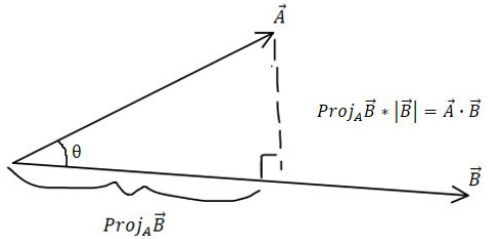
Ernst And Young GDS Hackpions 3.0

Team Members : Aavesh Kumar Sinha(Team lead)
Pratyush Harsh

Theme : CV Screening

Solution Approach and Intuition



- The approach is to find cosine Similarity / Dot product between 2 vectors
- Two vectors are closely related in high dimensional space when the dot product between the vectors yields a positive large number and when they are not related to each other then the cosine similarity between then is zero or sometimes a negative number
- The vector represents the word embedding of the vector obtained using Word2Vec (or Glove)
- Example v1 = “Captain_Marvel” and v2 = “woman” v3 = “Petrol”
- Similarity(v1,v2) = Embedding(v1)*Embedding(v2) = 10.99 (Large scalar)
- Similarity(v1,v3) = Embedding(v1)*Embedding(v3) = -3.44 (negative scalar)

Solution Approach Explained Further

- Here we got an idea that word2vec can be used for similarity measurement
- It can also be used for comparing related skills keywords in Job Description and the Resume

  For Example

- Similarity(“Python”,”Marketing”) = 0.78
- Similarity(“Python”,”sklearn”) = 12.44
- Similarity(“Java”,”Rest_API”) = 15.56
- Similarity(“Media”,”Marketing”)=13.56

But What About weighted Priority to the skills?

- For Giving a weighted priority to the skills I decided to find weighted dot product between the given skills embedding mentioned in Job description and the skills embedding extracted from resume
- I have used a premade library for resume parsing and Named entity retrieval
- Let’s have into a sample data

Example Case

The candidate must have five plus years of Experience in Python and Must have knowledge of hadoop , Java , REST and API

Keywords JD = [“Hadoop”,”java”,”rest”,”API”]

Keywords from Resume = [“Web”,”Spring”,”microservices”,”REST”]

- Keeping this example into mind let's proceed to the Algorithm

Algorithm

V1 = Embedding(Skill Keywords from Job Description)

V2 = Embedding(skills Keyword from Resume)

Alpha = Weighted Value to the Embedding corresponding to particular skill from Job description

V3 = [ ]

For i,j in V1,alpha {

    V3.append(i*j)

}

V3 = vectorize(V3)

V4 = Outer_Product(v3,v2) # Simultaneously finds outer product and dot product

Rank = sum_along_col(sum_along_row(V4))

Return Rank

Sort the resume as per rank and adjust the value of alpha parameter to obtain diverse mix of candidates

Conclusion

Diverse mix of candidates and top talents ensures

- The algorithm can be further improved by more datasets and the use of BERT for NER can further boost the performance
- Let’s proceed to Actual Demo