Ethics and Information Technology (ETIT)

DOI: http://doi.org/10.26480/etit.02.2020.99.103

ISBN: 978-1-948012-17-1

REVIEW ARTICLE

AN AUTOMATED RESUME SCREENING SYSTEM USING NATURAL LANGUAGE PROCESSING AND SIMILARITY

Chirag Daryani[a], Gurneet Singh Chhabra[b], Harsh Patel[c], Indrajeet Kaur Chhabra[d], Ruchi Patel[e]

[a,b] Department of Computer Science and Engineering, Medi-Caps University, Indore 453331, India
[d,e] Department of Computer Science and Engineering, Medi-Caps University, Indore 453331, India
****Corresponding Author Email: aldousbarrett2276@gmail.com***

ARTICLE DETAILS

Article History:

Received 25 October 2020
Accepted 26 November 2020
Available online 03 December 2020

ABSTRACT

A typical job posting on the Internet receives a massive number of applications within a short window of time. Manually filtering out the resumes is not practically possible as it takes a lot of time and incurs huge costs that the hiring companies cannot afford to bear. In addition, this process of screening resumes is not fair as many suitable profiles don’t get enough consideration which they deserve. This may result in missing out on the right candidates or selection of unsuitable applicants for the job. In this paper, we describe a solution that aims to solve these issues by automatically suggesting the most appropriate candidates according to the given job description. Our system uses Natural Language Processing to extract relevant information like skills, education, experience, etc. from the unstructured resumes and hence creates a summarised form of each application. With all the irrelevant information removed, the task of screening is simplified and recruiters are able to better analyse each resume in less time. After this text mining process is completed, the proposed solution employs a vectorisation model and uses cosine similarity to match each resume with the job description. The calculated ranking scores can then be utilised to determine best-fitting candidates for that particular job opening.

KEYWORDS

Information Retrieval, Natural Language Processing, Resume Parser, Resume Analyzer, Resume Summarisation, Ranking Candidates, e-recruitment, Cosine Similarity, Content-based Recommendation, Vector Space Model.

1. INTRODUCTION

With the rapid increase in internet connectivity, there has been a change in the recruitment process of all major companies. With the help of online job postings in various job portals and websites, recruiters are able to attract a wide variety of people for their openings. Though e-recruitment has provided convenience and savings for both the recruiters and the applicants, some new challenges arise. Large companies and recruitment agencies often receive thousands of resumes every day. This situation is even more aggravated due to the higher mobility of workers and in situations of economic distress, where many people are looking to get jobs. With less than 5% of people to be selected from these applications, it is impractical for the recruiters to manually go through each and every resume for these limited number of openings. Another problem faced by the organisations is that there is no one standard resume format used by these applicants. People come from varied fields of profession and have different backgrounds. Each one of them has had different types of education, has worked on different projects and thus has a unique style of presenting his/her credentials in the resume. Resumes are unstructured documents that come in various file formats (.pdf, .doc, .docx, .jpg, .txt etc.) and their content is not written according to standard formats or templates. This means reading resumes is not simple and thus recruiters spend a large amount of time going through the resumes for selecting the right candidates. Many job portals and external websites came up to reduce this difficulty of handling unstructured and diverse resumes. These require candidates to manually fill up all the information of their resume in an online form in a structured manner, thus creating a candidate metadata. The problem with this approach is that it requires redundant efforts on the part of the candidates, and they often miss out on filling complete information in these templates. These websites use a generic format that isn't domain-specific and thus is not optimal for all jobs. The employers then use these templates to apply the keyword-based search for shortlisting candidates. This keyword-based search functionality is insufficient to match candidates with the job description (Malinowski, Jochen, et al., 2006). This is so as it relies only on the existence of certain required keywords and has various extraction limitations like avoiding natural language semantics such as synonyms, word combinations, and contextual meaning of the content present in the resume (Singh, Amit, et al., 2010). Therefore, these Boolean search methods often give irrelevant results and deserving candidates miss out on opportunities of being shortlisted.

In order to get better results for the resume shortlisting, it is necessary to investigate more efficient approaches to candidate and job description matching. Our proposed solution will choose the best fitting candidates for a specific opportunity by relating the main features of the applicants' profile with the requirements defined in the job description. The system works in two main phases. In the first phase, all relevant candidate information like skills, work experience, years of education, certifications, etc. is extracted from the unstructured text in the resumes. The system

Quick Response Code	Access this article online	
	Website: www.intelcomp-design.com	**DOI:** 10.26480/etit.02.2020.99.103

uses Natural Language Processing to parse these relevant qualification details and then creates a summarised version of each resume (Allahyari, Mehdi, et al., 2017) irrespective of the order of content or the file format. With all the extraneous and irrelevant details removed, it becomes easy for the evaluator as he can quickly look at the summarised form and analyse the credentials of the candidates. The second phase of our system involves ranking the resumes based on the similarity of their content with the given job description. The documents are represented as vectors using Vector Space Model (Arguello and Jaime, 2013) and then similarity measures like cosine similarity (Huang and Anna, 2008) are used to measure which set of resumes are the best fitting for the particular job. In the end, a ranked list of applicants is obtained.

This paper is organised in the following sections: Section 2 describes the related work which has been done in this field. Section 3 introduces the detailed methodology and the theoretical concepts involved with our solution. Section 4 provides an insight into the system architecture we have developed. Section 5 represents the results of the experiment performed using our system and Section 6 concludes our work followed by the future scope in Section 7.

2. Related Work

The recruitment process in today's world has witnessed a major change with the evolution of technologies like the Internet. The following section summarises some of the literary work performed in this domain of e-recruitment systems. The proposed solutions use various approaches with the aim of achieving automated screening of candidates. The work presented as EXPERT (Kumaran, V.S. and Sankar, A., 2013) proposed the use of ontology mapping for screening candidates for the given job description. It included three phases of operation which were the creation of candidate ontology, construction of job criteria ontology document and then finally mapping of both of these to evaluate which candidates are eligible for the job. In 2012, an automated job screening system was proposed (Faliagka, Ramantas, Tsakalidis, and Tzimas). It discusses different machine learning algorithms and uses Support Vector Regression to create a list of ranked candidates for the given job. Another work presented (Weathington and Bechtel, 2012) that described how social media (e.g. LinkedIn, Facebook, etc.) information of the applicants can be used for recruitment decisions. In another approach, the work that was proposed (Laumer, S. and Eckhardt, A., 2009) described a collaborative filtering based system to recommend applicants that best fit a job. We also studied a work (Malinowski, Weitzel, and Keim, 2008) that considered matching interpersonal compatibility of the team members with the prospective hire to make recruitment decisions. Our work takes a different approach as it focuses mainly on the content of the resumes where we perform the extraction of skills and related parameters to match candidates with the job descriptions.

3. Methodology

In this section, we describe the concepts that facilitate the construction of the proposed Automated Resume Screening System. The system works in two phases as described below.

3.1 Information extraction

The first phase of our proposed system involves information extraction using Natural Language Processing. The information in the resumes is not present in a structured format. There are noises, inconsistencies and irrelevant bits of data which is of no use to the recruiters. The objective is to derive relevant keywords from the unstructured textual data in the resume without any need of human crawling efforts. Using techniques like Tokenization, Stemming, POS Tagging, Named Entity Recognition, etc., our system obtains important job-related content (skills, experience, education, etc.) from the uploaded candidate resumes. The result is a summarised version of each resume in a JSON format which can be easily used for further processing tasks in the next phase of this resume screening system.

3.1.1 Tokenization

After converting the various resume formats (.docx, .pdf, .jpg, .rtf, etc.) into text, we begin the tokenization process to identify terms or words that form up a character sequence. This is important as through these words, we will be able to derive meaning from the original text sequence. Tokenization involves dividing big chunks of text into smaller parts called tokens. This is done by removing or isolating characters like whitespaces and punctuation characters. Tokens are sentences initially (when tokenized out of paragraphs) and then are further split into individual words. By performing Tokenization, we can derive information like the number of words in a text, frequency of a particular word in the text and much more. The tokenization can be performed in multiple ways such as using Natural Language Toolkit [NLTK], the spaCy library, etc. Tokenization is a mandatory step for further text processing such as removal of stop words, stemming and lemmatization.

3.1.2 Stemming and lemmatization

It is frequently seen that a single word of the English language is used in various different forms in different sentences according to its grammatical rules. For example -implement, implemented and implementing are just different tenses of the same verb. This situation results in the need to reduce all the altered or derived forms of a word to their central stem or base so that these derivationally related words with similar meanings are not considered to be different from each other. Both Stemming and lemmatization have the same objective but differ in their approach.

"Stemming is the mechanism of reducing inflected or derived words to their word root, or stem. It is a crude heuristic process that involves chopping off the ends of words to achieve this objective, and often includes the removal of derivational affixes" (Jivani, A.G., 2011). These are rule-based algorithms in which a particular word is tested on a range of conditions and then based on a list of known suffixes, decides how to cut it down. It is noteworthy that the root derived after stemming may not be identical to the morphological root of the word. Due to the heuristic-based approach of stemming, it suffers from issues such as under-stemming and over-stemming. Some common stemming algorithms used are Porter-Stemmer, Snowball stemmer, and Lancaster stemmer. On the other hand, lemmatization is the process of utilising a language dictionary to perform an accurate reduction to root words. Unlike Stemming which simply cuts off tokens by simple pattern matching, lemmatization is a more careful approach that uses language vocabulary and morphological analysis of words to give linguistically correct lemmas. This means lemmatization utilises the knowledge of context and therefore can differentiate between words that have different meanings based on parts of speech. For the English language, our system uses the WordNet Lemmatizer (based on WordNew Database) provided by the NLTK python package.

3.1.3 Parts of speech (POS) tagging

It is a process of assigning grammatical information to a word based on its context and its relationship with other words in the sentence (Gelbukh, 2014). The part-of-speech tag specifies whether the word is a noun, pronoun, verb, adjective, etc. according to its usage in the sentence. It is important to assign these tags so as to understand the correct meaning of a sentence and for building knowledge graphs for named entity recognition. This process is not as simple as mapping a word to their corresponding part of speech tags. This is so as a particular word may have a different part of speech based on different contexts in which it is used. For example: In the sentence "I am building a software", building is a Verb, but in the sentence "I work in the tallest building of that street", building is a Noun. Also called grammatical tagging or word-category disambiguation, it is a supervised learning solution that analyses the features such as the preceding word, following word, first letter capitalized or not, etc. to label the words after tokenization. Rule-Based POS tagging, Stochastic POS tagging, and Transformation based tagging are mostly used (Hasan, 2006).

3.1.4 Chunking

Chunking is a process that aims to add more structure to sentences by grouping short phrases with parts of speech tags. Because parts of speech tags alone cannot give information about the structure of the sentence or the actual meaning of the text, chunking combines parts of speech tags with regular expressions to give a result as a set of chunk tags like Noun Phrase (NP), Verb Phrase (VP), etc. Also called Shallow Parsing, it involves the construction of a parse tree that can have a maximum one level of information from roots to leaves. This ensures there is more information than just part of speech of the word without needing to create a full parse tree. Chunking segments and labels multi-token sequences (Bird, Klein and Loper, 2009), mostly making groups of "noun phrases" that are used for finding named entities.

3.1.5 Named entity recognition

Named Entity Recognition is an information extraction technique which extracts relevant information by classifying chunks of unorganized text into predefined categories like names of persons, companies, contact info, educational credentials, and skills. After classifying the unstructured resume data into such different sets of categories, our aim is to use a similarity model to determine the similarity between the categorized resume data and the requirements provided by the recruiters. There are many approaches to implement the Named Entity Recognition (Mansouri, A., Affendey, L.S. and Mamat, A., 2008) in order to derive relevant categories from unstructured data. These include the Rule-Based approach in which we define our own algorithms according to the required domain. We can also use regular expressions, which finds patterns in a string to detect the named entities. Another approach is using Bidirectional-LSTM with the Conditional Random Field algorithm for named entity recognition as a sequence labelling problem (Huang, Z., Xu, W. and Yu, K., 2015).

We have used the spaCy module which consists of various pre-trained models that can recognize a number of default entities from the content of the documents. These models use language information to detect these entities. We also trained the model on a large annotated set of resume samples for better accuracy in the entity recognition. We could detect entities like name, phone number, email, educational institute, organisation etc. from the resumes as shown in figure 2.

3.2 Content based candidate recommendation

The second phase of our proposed system aims to build a content-based recommendation system (Guo, X., Jerbi, H. and O'Mahony, M.P., 2014) that utilises the extracted entities from phase 1 to recommend the most appropriate resumes for the given job description. The system employs concepts like Vectorisation (Salton, G., Wong, A. and Yang, C.S., 1975), importance or weight assigning techniques like TF-IDF (Jabri, Siham, et al., 2018) and similarity measures like cosine distance (Huang and Anna, 2008) for calculating the similarity among the contents of the documents.

3.2.1 Vectorization

Vector space is a geometric structure formed by a set of elements called vectors. These can be added together and can be multiplied (“scaled”) by certain numbers, called scalars in this context. It is an algebraic model for representing text information for Information Retrieval, Natural Language Processing and Text Mining. Representing documents in a vector space model is called vectorisation. It is the process of turning a document into a numerical vector. An important reason behind performing vectorisation is that most machine learning models require the input to be numerical vectors rather than strings. A common way of vectorising text is to map every possible word to a specific integer. If we have a large array then every word fits into a unique slot in the array. The value at that index is the number of times the word occurs. Generally, our array size is less than the corpus vocabulary. We should thus have a vectorisation strategy to account for this.

3.2.2 TF-IDF

TF-IDF stands for “Term Frequency – Inverse Document Frequency” (Stecanella, 2020). The TF-IDF weight is often used in text mining techniques. TF-IDF was invented for information retrieval and document search. This weight is a numerical measure to determine how important a term is with respect to a document in a collection or corpus. The importance increases proportionally to the frequency of a word within the document but is offset by the number of documents that contain the word. So, terms that are frequently used in every document, such as this, and, what, whom, is, the, if, etc. rank low even though they may appear many times since they don’t mean much to that document in particular (Stecanella, 2020). The TF-IDF value for a term in a document is calculated by multiplying two different metrics (Stecanella, 2020) as shown in equation (1) below.

$$TF-IDF\,(t,d) = TF\,(t,d) * IDF\,(t,d) \qquad (1)$$

- Term Frequency: It measures how frequently a word occurs in each document in the corpus. Since a word may occur more number of times in lengthy documents than shorter ones, so you need to adjust or normalize this frequency. A normalized term frequency is calculated by dividing the number of times a term appears in a document by the total number of terms in that document. Mathematically, we can write it as (Jabri, Siham, et al., 2018) shown below in equation (2).

$$TF\,(t,d) = \frac{freq\,(t,d)}{\sum_i^n freq\,(t_i,d)} \qquad (2)$$

Here, freq (t, d) is the count of the instances of the term t in document d, TF (t, d) is the proportion of the count of term t in document d, and n is the number of distinct terms in document d.

- Inverse Document Frequency: It measures how important a word is for all documents in the corpus. In other words, this metric helps to know how rare or common a word is across in the corpus. It weighs down the terms that occur more often while scaling up the rare terms. The terms that appear more often in the set of documents have IDF value close to 0 while the rare terms have a high IDF. It is calculated by dividing the total number of documents by the number of documents that contain a term and then calculating the logarithm (Stecanella, 2020). Mathematically, we can write it as shown below in equation (3).

$$IDF(t) = log\left(\frac{N}{count(t)}\right) \qquad (3)$$

Here, N is the number of distinct documents in the corpus and count (t) is the number of documents in the corpus in which the term t is present.

The product of these two metrics i.e. equation (2) and (3) results in a TF-IDF score of a word in a document. More relevance of a word in a document is reflected by its high TF-IDF score. In our system, we modelled the resumes and the job description document into a vector space. This is done by creating a dictionary of terms present in the documents and converting each term to a dimension in the vector space. We then computed the TF-IDF matrix for the CVs and the job query by using the CountVectorizer and the TfidfTransformer python modules. In the next step, we need to calculate the similarity score between the resumes and the job description.

3.2.3 Cosine similarity

A Similarity measure is a metric that determines how much the two objects are alike. Cosine similarity (Sidorov, Grigori, et al., 2014) is a measure to find how similar the two documents are regardless of their size. It represents the orientation of the documents when plotted on an N-dimensional space, where each dimension depicts the features of the object. It’s a symmetrical algorithm, which implies that the results from computing the similarity of item X to item Y is equal to computing the similarity of item Y to item X. Mathematically, we can represent it as shown below (Sidorov, Grigori, et al., 2014) in equation (4).

$$cos(\theta) = \frac{\vec{a}.\vec{b}}{\left|\left|\vec{a}\right|\right|\left|\left|\vec{b}\right|\right|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2}\sqrt{\sum_{i=1}^n b_i^2}} \qquad (4)$$

Here, $\vec{\mathbf{a}}.\vec{\mathbf{b}} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$ is the dot product of the two vectors. Using this formula, we calculate the cosine similarity between all pairs of elements. It can then be used to rank the resume documents with respect to a given vector of query words. However, cosine similarity focuses on features that are related to the text’s words only and will give less accurate results. The efficiency of similarity measures can be improved by the inclusion of semantic information. This will constitute the future scope for our automated resume screening system.

4. SYSTEM ARCHITECTURE

The fundamental solution to the problem is building a content-based job recommendation system that uses the Vector Space Model (VSM) in calculating the similarity between the content of the candidate resumes and the job requirements to recommend the best fitting candidates to the employer. At a broader level, the basic process of this system is as follows.

The foremost step is text mining or in other words, information retrieval from the unstructured resume documents. The next step is feature selection. In this, we identify the main features of the job requirements and the candidate profiles (resumes) which will be used for the matching process. We then employ the vector space model to represent these selected features in an appropriate form by converting both the job description and the resume documents as vectors. Finally, we use similarity measures like cosine similarity to calculate the ranking of the resumes to recommend the top ‘N’ candidates for the given job. This process of the system is illustrated in figure 1 given below.

4.1 Vector Space Model

After the extraction of relevant keywords from the text corpus, we need to transform these terms (part of our created vocabulary) into a numeric form that the machine can understand.

The vectorised form of the text can then be used as input for further processing tasks like calculation of text similarities. This task of representing each document in the form of vectors is performed using the

Vector Space Model. It is simple to use and hence is commonly used in information retrieval and content-based recommendation systems.

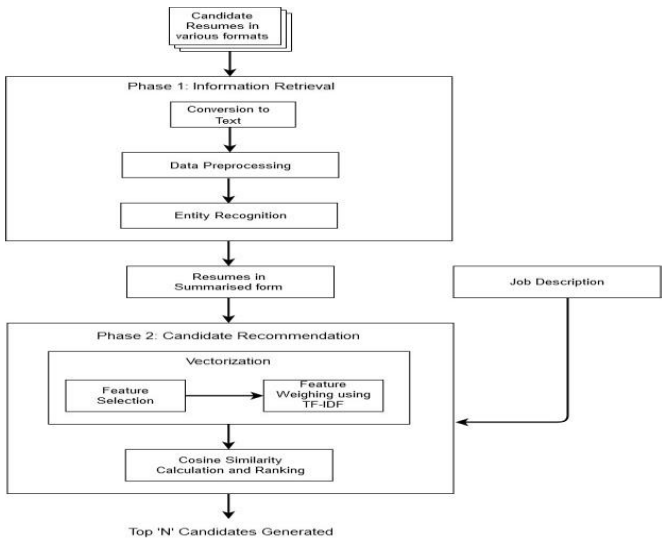

Candidate Resumes in various formats
Phase 1: Information Retrieval
Conversion to Text
Data Preprocessing
Entity Recognition
Resumes in Summarised form
Job Description
Phase 2: Candidate Recommendation
Vectorization
Feature Selection
Feature Weighing using TF-IDF
Cosine Similarity Calculation and Ranking
Top 'N' Candidates Generated

Figure 1: Architecture Diagram of our System

4.2 Candidate Recommendation Process using Vector Space Model

The first step we perform after information retrieval is feature selection for each item. In our content based job recommendation system, we build the job profile as well as the profile of each of the candidates (resumes) that apply for the given job position. We select the most relevant features in the candidate resume that are needed for the job and thus create the candidate profile. Using the vector space model representation, we can write each resume as a vector of n-dimensions where the number of dimensions depends on the total textual features from the collection. We create fixed-size vectors so that all resumes can be compared on the same set of features.

Let $R = (R1, R2, \ldots, RN)$ and $F = (f1, f2, \ldots, fM)$ denote a set of resumes and the set of features respectively. We can represent each Resume R_a as a vector of feature weights, where each weight w_i^a represents the importance of the particular feature F_i for the resume R_a. This is shown below in equation (5).

$$R_a = (w_1^a, w_2^a, \ldots\ldots\ldots, w_n^a) \tag{5}$$

As described in section 3.2.2, our system assigns the feature weights using TF-IDF. These are term frequency scores in which a higher score denotes that the word is frequent in a particular document but is rare across the documents. This representation of feature weights is shown below in equation (6).

$$\begin{aligned} w_i^a = TF - IDF\ (F_i, R_a) &= TF\ (F_i, R_a) * IDF\ (F_i) \\ &= TF\ (F_i, R_a) * log\ (N/Ni) \end{aligned} \tag{6}$$

Here, $TF(F_i, R_a)$ is the number of occurrences of feature F_i in the content of resume R_a, N_i is the number of resumes that contain the feature F_i, and N is the total number of resumes.

In a similar manner, we create the job profile from the most important requirements given in the job description and hence get the job query vectorised document. We have thus created a vector space where each point represents an applicant's resume or the job query.

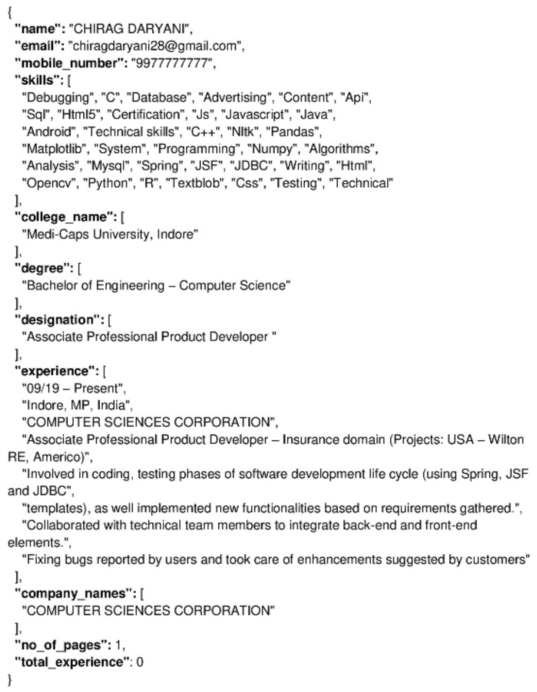
The next step in this job recommendation system is the similarity calculation between each set of resume vectors and job query vectors. The cosine similarity between Resume R_a, and job query J_b when each of them is represented in vector space containing 'n' features is given by the following equation.

$$cosine\_similarity(R_a, J_b) = \frac{\sum_{i=1}^{n} (w_i^a \times w_i^b)}{\sqrt{\sum_{i=1}^{n} (w_i^a)^2} \times \sqrt{\sum_{i=1}^{n} (w_i^b)^2}} \tag{7}$$

We then rank each resume according to its similarity score with the job query. Thereafter, the top-ranked resumes are recommended to the employer.

5. Results

For testing the system, we have used the job description posted by Amazon.com Inc. inviting applicants for the job position of a Software Developer Engineer at its Bengaluru office. We have taken some relevant resume samples from the Internet which we'll pre-process, perform extraction, summarise and then calculate cosine similarity on to create a ranked list of candidates for this job. For feature selection, we have selected parameters such as Educational Degree, University, Total Experience, Designation with the Organisation in which the candidate has worked in the past, and most importantly the skills that are needed for the job. Figure 2 given below shows the sample output after the information extraction from a resume is performed successfully.



```
{
  "name": "CHIRAG DARYANI",
  "email": "chiragdaryani28@gmail.com",
  "mobile_number": "9977777777",
  "skills": [
   "Debugging", "C", "Database", "Advertising", "Content", "Api",
   "Sql", "Html5", "Certification", "Js", "Javascript", "Java",
   "Android", "Technical skills", "C++", "Nltk", "Pandas",
   "Matplotlib", "System", "Programming", "Numpy", "Algorithms",
   "Analysis", "Mysql", "Spring", "JSF", "JDBC", "Writing", "Html",
   "Opencv", "Python", "R", "Textblob", "Css", "Testing", "Technical"
  ],
  "college_name": [
   "Medi-Caps University, Indore"
  ],
  "degree": [
   "Bachelor of Engineering – Computer Science"
  ],
  "designation": [
   "Associate Professional Product Developer "
  ],
  "experience": [
   "09/19 – Present",
   "Indore, MP, India",
   "COMPUTER SCIENCES CORPORATION",
   "Associate Professional Product Developer – Insurance domain (Projects: USA – Wilton
RE, Americo)",
   "Involved in coding, testing phases of software development life cycle (using Spring, JSF
and JDBC",
   "templates), as well implemented new functionalities based on requirements gathered.",
   "Collaborated with technical team members to integrate back-end and front-end
elements.",
   "Fixing bugs reported by users and took care of enhancements suggested by customers"
  ],
  "company_names": [
   "COMPUTER SCIENCES CORPORATION"
  ],
  "no_of_pages": 1,
  "total_experience": 0
}
```

Figure 2: This image presents the JSON output generated by our system after the completion of the information extraction process (phase 1) on a sample resume

The next phase (phase 2) of our system involves vectorization and similarity calculation. The results of the calculated cosine similarity measure between each of the four resumes and the job query is as follows:

- cossimilarity(**resume1**, **jobquery**, **similarity_matrix**)= **0.4907052756267933**
- cossimilarity(**resume2**, **jobquery**, **similarity_matrix**)= **0.6802823482591744**
- cossimilarity(**resume3**, **jobquery**, **similarity_matrix**)= **0.49850131321205904**
- cossimilarity(**resume4**, **jobquery**, **similarity_matrix**)= **0.6514716047844277**

The table below presents the ranked list of candidates according to the calculated cosine similarity values.

Table 1: Resultant ranked list of candidates prioritized by similarity score

Candidate Number (Resumes)	Cosine Similarity Score	Rank for the Job
Candidate 2	0.6802823482591744	1st
Candidate 4	0.6514716047844277	2nd
Candidate 3	0.49850131321205904	3rd
Candidate 1	0.4907052756267933	4th

Based on the results, we can see candidate 2 best fits the job posting followed by candidate 4. The candidates 3 and 1 are the least appropriate candidates in this sample.

6. Conclusion

In this paper, we presented an automated resume screening system that simplifies the e-recruitment process by eliminating the various problems faced by the recruiters as they relied on manual shortlisting of applicants for a given job position. Our system works on two fronts. Firstly, it uses Natural Language Processing to extract relevant information from the unstructured and wide-ranging formats of the resumes. It creates a summarised version of each resume which has only the entities that are pertinent to the selection process. With all the insignificant information removed, the task of the screening officials is simplified, and they can better analyse each resume with better efficiency. On the other front, our system provides the provision of ranking the applicants by using a content-based recommendation that uses the Vector Space Model and similarity to match the extracted resume features with the requirements in the job description. It calculates the similarity score value for each resume and thus creates a ranked list of top-N recommended candidates that best fit the particular job opening.

7. Future Work

Future work for this system includes mining social networking data (e.g. Facebook, LinkedIn, GitHub profiles) of the candidates and utilising this social behaviour information in combination with resume content to make even more improved recommendations. Another possibility is using a collaborative filtering based approach that can match the current applicant with a job according to how well other similar candidates (neighbours) are rated for it. Another scope of future work lies in the use of Latent Semantic Analysis (Berry, M., 2001) in the calculation of semantic similarity between the documents and then comparing it with the results of the term frequency based similarity approach.

Acknowledgements

This research was supported by Medi-Caps University, Indore. We would like to sincerely thank Prof. Indrajeet Kaur Chhabra, our guide, for her constant support and guidance. We are also grateful to Prof. Ruchi Patel for comments that greatly improved the manuscripts. We would also like to thank all anonymous reviewers for their valuable feedback.

References

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K., 2017. A brief survey of text mining: Classification, clustering and extraction techniques. arXiv preprint arXiv:1707.02919.

Arguello, J., 2013. Vector space model. Information Retrieval September, 25.

Berry, M., 2001. Computational Information Retrieval. Philadelphia: Society for Industrial and Applied Mathematics,121-144.

Bird, S., Klein, E. and Loper, E., 2009. Natural Language Processing With Python. Bejing: O'Reilly, 264.

Faliagka, E., Ramantas, K., Tsakalidis, A. and Tzimas, G., 2012, May. Application of machine learning algorithms to an online recruitment system. In Proc. International Conference on Internet and Web Applications and Services.

Gelbukh, A., 2014. Computational Linguistics And Intelligent Text Processing. Berlin, Heidelberg: Springer Berlin Heidelberg.

Guo, X., Jerbi, H. and O'Mahony, M.P., 2014, September. An analysis framework for content-based job recommendation. In 22nd International Conference on Case-Based Reasoning (ICCBR), Cork, Ireland, 29 September-01 October 2014.

Hasan, F.M., 2006. Comparison of different POS tagging techniques for some South Asian languages (Doctoral dissertation, BRAC University).

Huang, A., 2008, April. Similarity measures for text document clustering. In Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, 4, 9-56.

Huang, Z., Xu, W. and Yu, K., 2015. Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991.

Jabri, S., Dahbi, A., Gadi, T. and Bassir, A., 2018, April. Ranking of text documents using TF-IDF weighting and association rules mining. In 2018 4th International Conference on Optimization and Applications (ICOA), 1-6. IEEE.

Jivani, A.G., 2011. A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl, 2(6), 1930-1938.

Kumaran, V.S. and Sankar, A., 2013. Towards an automated system for intelligent screening of candidates for recruitment using ontology mapping (EXPERT). International Journal of Metadata, Semantics and Ontologies, 8(1), 56-64.

Laumer, S. and Eckhardt, A., 2009, May. Help to find the needle in a haystack: integrating recommender systems in an IT supported staff recruitment system. In Proceedings of the special interest group on management information system's 47th annual conference on Computer personnel research, 7-12.

Malinowski, J., Keim, T., Wendt, O. and Weitzel, T., 2006, January. Matching people and jobs: A bilateral recommendation approach. In Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), 6, 137c-137c. IEEE.

Malinowski, J., Weitzel, T. and Keim, T., 2008. Decision support for team staffing: An automated relational recommendation approach. Decision Support Systems, 45(3),429-447.

Mansouri, A., Affendey, L.S. and Mamat, A., 2008. Named entity recognition approaches. International Journal of Computer Science and Network Security, 8(2), 339-344.

Salton, G., Wong, A. and Yang, C.S., 1975. A vector space model for automatic indexing. Communications of the ACM, 18(11),613-620.

Sidorov, G., Gelbukh, A., Gómez-Adorno, H. and Pinto, D., 2014. Soft similarity and soft cosine measure: Similarity of features in vector space model. Computación y Sistemas, 18(3),491-504.

Singh, A., Rose, C., Visweswariah, K., Chenthamarakshan, V. and Kambhatla, N., 2010, October. PROSPECT: a system for screening candidates for recruitment. In Proceedings of the 19th ACM international conference on Information and knowledge management, 659-668.

Stecanella, B., 2020. What Is TF-IDF?. [online] MonkeyLearn Blog. Available at: <https://monkeylearn.com/blog/what-is-tf-idf/>.

Weathington, B.L. and Bechtel, A.R., 2012. Alternative Sources of Information and the Selection Decision Making Process. Journal of Behavioral & Applied Management, 13(2).