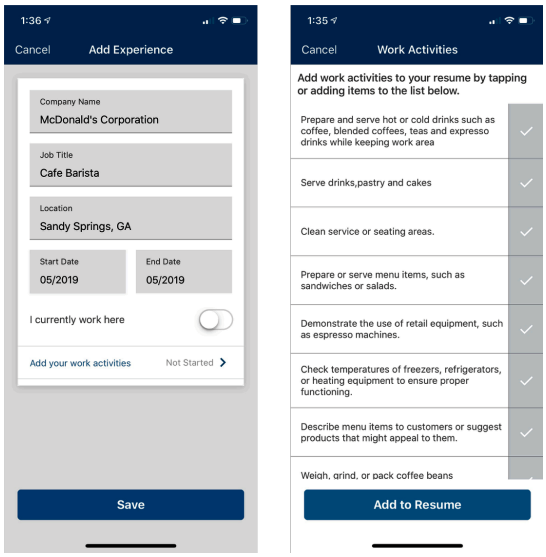

Figure 1: AI-Resume Tool

KaRS 2019 Second Workshop on Knowledge-Aware and Conversational Recommender Systems, November 3rd-7th, 2019, Beijing, China.

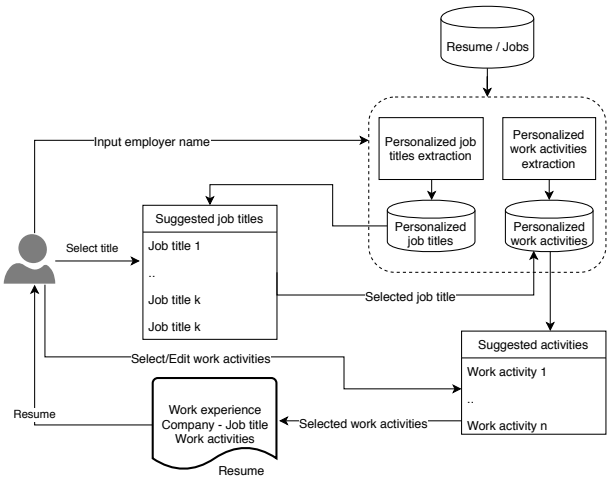

Figure 2: AIR system overview

resume and the job posting data to mine for personalized job titles and personalized work activities. We collected about 50 million work histories and 10 million job descriptions to be used to create AIR.

Job titles and activities extraction: The resumes and job postings are often whole blobs of text, and a fair bit of pre-processing is required to make the data usable. We use an in-built parsing service [29] to process the resumes and job postings and extract only the portions related to the work experience or job requirements. Likewise, we use our in-house job title classification called carotene [32] and employer name normalization [16] services to normalize the job title and employer name to normalized encodings. Once these pre-processing steps are done, we apply extensive cleaning, extraction and clustering mechanisms to generate the job titles and work activities.

User interaction: The user interacts with the system by providing the employer name and the system suggests the personalized job titles to the user. Once a user selects the job title, the system suggests personalized work activities for the selected company and job title. The user can select and edit the suggested work activities. As such, the user can easily create a resume with a detailed work experience section.

3 PERSONALIZED JOB TITLES

The goal of providing job titles personalized to an employer is to as closely as possible mimic the process of creating a resume. Though there are common job titles such as "Software Engineer" and "Customer Service Representative", we have observed that oftentimes companies like to go with a personalized job title that reflects the domain and the specific nature of work. For example, though commonly referred to as a "Cook" or a "Deli Worker", some companies in the restaurant business choose to call their food prep workers as "Sandwich Artist", thereby giving a personalized touch to the profession. It is also common to include specific departments within the job titles to differentiate the same position across different teams. "Nurse - ICU" and "Nurse - Oncology" give proper context into the nature of duties performed and such titles make more sense to go on a resume. We help the user identify the closest job title that they held in their employment by mining our database for job titles associated with the employer and then performing targeted cleaning and clustering to produce a list of job titles personalized to an employer. This section explains the challenges and methodology behind mining personalized job titles.

3.1 Challenges

The major challenge we faced in generating personalized job titles was to perform a targeted cleaning of the job title extracted from the resume or job posting. We have observed three different kinds:

Human Errors: Misspellings get introduced since human input is involved. Common job titles are often expressed using their abbreviations, e.g., *Mgr* for *Manager*, *SVP* for *Senior Vice President*, *Legal Asst.* for *Legal Assistant*. We want to resolve all variations to the complete job title as that is most widely used in professional documents.

Extraneous Information: The job titles in job descriptions are more prone to containing additional information related to the job being appended to the job title such as *company name*, *location*, specific timing information like the *shifts* involved, whether the job is *part time or full time*, etc.

Parsing Errors: These are the obvious errors when sections of the job posting other than the job title get passed in as the job title. We have sometimes even seen the entire job posting data being labeled as the job title.

These are typical noises that are routinely dealt with in text processing applications. But, what makes the cleaning process more difficult in our scenario is that what is defined as a noise varies with the kind of job title. For example, consider the job title *Communications Officer - Part-time, No Benefits*. In the context of this job title, "Communications Officer" is the true job title, whereas the rest of the text which is "Part-time, No Benefits" is noise. In routine text cleaning jobs, a list of stop words can be constructed to filter out irrelevant information. However, in our case, it is not straightforward, as the noise varies with context. Now consider the job title "Aflac Benefits Professional". In this case, *Benefits* is a legitimate word that needs to be included in the title, thus preventing the use of a global list of stop words. Due to the wide range of job titles available, constructing a comprehensive list of stop words for each category was not a feasible task. Hence we devised a combination of preliminary cleaning and clustering technologies to clean up the job titles.

3.2 Problem Definition

Given a company C and a set of job titles $J = \{J_i\}_{i=1}^{m}$, the aim of generating the personalized job titles is to find the set of titles $T \subseteq \{T_i \,|T_i = \sigma(J_i)\}_{i=1}^{m}$, where σ represents cleaning function to eliminate extraneous noise in the job titles.

3.3 Cleaning Methodology

The process of mining personalized job titles follows a two-step approach. First, a preliminary round of cleaning is performed to remove the obvious errors. We then used clustering technique to group related job titles into clusters and chose a representative from each cluster denote the respective clean title.

Preliminary Cleaning: For the preliminary cleaning, we extract the raw job title T_r, normalized employer id E, normalized

carotene code C and normalized onet code O. We start off the process by creating a set of general stop words S applicable across all occupations. These contain the common stop words like *is, are, that, was* etc.,. and also job title specific ones like *part time, seasonal, full time*. In order to handle the misspellings, abbreviations and short forms that could be present, we trained a Word2Vec[20] model and used the vector similarity to create a substitution dictionary of common misspellings to their proper forms. We then tokenized the job titles in our Carotene taxonomy into unigrams and bigrams to create a dictionary of tokens in each Carotene code, denoted d_1^C and d_2^C, respectively. Finally, we tokenized the entire job and resume corpus into unigrams and bigrams and calculated their tf-idf score on a Carotene level, denoted tf-idf$_1^C$ and tf-idf$_2^C$, respectively.

We approached the preliminary cleaning as finding the best substring of the given job title that is most relevant to the Carotene category identified. Accordingly, we tokenized the raw job title T_r into a set of tokens $\{T_r^i\}$ and formed all possible continuous subsets of length ranging from 1 to $n-1$ where $n = \|T_r^i\|$. We then scored each subset T_s and extracted the top k subsets for each title as possible candidates. The scoring method is given below:

$$S(T_s^i) = \frac{S_1(T_s^i) + S_2(T_s^i)}{2n} \tag{1}$$

$$S_1(T_s^i) = \sum_{j=1}^{\|T_s^i\|} \text{tf-idf}_1^C\left(T_{s_j}^i\right) \cdot w_{s_j}, \tag{2}$$

$$S_2(T_s^i) = \sum_{j=1}^{\|T_s^i\|} \text{tf-idf}_2^C\left(T_{s_j}^i\right) \cdot w_{s_j}, \tag{3}$$

$$w_{s_j} = \begin{cases} 2.0 & \text{if} \quad T_{s_j}^i \subseteq d_1^C \cup d_2^C \\ -1.0 & \text{if} \quad T_{s_j}^i \subseteq S \\ 1.0 & \text{otherwise.} \end{cases} \tag{4}$$

An example for the cleaning methodology is given in Figure 3. Once the job titles undergo a preliminary cleaning, the top k subsets sorted in descending order by their scores are selected for each title to be subjected to the clustering process that follows.

Title Clusters: The preliminary cleaning process removes most of the added noise. However, given the wide range of job titles present and their uneven distribution, we observed that it was necessary to augment the preliminary cleaning with a clustering strategy focused on each employer. The cleaning process described above is employer agnostic. Though it takes into account the job title classification, it does not capture all the specific employer-specific vagaries. The clustering process that follows attempts to form clusters of job titles from each company and choose a cluster representative for each cluster to be presented as the set of personalized job titles for the company. We started off by creating a graph with pair-wise distances between each raw title within each company. The distances were calculated as:

$$d(T_i, T_j) = \begin{cases} \frac{\|T_i^* \cap T_j^*\|}{\|T_i^* \cup T_j^*\|} & \text{if } m^* > 1 \\ \text{lev}(T_i, T_j) & \text{otherwise,} \end{cases} \tag{5}$$

where lev(T_i, T_j) denotes the Levenshtein [22] distance between T_i and T_j and T_i^* and T_j^* represent the tokens of T_i and T_j respectively,

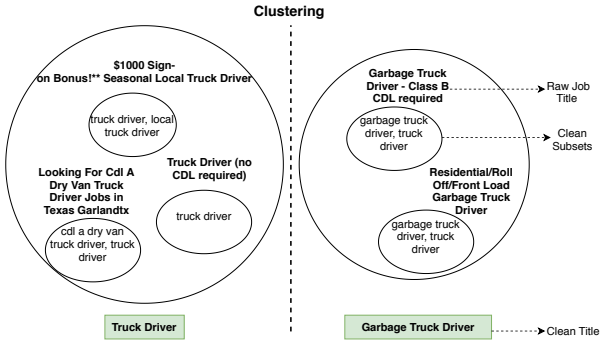

Figure 3: Two-stage Method: Preliminary Cleaning transforms raw job title to clean subsets and Clustering Stage extracts the clean job title.

and the minimum token length, m^*, is defined as,

$$m^* = \min_{i,j}(\|T_i^*\|, \|T_j^*\|) \tag{6}$$

Once the graph was formed, we used Correlation Clustering[4] to form smaller clusters that represent the same job title with minor variations in representation. After the clusters are identified, we choose a cluster representative for each cluster to denote the cleaned job title of the cluster. For the cluster representative, we chose the maximum common continuous sub-sequence among all the subsets that are part of the cluster. At the end of the clustering process, we were able to generate a list of personalized job titles for each company. An example of the clustering process and the cluster representative selection is given in Figure 3.

4 PERSONALIZED WORK ACTIVITIES

Once the user selects a company and a job title from the Personalized Job Title list (Section 3), the AIR tool generates a list of work activities that are suitable for the selected Company-Job Title combination. O*Net publishes a set of work activities that are common for each occupational code [1]. However, these are generic work activities that are collected at the occupational code level and might be too general for our use case. For example, the Occupational Code *15-1131* denotes *"Computer Programmers"* in general. However, several specialized occupations like ".Net Programmer, Computer Game Programmer, Database Engineer etc.," fall under the "Computer Programmers" category. Since we are building a resume, our aim is to provide a highly personalized set of job duties that are relevant to the job title. Furthermore, we wanted to drill down the activities to also be company specific instead of only being job title specific.

4.1 Challenges

We use an extraction-based approach to form the work activities rather than a generative approach. Our strategy was to collect all the resumes and job posting data for each company, group them by Carotene (job title), extract relevant activities and finally rank the activities for the selected Company and Carotene. As such, the majority of our challenges were in extracting the work activities from the available job descriptions.

Parsing errors: To extract the work activities, we would ideally want the source text to only contain the job duties in a job or work

[1] https://www.onetcenter.org/content.html/4.D

experience section in a resume. The source data of job postings and resumes consists of various irrelevant sections such as the company description, qualifications, benefits. In addition, the input text may contain some personal information such as phone number, email address, url. Therefore, we not only need to extract the relevant section, but also extract the phrases matching work activities and eliminate the irrelevant phrases.

Variability in the text: Since the job and resume data are coming from millions of customers and users, there is a lot of variability in the format, structure and writing of the text. Some characters are lost due to character encoding mismatch. The text might be in the form of bullet points, with missing bullet point characters and missing sentence boundaries which needs to be fixed for the activity retrieval process to work well.

4.2 Problem Definition

Let $D = \{D_i\}_{i=1}^{n}$ represent all the available job descriptions/resumes for a given company C and normalized job title T. Likewise, let S_i represent the set of likely job duty phrases from job description D_i. The complete set $S = \bigcup_{i=1}^{n} S_i$ contain all the likely work duties for the job description set D. Finally, the phrases in S are clustered and ranked to obtain the most relevant and unique set of work activities for the selected company job title. This process is applied for every company and job-title combination.

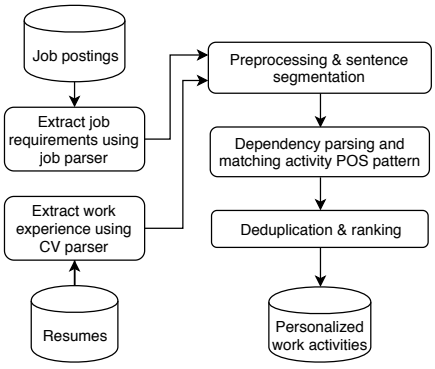

Figure 4: Work activity extraction process

4.3 Extraction Methodology

The first step in generating the work activities for a given Company-Job Title combination involves collecting the set of job descriptions/resumes belonging to the Company and Job Title in question. For each of job descriptions/resumes thus collected, we employ dependency parsing techniques and POS pattern match to extract relevant segments of the text that represent 'activities'. Finally, we score and rank the extracted work activity phrases by their relevance to the job title and select the top k work activities to be presented to the user. We used the Spacy natural language processing library [10] for the dependency parsing and the part-of-speech (POS) tagging because of its efficiency and reliability.

Figure 4 shows the complete pipeline of our work activities extraction process. We initially run through our Job/Resume parser service to extract the job requirement or work experience section from the input text. We then apply some pre-processing steps and extract phrase patterns matching work activity. This is followed by de-duplication and ranking step to recommend the most relevant and unique set of work activities to the user.

4.3.1 Activities Extraction. Here we describe the steps for extracting the work activities.

Pre-processing and Sentence Segmentation: The source resume or job text could be in HTML format. We use regular expressions to remove the HTML tags. Similarly, if the source text have identifiers such as email, phone number and url, we apply corresponding regular expressions in the text and replace these with their respective identifier tags. Later in the processing, we look for these tags and eliminate the phrases consisting such tags.

Jobs content often comes in the form of bullet points, having no defined sentence boundary markers. Oftentimes, the bullet point characters are lost due to encoding mismatch. It is important to identify and segment such lengthy text into different sentences because the reliability of an NLP system is highly dependent on the structure of the input data. We apply heuristics approach to define boundaries as follows:

- Add a sentence break after the preceding word if there is a verb starting with an uppercase, e.g., *customer satisfaction Establish daily* to *customer satisfaction. Establish daily.*
- Add a sentence break on words with lowercase followed by uppercase verb, e.g., *Coaching of employeesAnalyze call volume* to *Coaching of employees. Analyze call volume.*
- Identify bullet characters and replace them in the text with full stop. We used part-of-speech (POS) tagging and get the non-alphabetic characters preceding the verbs and identify bullets as the character group repeated the most.

Extracting activity phrase: Here, we apply the previous steps and get the list of sentences. We maintain a stop-verbs dictionary to filter non-activity verbs such as modal verbs (*have, must*) and continuity verbs (*including, following*). For each sentence, we look for the first candidate activity verb not present in the filter dictionary. We then get the dependency tree starting from this candidate verb and extract the phrase starting from this verb and ending in a noun or a pronoun. The final extracted phrase includes the tokens in the tree starting from candidate verb until noun or pronoun within minimum and maximum token threshold. We apply minimum and maximum word count thresholds of 3 and 30, respectively. Setting these thresholds allows us to get informative activities as well as to reduce noise.

Post-processing: From the list of likely activity phrases, we remove any phrases containing stop word list such as *email, phone, url* as tagged in the pre-processing step or few other tokens such as *qualifications, benefits, requirements*, commonly present as section headers in a job or a resume.

Figure 5 provides an example input text and the output along with the intermediate steps. In this example, the first likely activity VERB is the word *preparing*. The dependency tree starting from this word and ending in the noun/pronoun gives the text "Preparing/dispensing medications" as the final output.

4.3.2 Scoring, De-duplication and Ranking. Depending on the number of records, length of each text, and matching of POS patterns, we might end up with hundreds of likely work activities for each

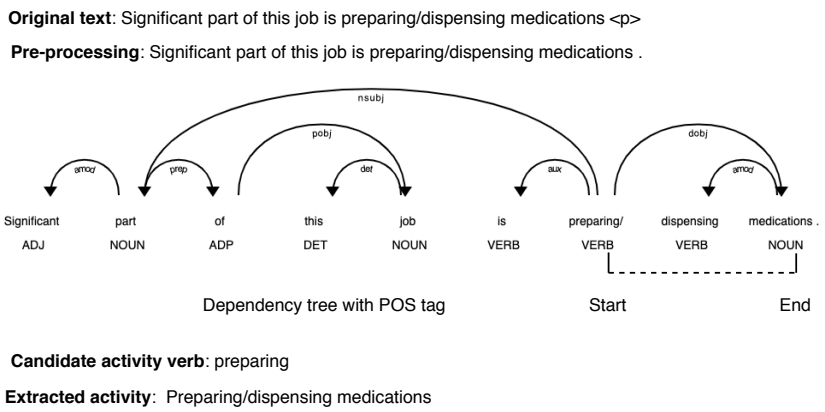

Figure 5: Example - work activity extraction

company-job title combination. Hence, we used a scoring scheme based on tf-idf to score each segment extracted and rank them based on the score. In particular, we tokenized the segment into the set of tokens $\{SegTok_0, SegTok_1, ...SegTok_n\}$ and computed the weighted tf-idf score as follows:

$$SegScore = \frac{1}{n}\sum_{i=0}^{n} \text{tf-idf}(SegTok_i), \tag{7}$$

where n is the number of tokens in the segment.

Since the activities are extracted from multiple descriptions, there may be several activities with overlapping content. To present the user with the most relevant and unique set of activities, we perform de-duplication based on overlap similarity and connected component labeling. For every pair of activities extracted for a company and job description, we calculate the overlap similarity. Let S_1 and S_2 be the set of tokens from a pair of activities, we calculate the overlap similarity between S_1 and S_2 as defined in (8).

$$s = \frac{\|S_1 \cap S_2\|}{\min(\|S_1\|, \|S_2\|)} \tag{8}$$

We use a threshold ($\tau = 0.7$) on the overlap similarity to form connected component labeling on the activities pairs. From each connected component, we select the activity with the highest tf-idf score. Thus selected activities are ranked in the decreasing order of weights and the top k are extracted to be presented to the user.

5 EXPERIMENTAL RESULTS

5.1 Personalized Titles

We evaluated the title personalization system by randomly sampling raw titles and their corresponding clean titles across job categories. We selected 405 job titles from resumes and 405 job titles from job postings for our evaluation and manually evaluated the corresponding clean titles. For each raw title, we extracted the top 5 clean titles returned from the cluster containing the raw title. We evaluate if the top clean title returned for the cluster was an acceptable clean representation, and if an acceptable clean representation was available in the top 5 results returned for the cluster.

We then calculated the accuracy as, $acc_{\text{top}N} = \|N_a\|/\|N_t\|$ where N_a and N_t are the number of raw titles that had an acceptable clean title version in its top N result and the total number of raw titles evaluated, respectively. The results are given in Table 1

Table 1: Personalized Job Titles Evaluation

Type	Accuracy - Top 1	Accuracy - Top 5
Job	0.6123	0.916
Resume	0.8826	0.9390

We observed that the accuracy was higher in job titles from resumes than those from job postings. This was because, the job titles from resumes have far less noise than those from job postings. Job titles from job postings are often corrupted with additional information about the job hours/benefits which are not present in those from the resumes.

5.2 Personalized Work Activities

To evaluate the effectiveness of our method, we randomly selected 900 descriptions (500 jobs and 400 resume profiles) corresponding to 30 most frequent carotene. For each carotene, 2-3 companies were selected with 3-4 descriptions per company and carotene. We evaluated our method using two methods: programmatically and manually:

Table 2: Work Activities Evaluation

Grade	No. of company - carotene	% of total
Good	106	51%
Ok	80	39%
Bad	20	10%
Total	206	100%

Programmatic evaluation: To evaluate the quality of extracted activities, we first wanted to determine the relevancy of extracted activities from each job description or resume input. The objective here is to determine the relevancy of activities prior to de-duplication and ranking. For this, we manually extracted all relevant job activities from the 900 job and resume profiles. We then applied the pre-processing, activity extraction and post-processing stage of our model and obtained the list of likely work activities for

each input. We evaluated the string to string matching of system activities and gold standard activities. Let S_g and S_a represent a pair of work activities from the gold standard and system extraction list, and n_g and n_a represent their respective word counts. We consider the pair a match if either string is a sub-string of the other and $\min(n_g, n_a)/\max(n_g, n_a) > 0.5$. With this programmatic evaluation, we achieve 75% precision and 55% recall. Note that this step considers all system and manual activities without ranking. There are hundreds of activities for each company job-title combination. Since we are interested in listing the top k, let's say 20 activities, 55% recall is an acceptable result. The false positives also decreases after applying ranking, de-duplication, and top k activities selection.

Evaluating activities per company-carotene: To measure the accuracy of our extracted job activities from the end-users perspective, we ran through our complete model of phrase extraction, de-duplication and ranking with the above manually annotated 900 jobs comprising 206 company carotene combinations.The extracted activities were then ranked as Good, OK or Bad. Table 2 provides the summary of our evaluation. Good indicates that all of the activities were relevant and had good sentence constructs. OK represented the group with mostly useful activities with a few irrelevant or bad phrase constructs. The users would find the recommended activities useful even if they prefer to edit the few irrelevant activities. Lastly, the Bad category is the group with majority irrelevant or badly structured phrases. This shows more than half of the results in the Good category and in total the Good's and OK's account for 90%.

6 RELATED RESEARCH

AIR is a unique tool and to the best of our knowledge, the first of its kind. The Personalized Titles suggestion is in its core an information extraction (IE) problem, where the goal is to extract the actual job title given a noisy text. The Personalized Work Activities recommendation can be linked to document summarization/information extraction, though the precise nature of the output poses additional challenges. This section glosses over the past research in the areas of information extraction, document summarization, author assistant systems and online recruitment space highlighting the differences and unique needs in our work.

Information extraction deals with identifying specific data of interest from natural-language. Classifying entities to pre-defined categories of objects such as names, places, organizations is called Named Entity Recognition (NER). Several works have dealt with this in different domains [9, 21, 23]. Most of the NER studies use the language model derived from complete sentences to identify entities [3, 18]. Rule based systems like [26, 31] use a set of lexical rules to extract entities. However, personalized titles suggestion involves working with phrases and partial sentences, thereby making the traditional NER methods unsuitable. The need for a well annotated KB also makes it difficult to apply NER principles.

Text summarization refers to the method of producing a summary with the most important information present in the source documents. Summarization can be single document vs multi-document [24]. Single document summarization uses single large text source input while multi-document summarization consists of several documents with possibly overlapping content as the input. For example, Generating a new job summary using several relevant job descriptions is a multi-document summarization problem. Summarization systems can also be categorized as extractive vs abstractive summarization [2]. The majority of prior work on summarization methods relies on extracting sections from the text and ranking them for potential inclusion in the summary. Various methods such as tf-idf and position weighting [27], recurrent-neural network approach [6], graph-based ranking [19] and clustering and optimization methods [1] have been proposed to determine sentence importance weights. There are also studies focusing on abstract summary using deep learning based summarization methods [14, 15, 30]. Our Personalized Work Activities follows extractive summarization approach.

Several work have been described in the literature on authoring assistant systems. The goal of such systems is to assist users in writing content by providing relevant suggestions. In [7], the authors describe a system to help review writers by suggesting review topics and relevant details on the topic. Similarly, the GhostWriter [5] uses case-based reasoning on existing product reviews to provide content suggestions to review writers. The AIResume system has a similar goal of suggesting the job titles and job duties to be filled in a resume. As discussed in the paper, it utilizes information retrieval, summarization and ranking techniques to make personalized recommendations. To our knowledge, there has been no prior work on such system.

In online recruitment space, machine learning techniques have been successfully applied in job search, job recommendation, and talent matching in various settings [12, 13, 25]. There have been few studies to automatically parse different sections in a job or a resume [8, 28]. These methods are very helpful for several down-stream automated applications. Having an intelligent system to suggest activities based on their employer and job title would allow the users to quickly build a detailed resume. This is what we aimed to solve via AIResume.

7 CONCLUSION

In this paper, we presented our novel AI based resume builder that helps job seekers build their resumes quickly and efficiently on their mobile devices. AIResume leverages a large dataset of job postings and resume profiles collected over several years and applies natural language extraction, extensive filtering and aggregation techniques to generate recommended job titles and work activities personalized to each company. Users find these job titles and work activities helpful for the work experience section in their resumes. This feature has been released in CareerBuilder App Store and has attracted great interest from the users.

While our current methods for AIResume provides a good baseline system, there are different areas we would like to improve upon to make the system more robust and useful. Currently, AIResume only supports building the work history section of the resume. Incorporating the other integral parts of the resume such as the educational qualifications, and skills are still a work in progress. There exists some non-activity phrases coming from irrelevant text and some noisy constructs. To this end, we would explore classification model to distinguish a phrase as a work activity or not. Likewise, we would like to explore sentence revision techniques on the extracted phrases to make the activities concise and informative.

REFERENCES

[1] Rasim M Alguliyev, Ramiz M Aliguliyev, Nijat R Isazade, Asad Abdi, and Norisma Idris. 2019. COSUM: Text summarization based on clustering and optimization. *Expert Systems* 36, 1 (2019), e12340.

[2] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919* (2017).

[3] Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in Named Entity Recognition. *Comput. Speech Lang.* 44, C (July 2017), 61–83. https://doi.org/10.1016/j.csl.2017.01.012

[4] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation Clustering. *Machine Learning* 56, 1 (01 Jul 2004), 89–113. https://doi.org/10.1023/B:MACH.0000033116.57574.95

[5] Derek Bridge and Paul Healy. 2012. The GhostWriter-2.0 Case-Based Reasoning system for making content suggestions to the authors of product reviews. *Knowledge-Based Systems* 29 (2012), 93–103.

[6] Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *Twenty-ninth AAAI conference on artificial intelligence*.

[7] Ruihai Dong, Kevin McCarthy, Michael O'Mahony, Markus Schaal, and Barry Smyth. 2012. Towards an intelligent reviewer's assistant: recommending topics to help users to write better product reviews. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*. ACM, 159–168.

[8] Shweta Garg, Sudhanshu S. Singh, Abhijit Mishra, and Kuntal Dey. 2017. CVBed: Structuring CVs usingWord Embeddings. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, 349–354. https://www.aclweb.org/anthology/I17-2059

[9] Phuong Hoang, Thomas Mahoney, Faizan Javed, and Matt McNair. 2018. Large-Scale Occupational Skills Normalization for Online Recruitment. *AI Magazine* 39, 1 (Mar. 2018), 5–14. https://doi.org/10.1609/aimag.v39i1.2775

[10] Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear* (2017).

[11] Christina L. James and Kelly M. Reischel. 2001. Text Input for Mobile Devices: Comparing Model Prediction to Actual Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '01)*. 365–371. https://doi.org/10.1145/365024.365300

[12] Faizan Javed, Qinlong Luo, Matt McNair, Ferosh Jacob, Meng Zhao, and Tae Seung Kang. 2015. Carotene: A job title classification system for the online recruitment domain. In *2015 IEEE First International Conference on Big Data Computing Service and Applications*. IEEE, 286–293.

[13] Krishnaram Kenthapadi, Benjamin Le, and Ganesh Venkataraman. 2017. Personalized job recommendation system at linkedin: Practical challenges and lessons learned. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 346–347.

[14] Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. *arXiv preprint arXiv:1708.00625* (2017).

[15] Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2018. Generative adversarial network for abstractive text summarization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[16] Qiaoling Liu, Faizan Javed, and Matt Mcnair. 2016. Companydepot: Employer name normalization in the online recruitment industry. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 521–530.

[17] Soukoreff R.,W. MacKenzie, I. S. 2002. Text entry for mobile computing: Models and methods, theory and practice. In *Human-Computer Interaction*. 147–198.

[18] Muhammad Kamran Malik. 2017. Urdu Named Entity Recognition and Classification System Using Artificial Neural Network. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 17, 1, Article 2 (Sept. 2017), 13 pages. https://doi.org/10.1145/3129290

[19] Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*.

[20] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013).

[21] David Miller, Sean Boisen, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. 2000. Named Entity Extraction from Noisy Input: Speech and OCR. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLC '00)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 316–324. https://doi.org/10.3115/974147.974191

[22] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. 2009. *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance.* Alpha Press.

[23] Raymond J. Mooney and Razvan Bunescu. 2005. Mining Knowledge from Text Using Information Extraction. *SIGKDD Explor. Newsl.* 7, 1 (June 2005), 3–10. https://doi.org/10.1145/1089815.1089817

[24] Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval* 5, 2–3 (2011), 103–233.

[25] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing Person-Job Fit for Talent Recruitment: An Ability-aware Neural Network Approach. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. ACM, New York, NY, USA, 25–34. https://doi.org/10.1145/3209978.3210025

[26] Steinberger Ralf, Pouliquen Bruno, and Ignat Camelia. 2008. Using Language-independent Rules to Achieve High Multilinguality in Text Mining.

[27] Yohei Seki. 2002. Sentence Extraction by tf/idf and position weighting from Newspaper Articles. (2002).

[28] Swapnil Sonar and Bhagwan Bankar. 2012. Resume parsing with named entity clustering algorithm. *paper, SVPM College of Engineering Baramati, Maharashtra, India* (2012).

[29] Melanie Tosik, Carsten Lygteskov Hansen, Gerard Goossen, and Mihai Rotaru. 2015. Word Embeddings vs Word Types for Sequence Labeling: the Curious Case of CV Parsing. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Association for Computational Linguistics, Denver, Colorado, 123–128. https://doi.org/10.3115/v1/W15-1517

[30] Mahmood Yousefi-Azar and Len Hamey. 2017. Text summarization using unsupervised deep learning. *Expert Systems with Applications* 68 (2017), 93–105.

[31] Wajdi Zaghouani. 2012. RENAR: A Rule-Based Arabic Named Entity Recognition System. 11, 1, Article 2 (March 2012), 13 pages. https://doi.org/10.1145/2090176.2090178

[32] Yun Zhu, Faizan Javed, and Ozgur Ozturk. 2017. Document Embedding Strategies for Job Title Classification. In *The Thirtieth International Flairs Conference*.