TEXT SIMILARITY FOR POTENTIAL TALENTS

Richard Han
rickyhan24@gmail.com

ABSTRACT

In this project, I perform text similarity analysis on a dataset consisting of candidate profile text data to identify the most promising candidates for a specific type of job role. The resulting model ranks candidates based on similarity to keywords chosen by the recruiter searching for promising candidates. Secondly, I show how the recruiter can give preference to a given candidate and thereby modify the ranking accordingly so that the ranking system incorporates human feedback. The ranking system allows for identifying the most relevant candidates for the job and providing a more targeted and efficient hiring process.

1 Problem Statement

The task is to build a ranking system that ranks candidates based on their profile header's similarity to a set of keywords such as 'aspiring human resources'. The top candidates should have profile headers, also called job titles here, that are most relevant to the set of keywords. The ranking system should also adapt to the preferences of the recruiter who stars a given candidate.

2 Data

The data consists of 104 rows and 4 columns–'id', 'job_title', connection, and location. For the purposes of this project, I ignored location and connection. Connection tells how many connections the candidate has (think of LinkedIn connections). The two columns relevant to this project are the candidate 'id' and their 'job_title'–which is a text description of their role such as 'People Development Coordinator at Ryan'.

3 Text Similarity Analysis

3.1 TF-IDF Approach

In this approach, we convert each job title to a 200-dimensional vector. We also vectorize the keywords "aspiring human resources" and "seeking human resources" to 200-dimensional vectors. For each job title, we compute the cosine similarity score using each keyword so that we end up with two similarity scores for each job title. Note that I'm using the word 'keyword' to mean any word or phrase. Then, we average the two similarity scores to get a single similarity score for each job title.

Based on the average similarity score, we can rank the candidates. Here are the top ten candidates based on the ranking:

	id	job_title	connection	average_similarity_score
72	73	Aspiring Human Resources Manager, seeking inte...	0	0.553403
45	46	Aspiring Human Resources Professional	0	0.539892
32	33	Aspiring Human Resources Professional	0	0.539892
16	17	Aspiring Human Resources Professional	0	0.539892
57	58	Aspiring Human Resources Professional	0	0.539892
96	97	Aspiring Human Resources Professional	0	0.539892
2	3	Aspiring Human Resources Professional	0	0.539892
20	21	Aspiring Human Resources Professional	0	0.539892
35	36	Aspiring Human Resources Specialist	0	0.498402
5	6	Aspiring Human Resources Specialist	0	0.498402

The job titles match very well our given keywords. However, take a look at the scores for 'HR Senior Specialist':

	id	job_title	connection	average_similarity_score
7	8	HR Senior Specialist	1	0.0
25	26	HR Senior Specialist	1	0.0
37	38	HR Senior Specialist	1	0.0
50	51	HR Senior Specialist	1	0.0
60	61	HR Senior Specialist	1	0.0

'HR Senior Specialist' has an average similarity score of 0. However, we wouldn't want to exclude such job titles since 'HR' stands for human resources and 'human resources' is very much relevant to what we're looking for. This shows the limitations of the TF-IDF approach. It doesn't understand that 'HR' stands for human resources. So we move on to the word embedding approach.

3.2 Word Embedding Approach

In this approach, we're going to use a word2vec model. First, we tokenize each job title to get a list of tokenized words for each job title. We load the pre-trained model 'word2vec-google-news-300' and, for each job title, vectorize each word in the job title. Then, take the average of the word vectors to get a single vector for the job title. Each job title is now a 300-dimensional vector.

For each of our keywords 'aspiring human resources" and "seeking human resources', we vectorize them using the same word2vec model.

Next, we compute the average cosine similarity score between each job title and the set of keywords. We can then rank the candidates based on the average similarity score.

Here are the top ten candidates based on the ranking:

	id	job_title	connection	average_similarity_score	average_similarity_score_w2v
72	73	Aspiring Human Resources Manager, seeking inte...	0	0.553403	0.851056
73	74	Human Resources Professional	0	0.421235	0.829245
23	24	Aspiring Human Resources Specialist	0	0.498402	0.824286
48	49	Aspiring Human Resources Specialist	0	0.498402	0.824286
59	60	Aspiring Human Resources Specialist	0	0.498402	0.824286
5	6	Aspiring Human Resources Specialist	0	0.498402	0.824286
35	36	Aspiring Human Resources Specialist	0	0.498402	0.824286
29	30	Seeking Human Resources Opportunities	0	0.476375	0.823213
27	28	Seeking Human Resources Opportunities	0	0.476375	0.823213
98	99	Seeking Human Resources Position	0	0.461987	0.822808

If we take a look at the full dataset, we notice that there may be job titles that we are interested in but that have low similarity scores:

	id	job_title	connection	average_similarity_score	average_similarity_score_w2v
72	73	Aspiring Human Resources Manager, seeking inte...	0	0.553403	0.851056
73	74	Human Resources Professional	0	0.421235	0.829245
23	24	Aspiring Human Resources Specialist	0	0.498402	0.824286
48	49	Aspiring Human Resources Specialist	0	0.498402	0.824286
59	60	Aspiring Human Resources Specialist	0	0.498402	0.824286
...	...	...	...	...	...
50	51	HR Senior Specialist	1	0.000000	0.164199
7	8	HR Senior Specialist	1	0.000000	0.164199
25	26	HR Senior Specialist	1	0.000000	0.164199
37	38	HR Senior Specialist	1	0.000000	0.164199
82	83	HR Manager at Endemol Shine North America	0	0.000000	0.162396

At the bottom of the list, the word2vec model assigns a very low score to 'HR Senior Specialist', which is better than assigning 0 to it like TF-IDF did. Still, this is a limitation of word2vec models since it's not capturing the meaning of 'HR' in the way that we want it to. We might want to look at sentencetransformers to try and overcome the limitations of word2vec.

3.3 Sentence Transformers Approach

In this approach, we load a pre-trained model 'all-MiniLM-L6-v2' and encode the job titles and keywords, compute their average cosine similarity scores, and rank the candidates based on those scores.

By glancing at some of the job titles on the list, we can select a good cut-off point for average similarity score to be 0.4. If we filter the list to only candidates with scores above 0.4 and then rank them, we get:

	id	job_title	connection	average_similarity_score	average_similarity_score_w2v	average_similarity_score_ST
96	97	Aspiring Human Resources Professional	0	0.539892	0.822808	0.861254
2	3	Aspiring Human Resources Professional	0	0.539892	0.822808	0.861254
57	58	Aspiring Human Resources Professional	0	0.539892	0.822808	0.861254
45	46	Aspiring Human Resources Professional	0	0.539892	0.822808	0.861254
32	33	Aspiring Human Resources Professional	0	0.539892	0.822808	0.861254
...	...	...	...	...	...	...
50	51	HR Senior Specialist	1	0.000000	0.164199	0.487802
7	8	HR Senior Specialist	1	0.000000	0.164199	0.487802
80	81	Senior Human Resources Business Partner at Hei...	0	0.138826	0.634236	0.471104
82	83	HR Manager at Endemol Shine North America	0	0.000000	0.162396	0.461107
74	75	Nortia Staffing is seeking Human Resources, Pa...	1	0.169875	0.690149	0.425718

Now, we see that 'HR Senior Specialist' has a higher score (0.487802) than that given by word2vec (0.164199), better indicating its relevance. The sentence transformer model is better able to capture the meaning of 'HR' in the context of the job titles.

4 Re-ranking Candidates Based on Starring

Now that we have our ranking system built, we want to allow the recruiter to give feedback to the ranking system by starring certain candidates he or she deems particularly fitting. For example, suppose the recruiter stars the candidate with id 28. Then, the recruiter can include the job title for candidate 28 in the set of keywords, then compute the average similarity scores based on the updated set of keywords. Based on these new similarity scores, the recruiter can re-rank the candidates. This way, the old keywords remain relevant, but the new keywords shift the rankings in their direction. To implement the re-ranking system, I define a function called 'star' that takes in the id of the starred candidate and returns the sorted dataset based on the set of updated keywords that includes the starred candidate's job title.

5 Conclusion

In this project, I built a ranking system that ranks job candidates based on a set of keywords. To do this, I applied TF-IDF, word2vec, and sentence transformers. We saw that sentence transformers did a better job of capturing the meanings of terms such as 'HR' that were relevant to our search criteria. Finally, I provided a way for recruiters to re-rank candidates based on their preferences for certain candidates. The ranking system is useful for targeting job candidates in an efficient way, and the re-ranking system allows for human feedback to inform the selection process.