Optimizing Text Encoding Model to Improve Matching Between Resumes and Job Postings

Andre Lustosa Motta
North Carolina State University
alustos@ncsu.edu

***Abstract*—Resume Job Matching is a classification task designed to be used by Application Tracking Systems. This task is supposed to prune the high number of resumes a job posting usually receives to ease the load on humans that have to go through those resumes. Most of those softwares only look to find specific keywords. And the problem of using text mining to rank resumes is considered a hard problem. One of the first steps of text mining is the encoding of the information. Or how we are going to represent the data. The vector based model of representation is widely used, but even though it has hyperparameters, those are rarely optimized to achieve a good representation to a specific domain. In this paper we propose the optimization of vector based encoding and demonstrate how by doing this we can improve classification.**

***Index Terms*—Resume/Job Matching, Application Tracking System, LDA, TF-IDF, classification techniques, experimental design, parameter optimization**

I. Introduction

The rise on the number of people graduating in Computer Science in the recent years and the fact that industry is more accepting of hiring people without college degrees has been causing a surplus on the market of Software Engineers. This surplus of available resumes has been met with the rise of Application Tracking Systems, which are automated systems that will analyze resumes before any human. These systems usually analyse the resume for keywords related to the job position and prune the number of resumes the HR department will have to read through to select the candidate.

However the literature on the problem shows that making predictions based on the extraction of information on resumes is a hard problem that usually leads to low accuracy results. And during the literature review on the problem we found no attempts at optimizing the encoding of the documents before passing it on to a classifier.

Given this we propose a tuned encoding model to generate better encodings relevant to the domain of the documents in order to generate better results. The proposed encoding model is the combination of TF-IDF [1] and an Latent Dirichlet Allocation(LDA) [2] implementation. The vectors generated from TF-IDF are fed into LDA and in order to rank the resumes to a certain job posting we would calculate the Cosine Distance between the target job and the resumes and rank the resumes according to that distance.

We used a database of 400 jobs and 400 resumes for Software Engineering positions. With those we were able to gather 388 matches of different combinations of jobs and resumes. These 388 matches were then used to test and evaluate the model.

II. Research Questions

Even though prior work suggests that resume classification is a hard test we have rarely seen attempts at optimizing the classifiers and no attempts at optimizing the encoding of the documents for their specific domain.

RQ1 How much does the performance of resume classification improve when automated parameter optimization of the encoding is applied?

When encoding a document, we are translating text to another dimensional representation. Our hope by using these encoding methods is that we keep as much semantic and important information on the new representation. However most of the encoding done will use default parameters of the chosen technique.

The question we are proposing then is whether using different hyperparameters can affect the encoding and whether we can optimize those to generate better representations for the domain of Resumes and Job Postings.

III. Case Study Approach

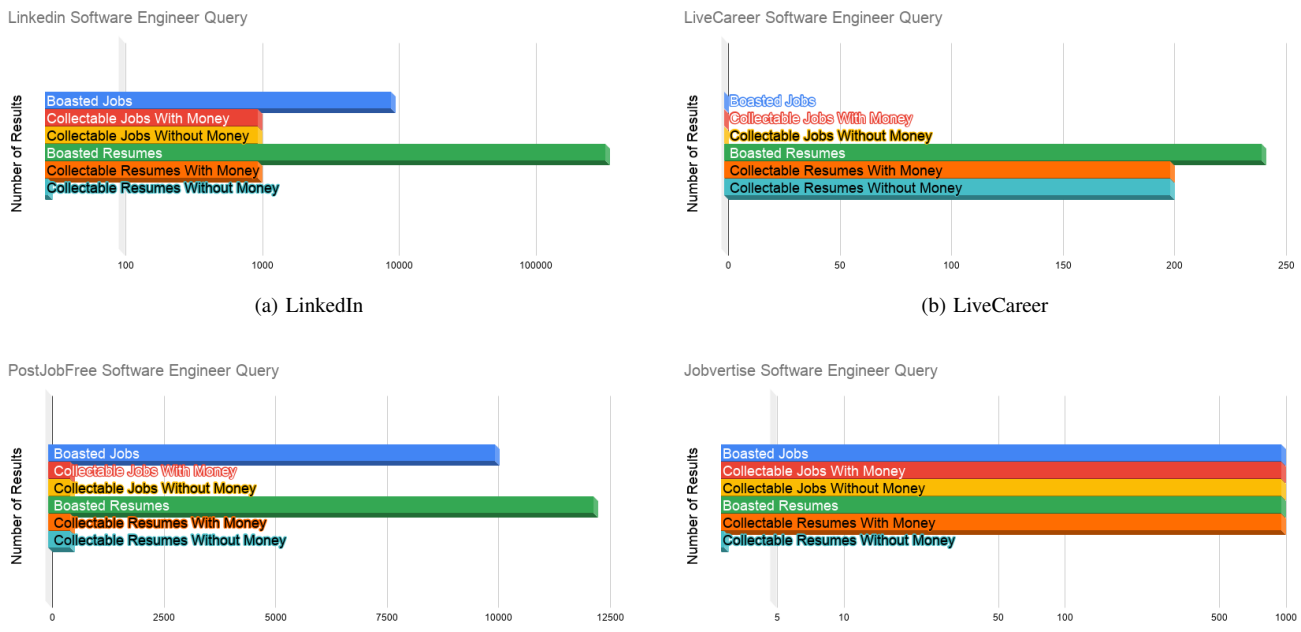
A. Data Collection

The challenge of data collection is to collect large scale unlabeled data of job posts and resumes.

1) What has been done: After carefully examined four online data sources (LinkedIn, LiveCareer, PostJobFree, and Jobvertise), we concluded that there is no free online data source that could provide large scale data for our project. The pattern we observed was that, usually those websites claimed a large number (millions) of resumes/job posts available, but in fact, only a few hundreds could be accessed by the users. Figure 1 shows the number of job posts/resumes can be collected from each data source. During Phase 1, we have collected 500 resumes and 500 job posts for the job title of software engineer.

B. Data Cleaning

The most important step in data cleaning is to remove the personally identifiable information (PII) in collected resumes. PII such as name and contact information should not be used or even accessible by the model for the subsequent training and analysis. Based on our analysis, the collected resumes from most data sources are not structured. Therefore it remains



(a) LinkedIn

(b) LiveCareer

(c) PostJobFree

(d) Jobvertise

Fig. 1: How much data can be collected.

a research challenge to remove PII from such unstructured resumes.

C. Data Labeling

Not only do we need large scale unlabeled data, but we also need ground truth labels for the required tasks. Without such ground truth labels, it is impossible to train the model as well as evaluating the performance of the model. Therefore the ground truth labels have to be collected for Job-resume matching: the ground truth of which resumes are qualified for which job posts.

To do so we generated 22500 triplets of possible matchings between resumes and a job post. A secondary resume was randomly selected to accompain the first one. This data was then passed to humans to classify which of the two resumes would be better for that job posting. This was achieved with the development of a cloud based server in .NET Framework and a React.js website which would allow for a scalable collection of the data.

1) The Triplet Test Rig: The server was developed in order to be reused. It supports any kind of triplet matching with textual data and can be scaled to any number of participants. The website is a simple webpage that shows the triplet. It can also be used to validate machine generated matches. Since it randomizes the order in which the options to be selected will appear. Giving no way to bias to be passed to the participant.

Unfortunately running such experiments proven to be hard and much of the collected data had to be cleaned. At the end we were able to collect 388 human made matches.

At the end our dataset for evaluation consisted of 388 pairing of resumes and job postings. These had a rate of agreement of over 90% and at least two human votes for the specific triplet selection.

D. Feature Engineering

This step focuses on finding the best representation of resumes and job posts to better serve the classification. In order to do this an extensive Literature Review on the topic of text encoding has been done.

After an extensive literature review on the topic of document encoding, we were able to assimilate four basic techniques to encode a document.

- **Vector Based Encoding**: Which are variations of techniques on frequency based generation of vectors that encode words, sentences, or entire documents. Some well known techniques that use Vector Based Encoding are the previously cited Term Frequency[1], LDA[2] and doc2vec[3]. Another technique for generating deeper vectors is the word vectors algorithm[4] (or word2vec), that similarly to doc2vec learns vector representations to a document at the level of each word. Which means each word will have its own vector at the end. One last interesting technique that we found interesting are the application of different deep neural networks to generate the document or word vectors[5]. By applying deep neural networks the intuition is to achieve better results in features on the vector space to better determine values of those features.
- **Table Based Encoding**: Which are another way of using frequency based techniques to encode documents into

tables[6] that can be compared to search for similarities and classification. These tables contain the frequency of each term in the document. And similarities may be computed by comparing the frequencies of the intersection set.
- **Graph Based Encoding**: Using a mixture of frequency based techniques and approaches of maximum contextual distance, encodes documents into a graph of relationships between words[7] [8]. The graphs can then be compared by how important words are in each of them (The degree of the vector multiplied by its weights being the importance factor).
- **Encoding by Self Organizing Maps (SOM)**: after transforming each word into a one-hot representation this approach generates an artificial neural network that represents the hierarchy of words in the document [9]. These hierarchies can then be used as an input to another classification tool to differentiate structures.

Usually those techniques are used together or combined with extra algorithms in order to perform different tasks, such as, Document Classification, Document Clustering, Document Summarizing, Document Similarity and Feature Extraction, to different degrees of success.

Out of all the papers reviewed on the topic of document encoding for different tasks, about 80% of them use Vector Based Encoding. The reason for that is that, out of the 4 techniques, Vector Based Encoding has the best trade off between complexity and performance. Graph Based Encoding and Self Organizing Maps, have a much higher computational overhead and while they may display better performance that might not be enough to make it worth the extra computational overhead. Only 2 papers out of the almost 100 studied used Table Based Encoding, therefore we don't have many results to compare the performance and complexity trade off between them.

With the review we were able to find some very interesting combinations using the following techniques:
- KNN (K-Nearest Neighbours)[10]
  Simple non-parametric classifier that uses the concept of distance to classify new data.
- SVM (Support Vector Machine)[11]
  Simple parametric linear classifier.
- NN (Multi Layered Perceptrons)[12]
  Combination of perceptrons into a parametric classifier.
- NB (Naive Bayes)
  Simple statistical analysis of the likelihood of the class given the evidence.
- CNN (Convolutional Neural Network)[13]
  Combinations of perceptrons that apply filtering convolutions into a deep neural network model for a parametric classifier
- GA (Genetic Algorithms)[14]
  Similar to optimization techniques approximates a desired function by mimicking natural evolutionary processes
- RNN (Recurrent Neural Network)[15]
  Combination of perceptrons into a parametric classifier with a complex architecture capable of retaining memory for deep neural processes.
- CL (Clustering Algorithms)
  Unsupervised algorithms for clustering together similar points in the database
- SK (Semantic Kernels)[16]
  Technique to identify the defining key words in a document

E. Model Definition

After this extensive literature review, and with our idea of testing solely the optimization of encoding techniques, we decided on the following model:

1) Encoding: Two different techniques have been combined to encode resumes and job posts. They are:
- **Term Frequency (TF-IDF)**: each document is encoded as a fixed length vector representing the frequency of each term appears. We kept 4,000 terms with highest tf-idf metric [1]. This has been the most commonly used and basic text embedding technique for tasks like text classification [1].
- **Latent Dirichlet Allocation (LDA)**: LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities [2]. LDA is widely used for topic modeling of text data and can be used as an representation of text.

2) Classification: To classify the resumes accordingly to a job posting we used a distance based technique. The Cosine Distance, which is defined as:

$$cos(Job, Resume) = \frac{Job \cdot Resume}{||Job|| \cdot ||Resume||} \tag{1}$$

We are then simply calculating the distance between the final encodings of the data. Without any further learning task in order to separate any improvement effect that a learner can cause on the comparison of optimized models.

3) Architecture: The architecture of the model as be seen in Figure 2 consists of three separate databases. The database containing the Resumes, the database containing the Job Postings and the database containing the Human-Made Triplets. The first two databases are piped through the Term Frequency encoder which generates a sparse matrix, and this sparse matrix is passed to the LDA model which returns the encoded texts. From there on we pass to the evaluation step. Where the model will receive a human made Triplet and compare the distance between resumes and jobs. The results are recorded on a file.

In this a job post and a resume match better if the distance between their representations is shorter. This is an unsupervised learning model which does not need ground truth labels to train, but only relies on the text representation technique. It can be used as a baseline model and can be used to evaluate the performance of different text representation techniques in the future.

TABLE I: Explored Combinations in Literature

Encoding Techniques	Learners Used									
	KNN	SVM	NN	NB	SVD	CNN	GA	RNN	CL	SK
Vector Based		[17]			[18]	[19] [20] [21]	[22]	[21] [23]	[17] [22] [24]	[16]
Table Based	[6]	[6]	[6]							
Graph Based		[7]							[8]	
Self Organizing Maps	[25]						[26]		[25]	

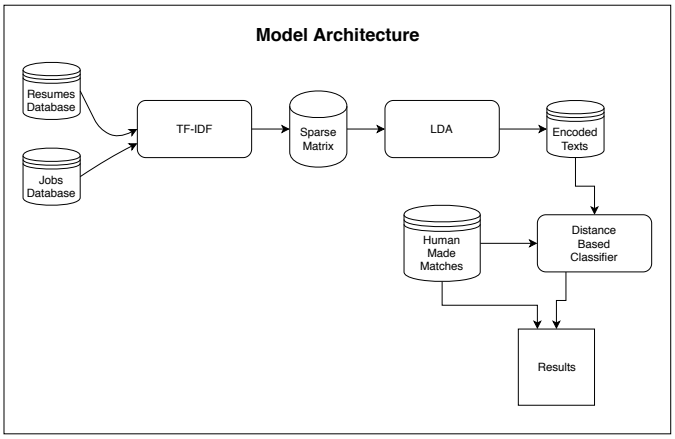


Fig. 2: Model Architecture

F. Optimizing the Model Parameters

The two steps of encoding into the model both have parameters which might be optimized for a better generation of a representation for each document.
TF-IDF Parameters:

- Norm: Normalization strategy for vectors
- Useidf: Whether to consider the inverse document frequency or not
- Max_features : Maximum number of features in vocabulary
- Sublinear_tf: replace tf with 1+log(tf)
- Decode_error (What to do on errors in decoding bytes from text)

LDA Parameters:

- N_topics: The number of topics to find
- N_iter: How many iterations of the algorithm to run
- Alpha: related to how many dominant topics a document has
- Beta: related to how many dominant words a topic has

For this work in progress paper 51 models were generated by generating one million random combinations of the 9 parameters above mentioned. After that we ran K-means with K = 50 on random weights assigned to each of the one million combinations. Through the centroids we determined the parameters for each of the models by selecting them on a distribution of the options on the random weights of the centroids. The 51st model was the selected baseline with the default parameters found in the python packages

IV. Testing the Models

The models were trained on all the job postings and resumes. Trained to be able to encode all 1000 documents. This was done by passing the documents to TF-IDF and then into LDA.
In order to evaluate the models we made the models rank the two resumes in the human made match against the job posting by using the cosine distance between the encodings. If the models were able to classify the human decided resume as best we counted as a successfull prediction.
The evaluation function was then just a measure of accuracy

$$Accuracy = TruePositives/Total \tag{2}$$

Initially we were able to see that some optimized models did outperform the baseline. However in order to trully understand wether that was a real difference in performance we ran each of the 51 models 20 times with the same dataset, from there we used the recorded accuracy for each run executed a Scott Knott's test.

With the Scott Knott test done we were able to confirm that some optimized models outperformed the baseline by about 10% in accuracy. Not only that but the baseline model was on the lower end of Scott Knott suggesting that the base parameters suggested by the package were not good enough for the task. The base parameters had an accuracy very close to a coin flip, suggesting that it did not learn much and was predicting randomly

V. Case Study Results

After the aforementioned analysis it was possible to get the following answer to the research question:

RQ1 As seen in Figure 3 we can safely assume that optimizing the encoding for the specific domain of Resumes/Job Postings leads to a positive impact in the accuracy of classification tasks even when coupled with a simple distance based classifier.

VI. Threats to Validity

- **Construct Validity:** The results from RQ1 show that optimizing the encoding improves the performance of matching resumes to job postings. However, the performance improvement may increase the complexity of the models, the increase on certain parameters both in TF-IDF and LDA can substantially increase the training time.
- **Internal Validity:** The performance of the classifiers was measured using accuracy. Other evaluation frameworks based on different approaches may have different results.

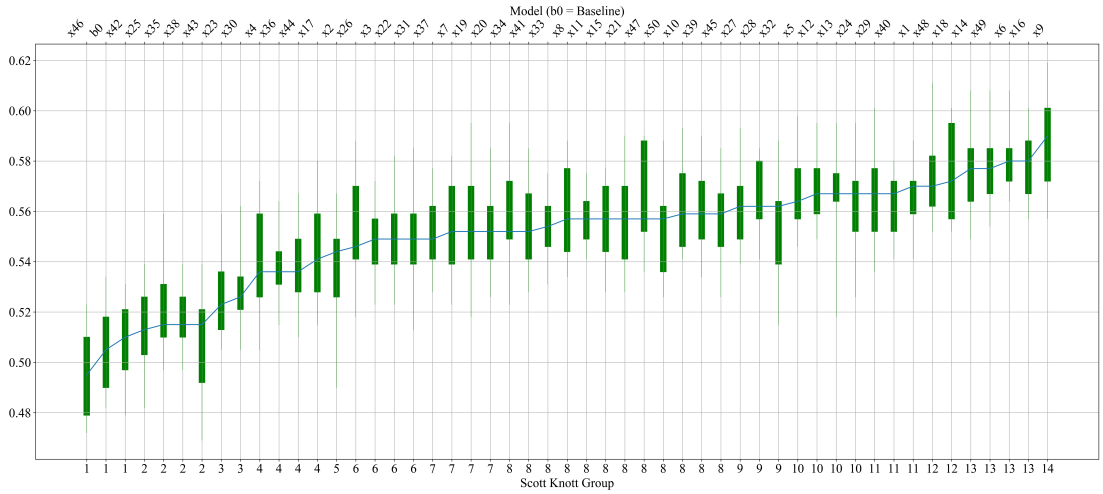


Fig. 3: Scott Knott Test on the Models Performance

- **External Validity:** The size of the dataset studied in this paper was limited. Therefore the results may not generalize to every job category. However the point of this paper is not to generalize but to show that optimization can be a good tool to improve the accuracy of the matchings.

VII. Future Work

Throughout the paper we only used a limited number of tools because of the labeling problem. We hope to, in the future, collect more human data and try to tackle the problem using different classifiers and other supervised techniques. With this we hope to understand wether the impact caused by optimizing the encoding will reflect on learners coupled with the suggested model.

References

[1] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[3] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *arXiv preprint arXiv:1507.07998*, 2015.

[4] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[5] V. Christlein and A. Maier, "Encoding cnn activations for writer recognition," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, April 2018, pp. 169–174.

[6] T. Jo and G. Jo, "Table based single pass algorithm for clustering electronic documents in 20newsgroups," in *2008 IEEE International Workshop on Semantic Computing and Applications*, July 2008, pp. 66–71.

[7] F. D. Malliaros and K. Skianis, "Graph-based term weighting for text categorization," in *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Aug 2015, pp. 1473–1479.

[8] B. F. Momin, P. J. Kulkarni, and A. Chaudhari, "Web document clustering using document index graph," pp. 32–37, Dec 2006.

[9] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, Jan 1982. [Online]. Available: https://doi.org/10.1007/BF00337288

[10] E. Fix and J. Hodges, "Discriminatory analysis, nonparametric discrimination: Consistency properties," *Technical Report 4, USAF School of Aviation Medicine, Randolph Field.*, 1951.

[11] L. A. Vapnik V, "Pattern recognition using generalized portrait method. automation and remote control," vol. 24, pp. 774–780, 1963.

[12] S. Kleene, "Representation of events in nerve nets and finite automata. automata studies." vol. 34, pp. 3–42, 1956.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[14] A. J. O. L. J. Fogel and M. J. Walsh, "Artificial intelligence through simulated evolution," 1966.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: http://www.nature.com/articles/323533a0

[16] S. Bloehdorn, R. Basili, M. Cammisa, and A. Moschitti, "Semantic kernels for text classification based on topological measures of feature similarity," in *Sixth International Conference on Data Mining (ICDM'06)*, Dec 2006, pp. 808–812.

[17] W. Song and S. C. Park, "A novel document clustering model based on latent semantic analysis," in *Third International Conference on Semantics, Knowledge and Grid (SKG 2007)*, Oct 2007, pp. 539–542.

[18] R. R. K. Menon and N. Aswathy, "Document summarization using dictionary learning," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2017, pp. 645–650.

[19] J. D. Prusa and T. Khoshgoftaar, "Training convolutional networks on truncated text," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, Nov 2017, pp. 330–335.

[20] I. Gallo, S. Nawaz, and A. Calefati, "Semantic text encoding for text classification using convolutional neural networks," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 05, Nov 2017, pp. 16–21.

[21] P. Chen, W. Guo, L. Dai, and Z. Ling, "Pseudo-supervised approach for text clustering based on consensus analysis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 6184–6188.

[22] E. Len, J. Gmez, and O. Nasraoui, "A genetic niching algorithm with self-adaptating operator rates for document clustering," in *2012 Eighth Latin American Web Congress*, Oct 2012, pp. 79–86.

[23] P. Yan, L. Li, and D. Zeng, "A shortcut-stacked document encoder for

extractive text summarization," in *2019 International Joint Conference on Neural Networks (IJCNN)*, July 2019, pp. 1–8.

[24] B. Saha, D. Phung, D. S. Pham, and S. Venkatesh, "Sparse subspace representation for spectral document clustering," in *2012 IEEE 12th International Conference on Data Mining*, Dec 2012, pp. 1092–1097.

[25] D. Zufferey, S. Bromuri, and M. Schumacher, "Case-based retrieval of similar diabetic patients," in *2015 9th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, May 2015, pp. 312–316.

[26] X. Luo and N. Zincir-Heywood, "Incorporating temporal information for document classification," in *2007 IEEE 23rd International Conference on Data Engineering Workshop*, April 2007, pp. 780–789.