A Comparative Analysis of Resume Screeners' Word Vectorization Using Cosine Similarity

Isaac B. Wan and Jack W. Keller

Department of Computer Science, Macalester College

COMP 484: Introduction to Artificial Intelligence

Professor Susan Fox

December 19th, 2023

Abstract

Due to the large influx of applicants received for each new online job posting, companies have widely adopted resume screeners that use artificial intelligence (AI) natural language processing techniques to quickly and efficiently evaluate potential candidates. Given this knowledge, the primary research conducted in this study is to learn how well these AI resume screeners perform on a given set of applicants' resumes to a job description and whether or not these resume screeners can discern between relevant and irrelevant candidates. This study compares four different Python resume screening implementations which all compute the cosine similarity from word vectors created for each resume to output an accuracy score for an AI-generated entry-level software developer job description. The researchers tested four Python libraries as resume screeners: scikit-learn, Numpy, Word2Vec, and a pre-trained language model from Google called BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers). To obtain a baseline, the researchers performed a hand-graded resume evaluation on the ten obtained resumes (seven student resumes, including their own, two ChatGPT-generated resumes, and one resume from the Kaggle Resume dataset in the Information Technology field) and then tested each candidate's resume on the four aforementioned Python libraries to make a ranking comparison. Additionally, the researchers tested the scikit-learn, Numpy, and BERT resume screeners with a Kaggle resume dataset using its resumes from the Information & Technology, Banking, and Chef career directories and determining their cosine similarity to the entry-level software developer job description. The results showed that simpler Python resume screeners (i.e. scikit-learn and Numpy) were able to clearly distinguish between relevant and irrelevant resumes, while the more complex Python resume screeners (i.e. BERT and Word2Vec) were not.

Keywords: Cosine Similarity, Word Vectorization, Resume Screening, NLP

1. Introduction

A common theme amongst undergraduate students studying in STEM is that they face heartbreaking rejection emails for jobs they applied for online without ever receiving a job interview. The harsh reality for these and many other undergraduate students is AI has drastically altered the hiring process for job applicants. Nowadays, artificial intelligence has become the primary method used by companies to screen resumes in the first stage of the hiring process. Due to the recent advancements in natural language processing (NLP), several companies have cut back on hiring dedicated human resource employees to read, evaluate, and review hundreds to thousands of candidates' resumes in favor of a more time and cost-efficient AI resume screener (Carpenter, 2023). As senior computer science undergraduate students, we have already begun applying for full-time job positions and fellowships. By researching and understanding the inner workings of AI resume screeners, we could learn specific methods to help other students navigate this arduous automated hiring system. In our study, we are especially curious to learn more about various AI resume screeners, how they are implemented/function, and what their advantages and disadvantages are. We are also interested in learning about potential areas of bias in AI resume screeners, and why employers have not completely shifted to solely using them.

For the purposes of our research study, we will be implementing four Python AI resume screeners (sci-kit learn, Numpy, and BERT) which make use of cosine similarity from their created word vectors. The cosine similarity method is particularly useful in the case of evaluating resumes because often when candidates apply online for jobs, they cater their resume to include the specific keywords mentioned within the job description’s “Minimum Requirements” and “Preferred Skills” sections. In our study’s comparative analysis, we analyzed each resume screener’s cosine similarity test results to learn which implementations were most similar to our

hand-graded evaluations on the sample resume dataset. We are comparing the resumes' results with our own evaluation to determine whether or not the resume screeners are able to distinguish between resumes that are related and unrelated to the entry-level software developer job description. Additionally, we tested the scikit-learn, Numpy, and BERT resume screeners with a Kaggle resume dataset using its resumes from the "Information & Technology," "Banking," and "Chef" career directories and determined their cosine similarity to the entry-level software developer job description (Bhawal, 2021). After performing this comparative analysis, we concluded that not all Python libraries' cosine similarity and word vectorization are created in the same way which can cause discrepancies in calculating each resume's cosine similarity score. Overall, however, scikit-learn and Numpy's resume accuracy score ranked the resume dataset closest to our own evaluations and BERT was the furthest away in ranking the resume dataset to our grades.

2. Background

A key measurement in our resume screener implementation is the use of cosine similarity. Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi (2012) explain that:

> Cosine similarity is a widely implemented metric in information retrieval and related studies. This metric models a text document as a vector of terms. By this model, the similarity between two documents can be derived by calculating cosine value between two documents' term vectors …the higher similarity score between document's term vector and query's term vector means more relevancy between document and query. (p. 1)

The cosine similarity formula is as follows: $similarity(A, B) = cos(\theta) = \frac{A \bullet B}{||A||B||}$

In the numerator, the dot product of *A* and *B* are calculated, while in the denominator, the magnitude of the vectors A and B are calculated and then multiplied together (Rahutomo et al. Figure 3, 2012).

In terms of previous research that has been completed on creating word vectors, Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean discovered "two novel model architectures for computing continuous vector representations of words from very large data sets," which are the Continuous Bag-of-Words (CBOW) model and the Continuous Skip-gram model (Mikolov et al., 2013, pg. 1). The main differences between a Continuous Bag-of-Words and a Continuous Skip-gram model are how each model is set up in the neural network and their word input(s) and output(s). In the case of the Continuous Bag-of-Words model, the neural network is given multiple surrounding words for a chosen word in a sentence in order to predict what the chosen word is. For example, a potential sample input is "Good morning, ___ are you?" where the middle word "how" (single output) is exempt from the sentence and the neural network must use "morning," "are," "good," and "you" (multiple inputs) to determine the blank word "how." In the case of the Continuous Skip-gram model, the neural network is only given a single word and must predict its surrounding words. Using the previous example, if given the word "how," (single input) the model predicts "morning," "are," "good," and "you" (multiple outputs). Based on the researcher's results, they found that in general the Continuous Bag-of-Words model is much more time efficient in training than the Continuous Skip-gram model and does a better job of predicting commonly used words. However, the Continuous Skip-gram handles a more compact dataset and less commonly used words better than the CBOW model.

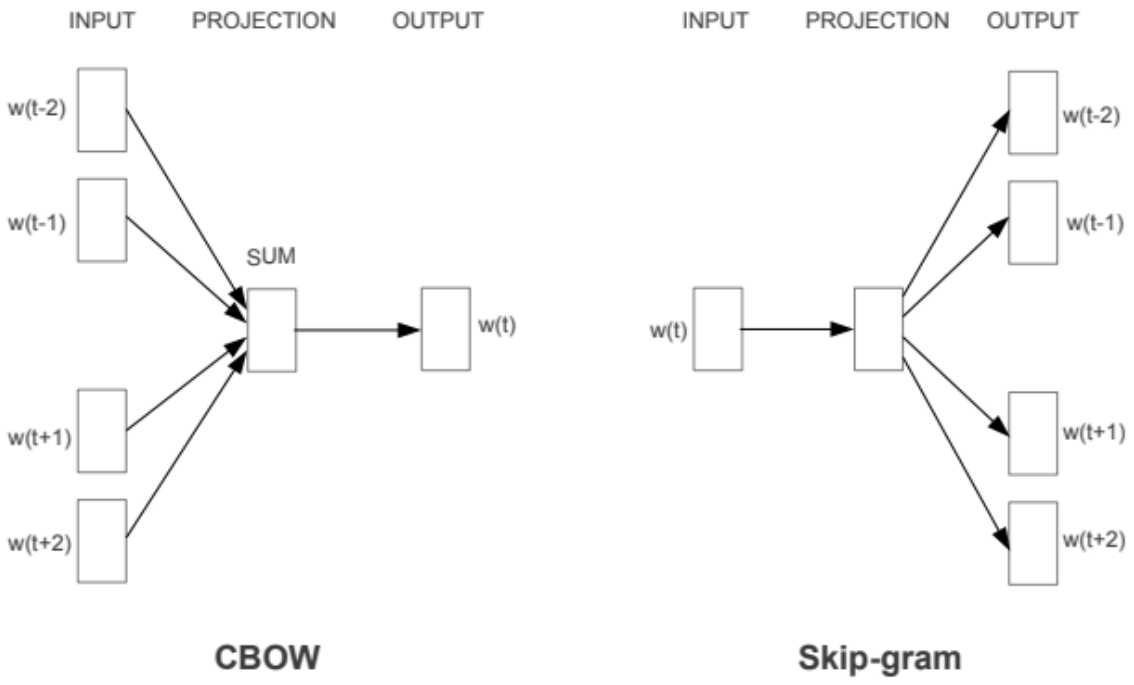

Figure 1 from Mikolov et al., 2013 shows the CBOW model and Skip-gram model neural networks

While we are not training any large language models in our study, it is important to understand that both word representation models are commonly used in many resume screeners whose pre-trained language models perform either the CBOW or Skip-gram training operation.

In our study, we tested the cosine similarity results of BERT, a pre-trained language model. BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) is a pre-trained language model from Google who's trained corpus consisted of data obtained from the English language of Wikipedia and BookCorpus. The combined word total of both equates to roughly 3,300 million words. BERT's unique unsupervised training methods, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), in tandem with its ability to read texts from both directions, help differentiate itself from the other large language models in its ability

to comprehend and link words in the text's context at a much deeper level than the vast majority of other large language models at the time (Devlin et al., 2018). We selected BERT for our study because of its unique language processing complexities compared to its competitors, scikit-learn and Numpy.

3. Related work

3.1 The Perceptions and Effects of Artificial Intelligence in the Job Recruitment Process

In "Perception of Artificial Intelligence AI-Based Tools in Recruitment" by Piotr Horodyski, he discusses the AI systems and how they have become increasingly commonplace in usage to support automating the hiring process of companies. However, the gap in research he is attempting to fill is the recruiter's perception of AI-based tools and the reasons for using AI in the recruiting process. This paper studies the perception of AI technology through the recruiter's lens by interviewing Human Resources professionals who have had experience with AI recruitment tools. This study employs the Unified Theory of Acceptance and Use of Technology (UTAUT) research model, which is made up of four important determinants as follows: performance expectancy, effort expectancy, social influence, and facilitating conditions. The UTAUT model showed that recruiters' behavioral intention was significantly and positively impacted by performance expectancy. Furthermore, effort expectancy, as well as efficiency gains were viewed as the most important benefits, while the main drawback was the AI's lack of nuance in human judgment seen by recruiters. Ultimately, this paper suggests that these findings can be useful for software developers and us to understand factors that increase or decrease the use of AI in recruitment.

In “Should I Trust the AI to Recruit?” by Alain Lacroux and Christelle Martin-Lacroux, their study’s purpose is to better understand recruiters’ reactions to resume screeners and why recruiters prefer human recommendations as opposed to algorithm-based recommendations. The main AI tool that they analyzed was called Algorithmic Decision Support Systems (ADSS) which can provide generated recommendations given a document. For the purposes of their research, they primarily focused on candidate recommendations generated by ADSS given their resumes and a job description. Their experiment involved manufacturing a job description for an HR manager position and giving participants two resumes where the applicants did not have the same level of qualifications. The participants then completed an attention survey to ensure they correctly identified the differences between the two resumes. Lastly, they provided their trust levels either falling into the category of trusting ADSS’ recommendation or trusting a human expert’s recommendation. The researchers discovered that participants tend to trust human expert recommendations rather than AI recommendations. The research paper is related to our study’s methodology because it informs us, as researchers, of our own potential innate biases towards trusting other humans. This applies when hand-grading human resumes in our dataset since it may be difficult to account for human trustworthiness bias.

“A Literature Review: Artificial Intelligence Impact on the Recruitment Process” by JihaD FraiJ and Várallyai László delivers a wide overview of how Artificial Intelligence has affected many aspects related to the human resource departments at companies. Generally, the authors found that other researchers agree AI is advantageous to the recruitment field and can reduce the overall time spent reviewing resumes and assist in prioritizing their time on performance and development. There are also alternative benefits besides time that come with automating such a monotonous task. For example, resumes that are either at the beginning or end

of the resume pile will likely be more memorable to a recruiter than resumes within the middle of the pile. The three main processes in the hiring process that the authors researched existing literature on are screening, human bias, and best-fit candidates. Screening is usually the first thing that comes to mind when researchers look into AI in HR. The primary concern for AI in this process is that it struggles to read emotional states through verbal and written communication, but the positives are that it saves companies plenty of time and money in return. In terms of human bias, AI can be used to allow candidates to have "blind auditions" for a given job, but if not calibrated for bias properly can lead to sub-par results. For candidates who are job hunting, AI can help by recommending jobs that best fit their skill set.

All three of these research papers focused on topics that surround the recruiter's perceptions and the effects artificial intelligence has had on the job recruitment process. Both FraiJ and Várallyai 's literature review and Horodyski's research demonstrate the large benefits AI has in the recruitment process by alleviating much of the stress hiring managers and company recruiters face while evaluating candidates. While Alain Lacroux and Christelle Martin-Lacroux's study on AI ADSS technology aims to examine the potential human bias involved in the hiring process, their research also reveals an important point about using AI as a resume screener as it ultimately removes any aspect of potential human bias towards candidates. Overall, the findings in these papers provided us, as researchers, with the recruiter's perspective on the use of AI in the job recruitment process.

3.2 Resume Screening Methods

Sunaina Arora and Neeraj Kumari's "Recruitment Search Engines for Screening Resumes through AI by Using Boolean Search Functions" explores the usage of artificial intelligence (A.I.) by recruiters for candidate resume screening and demonstrates searches on recruitment

database access tools to find qualified candidates. Their finding showed that Boolean operators (AND, NOT, OR, or AND NOT) can significantly lower the time and effort needed to gather necessary search results for recruiters and hiring managers by filtering out any irrelevant candidates in their job application database based on a job description. By examining services such as Linkedin and TimesJobs, this study examines the benefits of using Boolean operators and other filtration options to organize and sort qualified resumes from unqualified ones. This paper presents an example of an existing resume screening strategy (using Boolean operators) used to screen resumes.

"Automatic Software Engineering Position Resume Screening Using Natural Language Processing, Word Matching, Character Positioning, and Regex" by Dipendra Pant, Dhiraj Pokhrel, and Prakash Poudyal describes resume screening as an extremely tedious job, with possible mistakes due to human error or bias. They mention that automating this process with Natural Language Processing (NLP) resume screening, however, can reduce the effort, time, and cost required. This paper details the many straightforward methods for using NLP systems when screening structured resumes, such as the label character positioning-based technique. Additionally, using keyword and phrase matching is advantageous when screening unstructured resumes.

Lastly, in "Mitigating Demographic Bias in AI-Based Resume Filtering" by Ketki V. Deshpande, Shimei Pan, and James R. Foulds they attempted to mitigate demographic bias by using natural language processing and a de-biasing technique called "fair-tf-idf." Their research question stems from the existing issue that resumes are pieces of writing that contain socio-linguistic characteristics. These characteristics can inform the hiring manager of the job candidate's gender, race, and ethnicity which could cause possible bias and discrimination

against certain candidates. To remedy this, the researchers developed an A.I. resume-filtering system, which matches the relevant keywords in a candidate's resume to a job description's keywords and removes any irrelevant words. Their research results were that the Sigmoid Transformation method of "fair-tf-idf" performed best at this resume de-biasing technique because of its relatively low accuracy loss while still maintaining a high fairness rating across all resumes.

Ketki V. Deshpande, Shimei Pan, and James R. Foulds's research paper and Dipendra Pant, Dhiraj Pokhrel, and Prakash Poudyal's research papers directly contradict each other's perspectives on bias in AI resume screeners. While it is true that AI resume screeners can provide a means of reducing human bias and error, AI can also supply its own bias depending on its programming. Ultimately, Arora and Kumari's research on Boolean operators provides a reasonable middle-ground solution to effectively preventing both human and AI bias while screening resumes because Boolean operators can be used to filter resumes that only contain the relevant keywords in a job description, which removes any socio-linguistic characteristics and potential human error or bias.

4. Methodology

Our project runs out of four Python files, titled scklearn.py, BERT.py, numpyCosine.py and Word2Vec.py. Each file takes in a resume or directory of resumes and a job description, computing a cosine similarity score of how similar a given resume and job description are to each other. All of these methods use a different method of word vectorization and function for calculating cosine similarity. Both of these actions make use of the various Python libraries available such as scikit-learn, BERT transformer, Gensim, Numpy and PyTorch. With these varying libraries we aimed to test differing levels of complexity when it comes to resume

screening. For example, the scikit-learn and Numpy usage were meant to be as simple as possible, requiring no pre-training. BERT and Gensim's "glove vectors" word vectorization combined with PyTorch's cosine similarity on the other hand were both instances of more advanced pre-trained models that might produce more accurate results given their complexity. This array containing both simple and complex models was chosen with the goal of determining whether or not more advanced models could identify relevant resumes with higher accuracy than the simpler models.

The first Python model, sklearn.py, utilizes scikit-learn's countVectorizer tool to vectorize the text within the documents and the scikit-learn cosine_similarity(). Taking in a directory of resumes and a job description, this file computes the similarity between each resume and the job description, returning the highest graded resume. The countVectorizer method utilizes the bag of words method of word vectorization, and counts the instances, adding weights. Next, after being converted to a matrix, the cosine_similarity() function in scikit-learn's metrics pairwise library is used to compute similarity (Pedregosa et al., 2011). After each resume's score has been stored, the program will then return the cosine similarity score of the highest matching resume and the name of the resume. This model was meant to determine the accuracy of a more simplistic untrained vectorization strategy.

The second model, BERT.py, employs the pre-trained BERT transformer to vectorize document text and the PyTorch cosine similarity function. Similar to the first model, BERT.py takes in a directory of resumes and a job description, first extracting the word embeddings from the text with a pre-trained BERT tokenizer and encoder. Next, after obtaining these embeddings for the resumes and a job description, PyTorch cosine similarity computes the similarity between two tensor inputs (Paszke et al., 2019). After the similarity between each resume and job

description has been stored, the program will then return the cosine similarity score of the highest matching resume and the name of the resume. This model stands opposite to sklearn.py, attempting to analyze the accuracy of vectorization and word embeddings of a much more complex, trained model.

The third model, Word2Vec.py, can be described as a middle ground between the BERT and scikit-learn models. While the Word2Vec model implements pre-trained glove vectors from the Gensim library, the program synthesizes a more crude centroid vector to obtain a cosine similarity that strikes a balance between the simple and complex models. Since Gensim's Word2Vec models are only capable of word-to-word similarity, a centroid vector is derived as the average of all word vectors in the given resume and job description. Although not as complex as some other methods of converting word vectorization to document vectorization, this model acts as a stepping stone between the basic Sklearn.py and the sophisticated BERT.py.

The last model, numpyCosine.py is another simpler model. It derives feature vectors from the resumes and job description, before manually computing the cosine similarity with Numpy's dot product and vector magnitude tools. By finding all possible words between both documents, the Numpy model is then able to map the occurrence and prevalence of features in each document and compare the distance between vectors to obtain the cosine similarity. This direct, more rudimentary strategy of feature counting vectorization is similar to the Sklearn model and will provide further evidence to support the findings of these four separate models.

Each of these models provides a wide array of complexity and "intelligence" to examine upon testing. To examine these widely different models, two methods were taken to sufficiently analyze model performance. For the first hand-grading research, ten resumes were taken, made up of the following: five computer science student resumes of varying internship and job

experiences, one ChatGPT full-stack developer resume, one economics student's resume, one ChatGPT economics recent undergraduate resume, one geography student's resume, one highly testing information and technology resume pulled from a Kaggle dataset.

Each of these resumes was then tested with all four models against a ChatGPT entry-level software developer job listing for cosine similarity. With this variety in experience and field, the analysis would hopefully demonstrate the accuracy of each model. To determine that "accuracy," each resume was hand-graded and ranked in an order of highest to lowest. With the huge gap in mean score for each model, the ranking of each model's cosine similarity scores was compared to our hand-graded resume ranking to determine the success of a model.

Secondly, after recognizing the small dataset from the hand-grading research, it became necessary for a larger dataset of Kaggle resumes to be analyzed (Bhawal, 2021). This reflected a shift in focus from the specific ordering of hand-graded resumes to the model's ability to recognize relevant and irrelevant resumes. In this additional research, three separate resume directories of vastly different fields of decreasing job relevance were passed into the models: "Information & Technology," "Banking," and "Chef." The accuracy of each model could be better determined by counting the number of resumes in each directory that passed a certain threshold for each model. The specific thresholds for each model were: .18 for scikit-learn, .68 for Numpy, and .954 for BERT. With this second method of research, a more confident conclusion on the models' accuracy could be made.

5. Analysis

The following table illustrates the scores of resumes for each model from 0.0 - 1.0 similarity compared to the job description.

Resume File Names	Word2Vec	Numpy cosine	BERT	scikit-learn
ChatGPT Econ Resume	0.9540	0.540937	0.95540946	0.1374645
EllaDeMay_Resume	0.9458	0.477433	0.9309112	0.995270
Fullstack-Developer-Resume ChatGPT	0.9825	0.688766	0.973569	0.2591987
Jessica McAllister_Resume	0.9533	0.57216	0.935318	0.493617
KaggleIT_Resume	0.9533	0.680857	0.985602	0.2133144
ShomrikMondal_Resume	0.9749	0.6927965	0.9100272	0.3074170
AlbertLuna_Resume	0.9667	0.58647	0.909582	0.30146843
AsarTamurResume	0.9706	0.57550	0.904982	0.1783749
IsaacWan_Resume	0.9784	0.71300	0.89478272	0.2290347
JackKeller_Resume	0.9528	0.44962	0.9724664	0.18627700

Table 1 contains the cosine similarity scores for the ten resumes across the four implementations

However, the results from running tests of our ten resumes set against our models' results were not abundantly clear upfront. The results of the ranking order for each resume-job description's similarity against the four models can be seen below.

Resume File Names	Hand Grade	Word 2Vec	Numpy cosine	BERT	scikit-learn

AlbertLuna_Resume (CS)	2/4	5	5	8	2
AsarTamur_Resume (CS)	2/4	4	6	9	7
ChatGPT Econ Resume (Non-CS)	10	6	8	4	8
EllaDeMay_Resume (Non-CS)	8/9	10	9	6	9
ChatGPT FullStack Resume (CS)	1	1	3	2	3
IsaacWan_Resume (CS)	5	2	1	10	4
JackKeller_Resume (CS)	7	9	10	3	6
Jessica McAllister_Resume (Non-CS)	8/9	7/8	7	5	10
KaggleInformation&Technology_Resume (CS)	3	7/8	4	1	5
ShomrikMondal_Resume (CS)	4/6	3	2	7	1

Table 2 contains the numerical rankings (1 as the best and 10 as the worst) for the ten resumes across the Word2Vec, Numpy, scikit-learn, BERT implementations, and hand-graded evaluation

The hand-graded order of resumes sits on the left side of the table, with some resumes carrying two values denoting instances where the two hand-grading orders disagreed. By creating rankings for our ten resumes across all the resume screening methods, it becomes clear in this table that the resumes' rankings between models fluctuated greatly. Given the wide variety of ordering for these different models, the information in the table alone still does not demonstrate any clear trends. To better understand the accuracy of these models, the last step required examining each model's ranking in comparison to the top 6 hand-graded resumes. In doing so, the accuracy of each model became more apparent. Word2Vec - 5/6 (not Kaggle's), numpyCosine - 6/6 (Interestingly had a drop off in score after #4 from 68% - 58%), BERT - 2/6, Sklearn - 5/6 (not Tamur's). With such different rankings for each model and hand grading, this

method of examining the “top 6” resumes in a model reveals the lack of reliability in the most advanced model, BERT.py. On the other hand, the simplest model, numpyCosine, was able to match our hand-graded ranking for the top 6 resumes. While this may seem unimpressive, the most important job of a resume screener is to filter out any irrelevant applicants, and thus the specific ordering of “relevant” applicants could be thought of as less relevant. With this mindset, the hand-graded research shows that less complex models could filter out the irrelevant applicants from a pool of candidates, while the highly complex pre-trained BERT model could not.

However, these results only reflect an incredibly small sample size. The assurance of more successful less-complex models cannot be made with only ten resumes. The additional Kaggle directory research supports the findings from the hand-graded research and provides a more concrete answer on the effectiveness of certain models. This research determined the filtering ability of each model. The following graph demonstrates the results from running the three directories through the sciki-learn, BERT, and Numpy models. Word2Vec was not included in this portion of the research due to issues with directory implementation.

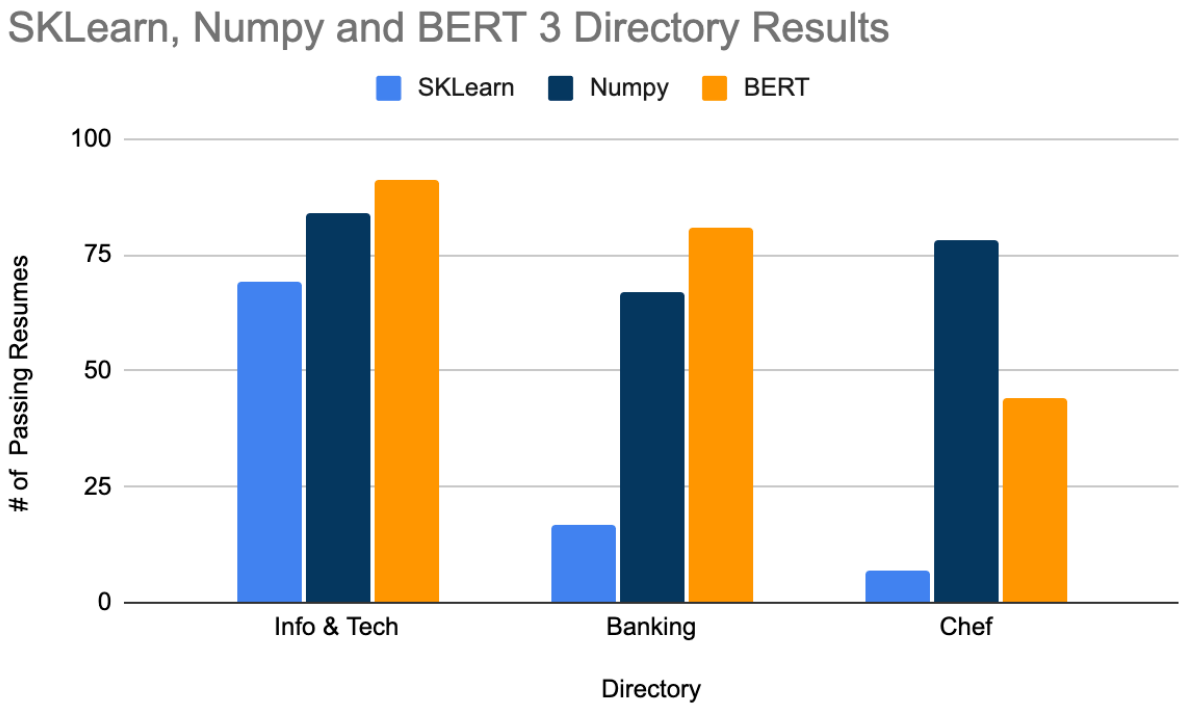

Figure 1 shows the number of resumes that passed the cosine similarity score threshold for three different resume directories across the SKLearn, Numpy, and BERT implementations

Firstly, this graph further demonstrates the ability of the scikit-learn model to identify relevance and filter out unsuitable resumes for the given job description. The model passed 69 Information & Technology resumes while only passing 17 banking and 7 chef resumes. This clearly demonstrates the success of scikit-learn's simple vectorization process. However, while Numpy.py performed the best in the hand-grading scenario, it proved unreliable with results not matching the proper job relevance. Lastly, the results from the pre-trained BERT model did not demonstrate a strong ability to recognize resume relevance, with an IT & INFO score of 91, a banking score of 81 and a chef score of 44. That being said, BERT did exhibit a trend of lower numbers given a decreasing connection between resume type and job description. In collaboration with the hand-graded method of research, this larger dataset confirmed the success

of scikit-learn's basic vectorization and the inaccuracies of the pre-trained BERT model. Overall, this demonstrated that the simpler models of word vectorization have a stronger ability to identify relevance between a resume and job description using cosine similarity.

6. Limitations

There were two large limitations that impacted the results found in this research. Unfortunately, the Word2Vec model produced inconsistent results and could not be included in the Kaggle results section. The erratic output of this model caused it to be incompatible with inputting a resume directory. Although the Word2Vec grading matched well with the hand-grading done, this could not be further proved with a larger sample size.

Secondly, the process of automated .pdf to .txt conversion caused random and inconsistent white space and formatting issues. While it was possible to manually clean the .txt files for the hand-grading portion of the research, it was necessary to rely on the pdftotext python library for the Kaggle research (Palmer, 2022). This led to inconsistent conversions to .txt files, with random white space and formatting errors. With this level of unpredictability for the conversion of over 300 resumes, there is a possibility that the data bears imprecise results.

7. Future work

Because we limited our research to use only existing Python libraries' cosine similarity and word vectorization methods, we could improve our testing by incorporating libraries outside of the Python coding language and comparing our resume screeners implemented in other coding languages, such as Java or C#, to see if there are any major differences. Another improvement to our program that we would have liked to make is having our code take in an entire resume dataset and return the ranked results. This would be extremely useful because it would allow us,

as researchers, to separate between the top, average, and subpar resume candidates and analyze the finer details that separate each of the resumes in these three categories. Unfortunately, we only performed tests on the complex large pre-trained language model, BERT, which still leaves unanswered questions regarding the cosine similarity scores on simpler smaller pre-trained language models in our resume screener. Our inference is that these simpler smaller pre-trained language models would perform similarly to our scikit-learn and Numpy resume screening implementation, but only one can tell with cold hard concrete evidence produced through testing.

8. Conclusion

Before conducting our research on resume screeners, word vectorization, and cosine similarity, we knew very little about how AI resume screeners operated and performed. To state an example, our initially proposed research question was to analyze potential areas of bias found in resume screeners due to the numerous job rejections which left us feeling that all resume screeners have at least some kind of bias to them. While there is still some truth in this from Deshpande, et al.'s research, bias in AI clearly does not account for this entire issue, which has now been backed by our findings through resume screening experimentation.

After performing two different resume screening experiments, both results were indicative of higher success in the simpler models. In one, we screened a set of ten resumes (6 computer science-related and 4 non-computer science-related) using two complex (BERT and Word2Vec) and two simplistic (Numpy and scikit-learn) models to calculate each resume's cosine similarity score to a ChatGPT entry-level software developer job description. Second, we screened all resumes within three distinct career directories (Information & Technology, Banking, and Chef) from Kaggle using one complex (BERT) and two simplistic (Numpy and scikit-learn) models and again found that the simpler, more straightforward resume screening

models demonstrate a solid aptitude for identify relevance between a resume and job description using cosine similarity. Our theory as to why stems from understanding how each model vectorizes words and/or text documents. In the case of these large complex pre-trained models, there are substantially more word vector connections it can recognize when compared to these simpler untrained models. For the sole purpose of screening resumes using cosine similarity, it was likely doomed to fail from the very beginning due to this innate pre-training. However, for other NLP tasks, such as understanding tasks like translation, Q&A, sentiment analysis, and sentence classification, these large complex pre-trained models are quite literally the ideal candidates for the job ("What is Bert?," n.d). While these complex models did not achieve our initial hopes for this resume screening task, we gained a strong understanding of the word vectorization process, the intricacies of the resume screening process, and the knowledge to apply these pre-trained effectively elsewhere.

References

Arora, S., & Kumari, N. (2021). Recruitment search engines for screening resumes through AI by using boolean search functions. *Journal of Asian Development, 7*(2), 16-26.

Carpenter, S. (2023, November 8). 44% of companies say AI is likely to replace employees next year. *Charter Communications*. https://ny1.com/nyc/all-boroughs/technology/2023/11/08/40--of-companies-say-ai-is-likely-to-replace-employees-next-year?cid=share_clip

Bhawal, S. (2021). *Resume Dataset* [Data set]. Kaggle. https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset

Deshpande, K. V., Pan, S., & Foulds, J. R. (2020, July). Mitigating demographic Bias in AI-based resume filtering. In *Adjunct publication of the 28th ACM conference on user modeling, adaptation and personalization* (pp. 268-275).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

FraiJ, J., & László, V. (2021). A literature review: artificial intelligence impact on the recruitment process. *International Journal of Engineering and Management Sciences, 6*(1), 108-119.

Horodyski, P. (2023). Recruiter's perception of artificial intelligence (AI)-based tools in recruitment. *Computers in Human Behavior Reports, 10*, 100298.

Lacroux, A., & Martin-Lacroux, C. (2022). Should I trust the artificial intelligence to recruit? Recruiters' perceptions and behavior when faced with algorithm-based recommendation systems during resume screening. *Frontiers in Psychology, 13*, 895997.

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

Palmer, J. (2022). pdftotext: Simple PDF text extraction. *GitHub*. https://github.com/jalan/pdftotext

Pant, D., Pokhrel, D., & Poudyal, P. (2022, March). Automatic Software Engineering Position Resume Screening using Natural Language Processing, Word Matching, Character Positioning, and Regex. In *2022 5th International Conference on Advanced Systems and Emergent Technologies (IC_ASET)* (pp. 44-48). IEEE.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems, 32*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12, 2825–2830*.

Rahutomo, F., Kitasuka, T., & Aritsugi, M. (2012, October). Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST* (Vol. 4, No. 1, p. 1).

Rehurek, R., & Sojka, P. (2011). Gensim–python framework for vector space modeling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic, 3*(2).

What is Bert?. NVIDIA Data Science Glossary. (n.d.). https://www.nvidia.com/en-us/glossary/data-science/bert/