IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A REVIEW ON KNOWLEDGE EXTRACTION FOR AUTOMATED RECRUITMENT PROCESS USING NLP

[1]Sushit Kumar, [2]Shalini Bhadola, [3]Kirti Bhatia

[1]M.Tech Student, [2]Assistant Professor, [3]Assistant Professsor
[1]Computer Science & Engineering,
[1]Sat Kabir Institute Of Tehnology & Management, Bahadurgarh, Haryana, India

Abstract: CV/Resumes taken from different professions use different formats, fonts, styling and structuring and vocabulary. We are focusing the concept of Automated Recruitment process and the mail logic behind it is to provide ease to Recruitment Process with the help of Technology. In this paper we are using concept to Text-Mining, Pdf-Mining, tokenization, parts of speech and many other steps which would require to build algorithm for our thesis. Tokenization is a key step in the process of information extraction from the given input CVs, the tokenizer module of natural language processing libraries are compared with regard to per- formance, throughput and applicability. At the later stages the dissertation will look into the inner workings of open-source parsers, measures their precision and recall, comparing the result to the previous CVs. The performance of three commercial resume parsers were also evaluated. Lastly the steps of an iterative development are discussed, by which a new resume parser was developed, where the emphasis was on using the resume's structural and visual information, rather than relying only on its raw text.We will evaluate the performance of newly developed resume parser is measured, and compare its output to another set of pre-annotated resumes. Each step of the parsing process will be examined closely and recommendations will made for future researches.

1. ANALYSIS ON THE PREVIOUS RESEARCH WORKS DONE

1.1 Open-source resume parsers

1.1.1 Brendan Herger's resume parser

The program was implemented in Python and also the process breaks all the way down to the subsequent steps:

- Text extraction from the PDF file using the PDFMiner library, which was previ- ously discussed in Section 2.2.1.2.
- Text cleaning using Regexps: since the converted text contains escape sequences (\n, \t etc.) and other special characters (such as bullet-points utilized in the resume), the text is cleaned from these expressions.
- Identification of e-mail address, signaling, and postal address using complex regular expressions.
- Counting the quantity of occurrences of programming languages using basic regular expressions.
- Writing the results to a CSV file.

Advantage of this approach is simplicity, because it takes almost no time to read through and understand the ASCII text file. Also if the end-user wants to feature additional programming languages to count, the code is simple to change, but provided that the user is acquainted with Python and regular expressions. Finally such an approach is additionally language independent, meaning that it works both on English and German resumes.

On the opposite hand this solution assumes, that there's a linear relation between the amount of occurrences of a programing language and also the applicant's capability to use this language. However this assumption can be flawed, if the applicant writes an in depth text on an old work experience (thus mentioning programming languages more frequently), meanwhile keeping the descriptions of current work experiences shorter. a significant disadvantage of the parser the data loss on start-date, end-date and verbal description.

1.1.2 Skcript Technologies' resume parser

Technologies has made the resume scanner openly available on their GitHub page2. Their solution focuses on extracting skills, organizations, qualifications and job titles. Text extraction from PDF files is implemented using the PDFMiner library, with additional text cleaning. However here a instruction Interface (CLI) is given to the end-user to feature or remove values from the varied fields (skills to seek out, known organizations), thus the user doesn't must touch the source-code to feature new search values. The names of the known fields are saved locally on the pc, thus the user doesn't must add them on each session. The rummage around for skill names is implemented using Regexps. Interestingly the developers intended to acknowledge the length of every work experience (measured in years), however because of the mediocre date recognition algorithm it fails most of the time in practice.

In conclusion Skcript's resume parser could be a dictionary based tool that tries to parse out skills and work experience of candidates. However thanks to the big dictionaries, that got to the user by default, the software captures too many irrelevant words listed as skills, like "video", "adobe", "fluent", "other". Finally the tool tries to quantify the work experience of the applicant in years, however in practice it fails most of the time.Antony Deepak's Resume Parser

Unlike the previously discussed resume parsers Antony Deepaks' solution was written in Java using the Gate framework's3 "ANNIE" plugin4 for text analysis. The text is extracted from the PDF files using Apache's Tika library. The recognition of work experience and skills is based on Gate's rule-based "JAPE" language5, that is "a Java compatible language used in GATE to apply rules to text annotations" (Rosier, Burgun, & Mabo, 2008). As seen later the advantage of such an approach is the enhanced accuracy. On the other hand the system designed by Deepak is rather complex: the combination of different frameworks (Gate, ANNIE, JAPE language) makes it more difficult for newcomers to dive into the project and fully understand the code behind it.

1.1.3 Conclusion

After investigating the openly available parsers, the subsequent is concluded:

• Namely all of the resume parsers lose structural information like font-size, font-color or tabbing and thus lose the flexibility to exactly identify sections (skills section, work experience section), in addition as individual work experiences.

• The resulting output text still contains HTML and Unicode characters, which could be disturbing for the top user. Also such expressions can have a negative impact on NER, that's performed later while recognizing dates.

• The parsers that specialize in job extraction are only ready to handle resumes written in English. Dynamic language detection would be an excellent feature to possess.

• Applying a more sophisticated, but significantly tougher extraction process (as drained Deepak's parser) isn't definitely beneficial when developing open- source projects, because it takes longer and expertise to grasp the underlying code.

• Allowing the user to feature new keywords (programming languages or libraries) should be done via a Graphical programme (GUI) or a minimum of via the program line interface, but definitely not by modifying, and hard-coding these values within the source-code itself. New keywords should be stored locally, in order that the user doesn't need to add these on each new session.

• The quality of the talents dictionary is additionally important. it's better letting the user to create his own dictionary, instead of providing an outsized skill dictionary by default, that may cause identifying irrelevant and meaningless words as skills.

1.2 Measurement

1.2.1 Evaluation approach

The goal of the evaluation was to live the precision and recall with relation to the extraction of labor experience and skills of open-source resume parsers. This was achieved by feeding 25 resumes to the parser as input and manually comparing its output with the first resume, while counting the amount of correctly identified elements (positives), the amount of incorrectly identified elements (false positives) and also the number of incorrectly unidentified elements (false negatives). for every element within the resume, the result can be one (or multiple) of the subsequent outcomes:

• Work experience: – Start date and end date:

* True positive, if the extracted date is that the same, because the one within the resume.

* False positive, if the date extracted differs from the one within the resume.

* False negative, if a date, that ought to are extracted wasn't found by the parser.

– Work experience description (text)

* True positive, if the extracted text contains only relevant information (names of technologies and programming languages) that were actually mentioned within the description resembling the work experience. (Thus it's not considered miscalculation, if a text that originally didn't belong to the work description is now identified as work experience, as long because the additional text doesn't contain relevant information. for instance it wasn't counted as a mistake, if the following job's location/organization was merged into the previous job's description.)

* False positive, if additional text was taken as description that contains relevant information, that didn't originally belong to this work experience.

* False negative, if there was a block of text containing relevant informa- tion, that wasn't identified as work description.

• Skills– Some parsers simply take the initial resume's skill section and paste it into the output's skill section, while others perform a skill searching process and only write the found skills to the output. While the primary approach leaves the burden of skill identification to the top user, it doesn't lose any information. On the contrary, within the second version the found skills are given to the user as an inventory, but the chance of losing information is significantly higher. for instance, if a skill is left undetected, then this information is lost for the top user, because the raw text of the skill section isn't present within the output, hence the user can't seek for skills manually. Since (on a subjective level) the loss of data is worse than trying to find skills manually, the subsequent methodology was used for measuring the performance of the skill parser:

* True positive for every skill that was correctly identified. The parser that correctly copied the complete text of the first skill section achieved a 100% precision, as its text contains all the talents of the first resume.

* False positive for every skill that wasn't listed originally within the skill section of the resume.

* False negative for every skill that ought to are identified within the skill section, but wasn't found by the parser.

1.3 Measurement evaluation

1.3.1 Performance check on the parsers

Brendan Herger's parser only extracts the names and occurrences of the programming languages, that were previously stored in every dictionary, but it doesn't explore the applicant's work experience. Thus this parser was found to not be relevant for the measurement.

The parser gives a trial to extract the applicant's job titles with success, because the output was often confusing. Results like "Faculty", "Analyst", "Engineer", "Microsoft", "Member" for an applicant's job titles weren't uncommon. The corresponding start- and end-date likewise because the description weren't extracted by the parser. The skill extraction capability of Skcript is additionally not satisfying, because the extracted skill are taken from the complete resume (instead of specializing in the skill section), and also the output contains an excessive amount of irrelevant information. Lists of 100 skills weren't uncommon, meanwhile they contained words (classified as skills) like "Sun City", "Team", "Teams", "Thermodynamics", "Tongue", "Understanding", "V5", "Writing" and "Yahoo". because of these flaws, Skcript's parser was also excluded from the measurement.

Lastly Antony Deepak's resume parser was examined. This parser extracts skills taken from the skill section of the resume and it also extracts the job's description, start- and end-date. As Deepak's parser fulfilled the above listed criteria, hence it had been measured for accuracy and recall.

1.3.2 Precision and Recall

As described by Powers (Powers, 2011), precision is the fraction of retrieved items that are relevant to the search, while recall is the fraction of relevant items that were revealed by the search. The two terms are visualized on Figure 10.

1.3.3 Measurement results

The measurement results conducted on Antony Deepak's parser are summarized on Table 4. This means, that when the parser has extracted a date or skill, then the output was correct ca. 85% of the time. However the parser has a poor recall of these fields (ranging between 25-66%), meaning that on average the parser only finds the skills, start- and end-dates with 66.66%, 40.27%, 25.37% respectively.

As seen on Table 4, dates have a significantly poorer recall than work description or skills. This has numerous reasons.

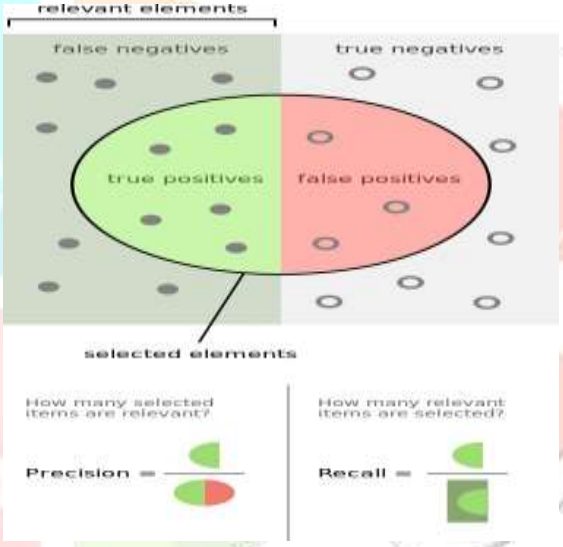

Figure 09: Precision and recall visualized (Walber, 2017)

1.3.3.1 Firstly the parser often made mistakes while separating individual work experi- ences from each other, and the output was a "merge" of two work experiences. While doing so, the second job's details (dates and description) were added to the description of the first job. However this argument only says, that the recall of dates should be just as bad as of the work description, but it doesn't say why these are even worse.

1.3.3.2 Secondly, the way people write dates often differ. The date "October 2017" can be written (among many ways) as "Oct. 2017", "Oct/2017", "10/2017", "10. 2017", "Oct. 17". If the parser is only capable of recognizing one or two of the representations, then the information written in other date formats are lost. This argument explains why the recall of dates are worse, but doesn't explain why end-date's recall is even lower.

1.3.3.3 Thirdly end-dates are often open ended, if the applicant currently works at an organization. Then people usually write "2017. Oct. - now", "2017. Oct. - present", or they just leave the closing part of the end-date empty: "2017. Oct. - ". Not treating all these cases results in a lower recall on end dates, as the parser is not able to find the end date at all. This argument finally explains why end-date performs significantly worse, than any other fields in the resume.

Table 1: Performance measurement on Antony Deepak's resume parser

	Start-date	End-date	Work Description	Skills
Precision (%)	85.29	85	66.12	80
Recall (%)	40.27	25.37	52.56	66.66

1.3.4 Conclusion

In conclusion it can be stated that the extraction of dates are probably the hardest task while parsing resumes, as there are so many different ways to express the same date. Moreover Deepak's parser often lost the original structure of the text, and hence it was not able to clearly distinguish between work experiences. When work experiences were merged together, not only information was lost, but the extracted information was also flawed, resulting in a high penalty while measuring precision and recall.

1.4 Commercial Resume Parsers

1.4.1 Measuring the perormance of commercial parsers

On the website of Rapidparser[1] one can freely upload resumes and view the result online. Using this feature, a similar measurement was conducted, using the same input data. The results are shown on Table 2.

Table 2: Performance measurement on Rapidparser's resume parser

	Start-date	End-date	Work Description	Skills
Precision (%)	95.77	1	97.68	100
Recall (%)	93.15	94.44	93.67	89

Except for the recall of skills, all the values are above 93%, which could be a significant improvement compared to the results of Deepak's resume parser. Rapidparser was also given a resume written in German, however it had been powerless to process it and after a while returned the "unsupported language (de)" error. Finally Rapidparser not only extracts skills, description, start- and end-date of labor experiences, but it also automatically recognizes names of programming languages and skills, that were used during a given work experience and outputs these in an "additional information" field for every work experience.

Upon the request Hireability and Daxtra also provided an online page, where their product might be tested. Hireability only allowed to parse 20 resumes. The performance of their resume parser is summarized on Table 3.

Table 3: Performance measurement on Hireability's resume parser

	Start-date	End-date	Work Description	Skills
Precision (%)	92.5	82.05	89.13	100%
Recall (%)	66.07	60.37	63.07	59.29

Unlike Rapidparser, Hireability's parser didn't take the whole skill section from the resume and inserted into the computer file, but it ran a "skill recognizer" on the whole resume, where only the previously known skills were matched and copied to the output. the present evaluation setup doesn't favor such an answer as less known frameworks, that weren't present within the default dictionary of the parser weren't found within the output. To demonstrate this with an example, assume that the HR is trying to find developers, who have experience in new technologies' trending frameworks (e.g.: machine learning frameworks like "Caffee" or "TensorFlow"). The parser needs to start the parsing process from the start (namely taking PDF files and converting these to outputs), rather than taking the prevailing outputs (e.g.: JSON files), that already contain the extracted skill section's text, and match the new framework names only during this part, thus avoiding the tedious tasks of knowledge extraction like PDF to text conversion, text cleaning, recognition of sections etc.

Daxtra's parser followed the identical skill extraction process as Hireability's, however Daxtra didn't limit the quantity of resumes within the test phase, thus the complete test set (25 resumes) can be evaluated. The results are summarized on Table 7.

Daxtra's and Hireability's parsers achieved 100% precision while finding skills. This was (probably) because their dictionaries only contained programming languages and frameworks, so no additional noise was added to the output.

Table 4: Performance measurement on Daxtra's resume parser

	Start-date	End-date	Work Description	Skills
Precision (%)	96.49	1	92.85	100%
Recall (%)	73.33	74.32	66.67	65.71

1.4.2 Conclusion

Commercial parsers have more sophisticated algorithms for date recognition than open-source parsers, which allows them to correctly identify these values with a higher recall. Improving date recognition is a matter of developing complex Regexps that match unique date time formats, that haven't been matched before.
However the strength of commercial parsers' probably lies in a structural analysis of the resume, that allows them to identify sections (skill section, work experience section), as well as individual work experiences with a higher accuracy, leading to a significantly higher precision and recall in the evaluation.

2. Final Conclusion

In this paper, we have discussed many open source and commercial resume parsers and their precision and recall percentage. We have done an analysis on the open source resume parsers and accuracy percentage are found very low. This impacts the extracted knowledge i.e it cannot extract information fully from the cv or resume. On the other hand commercial resume parsers are paid and most of them are customized for the organization. Commercial parsers have greater precision and recall value but are also not the 100% accurate. This parsers also not provide all the required information from the CV.

Here we got an idea for development and analysis of an algorithm for Knowledge Extraction for Automated Recruitment Process Using Natural Language Processing.
In the proposed solution, knowledge facts are extracted from resumes with different text formats and file types in our algorithm. The algorithm consists of two processing step, which are text block identification and name entity recognition. This work aims to improve the accuracy of extracting information from personal resumes. It is useful to build resume repository for head-hunters and companies focus on Internet-based recruiting. In the second processing step, we propose a Writing Style to distinguish different lines. Compared to those extracting algorithms, based on either HMM or CRF, our approach does not need too much manually annotated training set, which can save lots of human efforts and time. Meanwhile, experimental results on real-world dataset indicate that the precision and recall of free text resume extracting are better than them.

3. References

Eysenck, M. W. (2006). *Fundamentals of cognition*. Psychology Press.

Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

Gil, R. (2015). Introduction to web scraping. Retrieved January 5, 2018, from http: //ruipgil.com/2015/12/17/introduction-to-web-scraping.html

Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. *J. Artif. Intell. Res.(JAIR)*, *57*, 345–420.

Gorkovenko, M. (2017). Scraping a JS-Rendered Page. Retrieved January 5, 2018, from http:// stanford. edu/~mgorkove/ cgi- bin/ rpython_ tutorials/ Scraping_ a_ Webpage_Rendered_by_Javascript_Using_Python.php

Hardeniya, N. (2015). *NLTK essentials*. Packt Publishing Ltd.

He, Y. & Kayaalp, M. (2006). A Comparison of 13 Tokenizers on MEDLINE. *Bethesda, MD: The Lister Hill National Center for Biomedical Communications*, 48.

Hop, A. (2016). How to Scrape Javascript Rendered Websites with Python &Selenium. Retrieved January 5, 2018, from https://medium.com/@hoppy/how-to-test-or-scrape-javascript-rendered-websites-with-python-selenium-a-beginner-step-by- c137892216aa

Jones, K. S. (1994). Natural language processing: a historical review. In *Current issues in computational linguistics: in honour of Don Walker* (pp. 3–16). Springer.

Jurish, B. & Würzner, K.-M. (2013). Word and Sentence Tokenization with Hidden Markov Models. *JLCL*, *28*(2), 61–83.

Kaur, A. & Chopra, D. (2016). Comparison of text mining tools. In *Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), 2016 5th International Conference on* (pp. 186–192). IEEE.

Kibble, R. (2013). Introduction to natural language processing.

Kiss, T. & Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, *32*(4), 485–525.

López, F. & Romero, V. (2014). *Mastering Python Regular Expressions*. Packt Publishing Ltd.

Manning, C. D., Schütze, H. et al. (1999). *Foundations of statistical natural language processing*. MIT Press.

Màrquez, L., Padro, L., & Rodriguez, H. (2000). A machine learning approach to POS tagging. *Machine Learning*, *39*(1), 59–91.

Meystre, S. M. & Haug, P. J. (2005). Comparing natural language processing tools to extract medical problems from narrative text. In *AMIA annual symposium proceedings* (Vol. 2005, p. 525). American Medical Informatics Association.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nunley, J. M., Pugh, A., Romero, N., & Seals, R. A. (2016). College major, internship experience, and employment opportunities: Estimates from a résumé audit. *Labour Economics*, *38*, 37–46.

Palmer, D. D. (1995). SATZ-an adaptive sentence segmentation system. *arXiv preprint cmp-lg/9503019*.

Perkins, J. (2014). *Python 3 Text Processing with NLTK 3 Cookbook*. Packt Publishing Ltd.

Pinto, A., Gonçalo Oliveira, H., & Oliveira Alves, A. (2016). Comparing the performance of different NLP toolkits in formal and social media text. In *OASIcs-OpenAccess Series in Informatics* (Vol. 51). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, in- formedness, markedness and correlation.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Rey, G. & Wender, K. (2008). *Neuronale Netze: eine Einführung in die Grundlagen, Anwen- dungen und Datenauswertung*. Aus dem Programm Huber: Psychologie-Lehrbuch.

H. Huber.

Reynar, J. C. & Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the fifth conference on Applied natural language processing* (pp. 16–19). Association for Computational Linguistics.

Riley, M. D. (1989). Some applications of tree-based modelling to speech and lan- guage. In *Proceedings of the workshop on Speech and Natural Language* (pp. 339–352). Association for Computational Linguistics.

Rosier, A., Burgun, A., & Mabo, P. (2008). Using regular expressions to extract infor- mation on pacemaker implantation procedures from clinical reports. In *AMIA Annual Symposium Proceedings* (Vol. 2008, p. 81). American Medical Informatics Association.

Sonar, S. & Bankar, B. (2012). Resume Parsing with Named Entity Clustering Algorithm.

Steiner, B. (2017). Vector Representations of Words. Retrieved September 25, 2017, from https://www.tensorflow.org/tutorials/word2vec

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, *13*(2), 260–269.

Walber. (2017). Precision and recall. Retrieved October 17, 2017, from https:// en. wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg