Project Report

On

Topic 2 - Text Processing

Prepared in partial fulfilment of the course

CS F469: Information Retrieval

Submitted to :

Dr. Abhishek, Dr. Vinti Agarwal

(Dept. of CSIS)

On November 23, 2020

Prepared By : GROUP 5

NAME	ID
ANIKET UPADHYAY	2017B3A71005P
RUSHIKESH ZAWAR	2017B1A70977P
KANAV	2017B3A70305P
RISHAV MISHRA	2017B3A70557P
NISHCHIT SONI	2017B3A71035P

TABLE OF CONTENTS

Steps performed in building the inverted index:

1. **Parse the documents and make a corresponding list of document titles:**

   We have implemented the *extract function,* which takes the path of the file as an argument and returns a list of titles, and a list of parsed document text for all documents in the corpus. It uses *BeautifulSoup,* a python library to extract data from HTML files.

   A list of all documents (in the form of tags and contents) present in the file is compiled using *BeautifulSoup* library and then titles are extracted from this list in *title_list*, and the parsed text in *doc_list.*

2. **Pre-Processing the document text:**

   As a second step, we perform pre-processing on each document text in the *doc_list*. It involves removal of unwanted punctuations, conversion to lowercase and then creation of a two-dimensional token list, where the element *token*[i][j] represents jth token of ith document.

3. **Frequency list creation:**

   This step involves calculating the occurrence frequency of each unigram or token created in the second step. The *get_unigrams* function uses the counter module to populate the frequency list.

   For Ex- *unigram[1]['the'] = 607* means doc id 1 contains 'the' 607 times.

4. **Creation of Inverted Index:**

   This step involves the creation of an inverted index for the corpus. It takes a list of dictionaries *freq_list created* in *step 3* and returns a dictionary of key-value pairs, with unique terms in the corpus acting as keys and the list of document id - frequency tuples as values.

   For Ex- a key value pair in the inverted index may appear like -

   'the':[(1,100), (2,250), ..]

5. **Storing the inverted index as JSON files:**

   The inverted index hence created is stored in the required directory if the directory path is given. In case no address is given, a directory is created to store the inverted index, frequency list, and title list describing the corpus. These files ensure that test_queries.py implementation never uses the text corpus again, and always queries the stored inverted index.

Using TF-IDF Scores to Rank The Documents

1. **Pre-processing of the query :**
   Similar to the pre-processing strategy used for documents in the index-creation part, here we pre-process the user query. It involves removing punctuations and converting each term to lower case.

2. **Calculating the normalized query weights (ltc)**
   The log frequency weight of term **t** in **query q,** according to ltc weighing scheme is given by:

$$\mathrm{wf}_{t,d} = \begin{cases} 1 + \log \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}.$$

$$\mathbf{idf_t = log10(N/df_t)}$$

$$\mathbf{w_{t,d} = (1 + log10(tf_{t,d})) * log(N/df_t)}$$

   The function *get_normalized_query_scores* takes in *query_terms* (the number of terms in the query), freq_list (the list of term-term frequency dictionaries returned from *get_unigrams*), and *inverted_index* as arguments, and returns the ltc weights for the user query.

3. **Calculating the normalized doc weights (lnc)**
   Similar to the calculation of normalised query weights, we have performed the calculation for normalised doc weights in the *get_normalized_doc_weights* function. The only difference is that instead of idf_t , the weight is simply given by the tf_t values of the document terms.

4. **Computing the final scores**
   For computing scores of doc-query similarity pairs, we have written the *compute_scores* function, which performs the dot product of weight vectors for query and document and returns the documents in decreasing order of their similarity scores to the user query.

5. **Printing the TOP 10 relevant documents**
   The *search* function combines the functionalities of steps 1-4 and prints the document id, title and similarity scores for the top 10 documents (documents with the largest scores).

MODEL1 search()

The Working

Examples

10 of 10

- **1 of 10**

Query	Top 10 documents	Score	Is the document relevant to the query?
Different kinds of sports	Asociación Alumni	0.058	Yes
	Arabian Prince	0.054	No
	Auto racing	0.046	Yes
	Aarhus	0.044	Yes
	Adventure	0.044	Yes
	Ambrosia	0.041	No
	Assistive technology	0.039	Yes
	Alabaster	0.038	No
	Antiparticle	0.038	No
	Audi	0.037	Yes

Precision = 0.6

- **2 of 10**

Query	Top 10 documents	Score	Is the document relevant to the query?
Apple company	Apple I	0.121	Yes
	MessagePad	0.067	Yes
	Alan Kay	0.065	Yes
	Apple Inc.	0.063	Yes
	AOL	0.044	Yes
	Alan Turing	0.036	No
	Asteroids (video game)	0.034	Yes
	Transport in Angola	0.033	No
	Aphrodite	0.03	No
	Assitive technology	0.028	Yes

Precision = 0.7

- **3 of 10**

Query	Top 10 documents	Score	Is the document relevant to the query?
Java Program	Applet	0.12	Yes
	Ada (programming language)	0.037	Yes
	APL (programming language)	0.035	Yes
	American Film Institute	0.023	No
	American National Standards Institute	0.02	No
	AWK	0.019	Yes
	Alan Kay	0.018	No
	ALGOL	0.017	Yes
	Assembly language	0.016	No
	Astronaut	0.016	No

Precision = 0.5

- **4 of 10**

Query	Top 10 documents	Score	Is the document relevant to the query?
Who gave the theory of relativity	Albert Einstein	0.062	Yes
	Astronomical unit	0.061	Yes
	Alfred Lawson	0.054	Yes
	Axion	0.051	Yes
	August	0.046	No
	Augustin-Jean Fresnel	0.039	No
	Almost all	0.038	No
	Applied ethics	0.037	No
	Alain Connes	0.037	No
	Andrey Markov	0.034	No

Precision = 0.4

- **5 of 10**

Query	Top 10 documents	Score	Is the document relevant to the query?
The railway system	Transport in Angola	0.098	Yes
	Abensberg	0.056	Yes
	Arecales	0.054	No
	Agrarianism	0.051	No
	Economy of Azerbaijan	0.048	Yes
	Park Güell	0.048	No
	Abbotsford House	0.048	Yes
	Assembly Line	0.048	No
	Alexander Mackenzie (politician)	0.045	Yes
	Telecommunications in Anguilla	0.044	No

Precision = 0.5

- **6 of 10**

AMBIGIOUS QUERY

Query	Top 10 documents	Score	Is the document relevant to the query?
Plant	Annual plant	0.062	Yes, tree plant
	Asterales	0.061	Yes, tree plant
	Apiaceae	0.064	Yes, tree plant
	Anadyr River	0.053	Yes, tree plant
	Alismatales	0.051	Yes, tree plant
	Agricultural science	0.046	Yes, tree plant
	Acacia sensu lato	0.046	Yes, tree plant
	Afro Celt Sound System	0.045	No
	Asparagales	0.043	Yes, tree plant
	Asteraceae	0.042	Yes, tree plant

Precision = 0.9

Ambiguity involved : We cannot say whether industrial plant or plant species was needed.

- **7 of 10**

Query	Top 10 documents	Score	Is the document relevant to the query?
Albert Einstein	Albert I	0.263	No
	Albert II	0.263	No
	Albert III	0.263	No
	America's National Game	0.064	No
	Albert Einstein	0.063	Yes
	Albert the Bear	0.059	No
	Absolute zero	0.057	Yes
	Albert Alcibiades, Margrave of Brandenburg-Kulmbach	0.056	No
	Axiom	0.055	Yes
	Albert of Brandenburg	0.052	No

Precision = 0.3

- **8 of 10**

RARE TERM

Query	Top 10 documents	Score	Is the document relevant to the query?
Alzheimer	Alfons Maria Jakob	0.065	Yes
	Aluminium	0.03	Yes
	A. E. van Vogt	0.03	Yes
	Astronaut	0.025	Yes
	Aspirin	0.019	Yes
	Agatha Christie	0.018	Yes
	Anarchism	0.0	No
	Autism	0.0	No
	Albedo	0.0	No
	A	0.0	No

Precision = 0.6

- **9 of 10**

Query	Top 10 documents	Score	Is the document relevant to the query?
Main cause of poverty	Extreme poverty	0.064	Yes
	Aegean Sea	0.05	No
	A modest Proposal	0.049	Yes
	Economy of American Samao	0.046	No
	Affirming the consequent	0.039	No
	Albert, Duke of Prussia	0.037	No
	Motor neuron disease	0.035	No
	Abscess	0.035	No
	Abortion	0.034	No
	Economy of Armenia	0.034	No

Precision = 0.2

- **10 of 10**

Query	Top 10 documents	Score	Is the document relevant to the query?
Tourist destination	Economy of Azerbaijan	0.061	Yes
	Acapulco	0.048	Yes
	Aruba	0.044	Yes
	Achill Island	0.039	Yes
	Alberta	0.035	Yes
	Abbotsford House	0.034	No
	Azerbaijan	0.033	Yes
	Aarhus	0.033	Yes
	Anguilla	0.032	Yes
	Alps	0.032	Yes

Precision = 0.9

IMPROVEMENT #1:

1. Issues with the TF-IDF based IR system:

- First of all, this model does not care for the term frequency saturation. It simply assigns higher scores to documents which have higher term frequency of the query terms.

  But, say a document contains 200 occurrences of the term “elephant”. It won't be exactly twice as relevant as a document that contains 100 occurrences of this term. It might so happen that after a large enough number of occurrences of the term “elephant”, say 100, the document is almost certainly relevant, and any further mentions of the term don’t really increase its relevance. So once a document is *saturated* with occurrences of a term, more occurrences don’t have a significant impact on score.

- Secondly, Let’s say that terms “cat” and “dog” have the same IDF values. If we search for “cat dog”, a document that contains one instance of each term is better than a document that has two instances of “cat” and none of “dog.” But the TF-IDF model ignores this fact.

- Thirdly, say “elephant” is one of the query terms. Now if a document happens to be really short and it contains “elephant” once, it’s a good indicator that “elephant” is an important term in the document. But if the document is very long and it mentions “elephant” only once, the document is probably not about elephants. Though the TF-IDF model involves normalisation with respect to document length, the improvement we are proposing handles it more robustly.

2. The improvement proposed:

We are proposing the BM25 model as an improvement over the TF-IDF model. This model tries to improve upon the rankings provided by our original model as it accounts for **document length** and **term frequency saturation.**

The scoring function used is:

$$RSV_d = \sum_{t \in q} \log\left[\frac{N}{\mathrm{df}_t}\right] \cdot \frac{(k_1+1)\mathrm{tf}_{td}}{k_1((1-b)+b \times (L_d/L_{ave})) + \mathrm{tf}_{td}}$$

tf_{td} : the frequency of term t in document d,
L_d : length of document d
L_{ave} : average document length in the corpus.
k1 : positive tuning parameter that calibrates the TF scaling.
b : determines the scaling by document length

k1 = 0 corresponds to a binary model (no term frequency), while a large value corresponds to using raw term frequency. b = 1 corresponds to fully scaling the term weight by the document length, while b = 0, corresponds to no length normalization. (NOTE: we have used k = 0.5 and b = 0.5 as the values of these hyperparameters)

3. How does the proposed improvement address the issues?

- To account for term frequency saturation, BM25 puts a bound on Term Frequency's contribution to the score by using a factor: **TF/(TF + k).** It still captures the fact that a document with higher term frequency is more relevant, but restricts its contribution to the document score in case the TF values are very very high. So, it accounts for the term frequency saturation problem.

- Secondly, it also rewards complete matches over partial ones, i.e. documents that match more of the terms in a multi-term query are more relevant than documents that have lots of matches for fewer of the terms.

  In our earlier example for "cat dog" query, consider k=1. If a document includes "cat dog", "cat" and "dog" each have a TF=1, so each term will contribute TF/(TF+1) = 1/2 to the score, for a total of 1. In some other "cat cat" document, "cat" has a TF of 2, so it will contribute TF/(TF+1) = 2/3 to the score. So, the "cat dog" document ends up getting a larger score. The TF-IDF scheme, instead, assigned higher weight to the "cat cat" document, as the contribution of logarithmic term-frequency is **1 + $\log_{10}$(2)** to the document score, which is higher than **1** + **1** for "cat" and "dog" separately in the "cat dog" document.

- Thirdly, BM25 rewards matches in shorter documents, while penalizing matches in longer documents. To penalize long documents, this adjusts k up if the document is longer than average, and adjusts it down if the document is shorter than average. To achieve this, BM25 multiplies k by the ratio dl/adl, where *dl* is the document's length, and *adl* is the average document length across the corpus.

  Also, there can be some collections where length matters a lot and some where it might not. To account for this, a parameter b is used by BM25 to control the importance of document length and can be varied as per the corpus/ requirement. Here b can be a value between 0 and 1. So, k is instead multiplied by the following factor based on dl/adl and b:

  (1 – b) + b*dl/adl

4. A corner case (if any) where this improvement might not work or can have an adverse effect:

The term frequency (TF) component is not lower-bounded properly by BM25. As a result, very long documents tend to be overly penalized.[REF] Such documents, which might match well with the query term, get scored unfairly.

IMPROVEMENT 1
DEMONSTRATION

The Working

Examples

3 of 3

1. QUERY: Main Cause of poverty

1a. Query(**Model1**)	Top 10 documents	Score	Is the document relevant to the query?
Main cause of poverty	Extreme poverty	0.064	Yes
	Aegean Sea	0.05	No
	A modest Proposal	0.049	Yes
	Economy of American Samao	0.046	No
	Affirming the consequent	0.039	No
	Albert, Duke of Prussia	0.037	No
	Motor neuron disease	0.035	No
	Abscess	0.035	No
	Abortion	0.034	No
	Economy of Armenia	0.034	No

1B. Query(**improvement1**())	Top 10 documents	Score	Is the document relevant to the query?
Main cause of poverty	Extreme poverty	2.583	Yes
	American Civil War	2.28	Yes
	A modest Proposal	2.253	Yes
	Agriculture	2.233	Yes
	Economy of Armenia	2.224	No
	Ankara	2.148	Yes
	Anthropology	2.131	No
	Aegean Sea	2.091	No
	Antisemitism	2.073	No
	Abortion	2.045	No

Improvement 1() vs Model 1 :

	Improvement1()	**Model1**
Precision	0.5	0.2

Mean Average Precision	=1/5*(1/1+ 2/2+ 3/3+ 4/4+ 5/6)= 0.966	1/2 * (1/1 + 2/3) = 0.83

FINAL CUSTOM SCORE	0.483	0.166

Final Custom Score = Mean Average Precision * Precision

2. QUERY: Albert Einstein

2a. Query(**Model1**)	Top 10 documents	Score	Is the document relevant to the query?
Albert Einstein	Albert I	0.263	No
	Albert II	0.263	No
	Albert III	0.263	No
	America's National Game	0.064	No
	Albert Einstein	0.063	Yes
	Albert the Bear	0.059	No
	Absolute zero	0.057	Yes
	Albert Alcibiades, Margrave of Brandenburg-Kulmbach	0.056	No
	Axiom	0.055	Yes
	Albert of Brandenburg	0.052	No

2B. Query(**improvement1**())	Top 10 documents	Score	Is the document relevant to the query?
Albert Einstein	Albert Einstein	3.638	Yes
	Axiom	3.254	Yes
	Absolute zero	3.109	Yes
	Android (robot)	3.089	Yes
	Atom	2.769	Yes
	Albert Schweitzer	2.697	Yes
	Alternate history	2.616	Yes
	Alan Turing	2.416	Yes
	Arthur Schopenhauer	2.413	Yes
	Alfred Lawson	1.679	Yes

Improvement1() vs Model1 :

	Improvement1()	**Model1**
Precision	1.0	0.3

Mean Average Precision	=1/10*(1/1+2/2+3/3+4/4+5/5+6/6+7/7+8/8+9/9+10/10) = 1.0	=1/3 *(1/5 + 2/7+ 3/9) = 0.273

FINAL CUSTOM SCORE	1.00	0.082

Final Custom Score = Mean Average Precision * Precision

3. QUERY: The railway system

3a. Query(**Model1**)	Top 10 documents	Score	Is the document relevant to the query?
The railway system	Transport in Angola	0.098	Yes
	Abensberg	0.056	Yes
	Arecales	0.054	No
	Agrarianism	0.051	No
	Economy of Azerbaijan	0.048	Yes
	Park Güell	0.048	No
	Abbotsford House	0.048	Yes
	Assembly Line	0.048	No
	Alexander Mackenzie (politician)	0.045	Yes
	Telecommunications in Anguilla	0.044	No

3B. Query(**improvement1**())	Top 10 documents	Score	Is the document relevant to the query?
The railway system	Aachen	2.068	Yes
	Amsterdam	2.043	Yes
	Economy of Armenia	2.03	Yes
	Alberta	2.009	Yes
	Aarhus	2.004	Yes
	Economy of Azerbaijan	1.936	Yes
	Assembly Line	1.923	No
	Alps	1.914	Yes
	Athens	1.912	Yes
	Ankara	1.842	Yes

Improvement1() vs Model1() :

	Improvement1()	**Model1**
Precision	0.9	0.5

Mean Average Precision	=1/9*(1/1+2/2+3/3+4/4+5/5+6/6+7/8+8/9+9/10) = 0.963	=1/5*(1/1+ 2/2+ 3/5+ 4/7+ 5/9) = 0.745

FINAL CUSTOM SCORE	0.867	0.3725

Final Custom Score = Mean Average Precision * Precision

IMPROVEMENT #2:

1. **Issues with the TF-IDF based IR system:**

   A user unaware of the exact terms to reflect his information need, might use some related words in the query and would in turn expect to find the relevant documents. It might so happen that the terms in the user query are not at all present in a document, but the query is expressive enough for the IR system to retrieve that document if it satisfies the information need of the user.

   The standalone TF-IDF based IR system, however, would not be able to retrieve such relevant documents if the user does not use the exact words in the query which he wants to search in the corpus. For ex- If a user searches for "Albert was intelligent". The user wishes to search for the famous scientist - Albert Einstein, and it might so happen that a document in the corpus talks about the smartness of Einstein and is not retrieved (because both the terms "albert" and "intelligent" are not present in the document).

2. **The improvement proposed:**

   We propose to use a **vector representation** based on **GloVe** (Global Vectors for words representation). We aim to identify the words related to the query terms, and use them (alongside the query terms) in our standard TF-IDF retrieval model.

3. **How does the proposed improvement address the issues?**

   GloVe is an algorithm that helps obtain vector representations for our query terms. The GloVe model has been trained on a global word-word co-occurrence matrix, which tabulates how frequently the words co-occur with one another in the corpus used for its training.

   We have used the *spacy* library in python, which uses this GloVe database and provides us words which have some relation to the query terms. Such relations have been learnt by the GloVe model using the training set. Then, in addition to the usual query terms, we use terms which have been found to be highly related to these query terms and search the corpus. This ensures that the IR system better understands the information needed by the user.

   The idea behind the improvement is that a certain word generally co-occurs more often with one word than another. For example- The word *ice* is more likely to occur alongside the word *water* than the word *roof*.

4. **A corner case (if any) where this improvement might not work or can have an adverse effect:**

This improvement requires an external corpus for finding similar words. That makes this model very computationally intensive and it takes a considerably larger amount of time to give results.

INNOVATION: Since this model is a bit computationally expensive, we have implemented a data structure that would store the top ‘n’ related terms obtained from the GloVe database. This improves the performance of the system. A bit of extra memory used to store the results saves a lot of computation time. In our model, we have stored the top 2 related words of some of the terms that we use in our queries to speed up the retrieval process.

IMPROVEMENT 2
DEMONSTRATION

The Working

Examples

3 of 3

1. QUERY: Main Cause of poverty

1a. Query(**Model1**)	Top 10 documents	Score	Is the document relevant to the query?
Main cause of poverty	Extreme poverty	0.064	Yes
	Aegean Sea	0.05	No
	A modest Proposal	0.049	Yes
	Economy of American Samao	0.046	No
	Affirming the consequent	0.039	No
	Albert, Duke of Prussia	0.037	No
	Motor neuron disease	0.035	No
	Abscess	0.035	No
	Abortion	0.034	No
	Economy of Armenia	0.034	No

1B. Query(**improvement2**())	Top 10 documents	Score	Is the document relevant to the query?
Main cause of poverty	Extreme poverty	0.042	Yes
	Abortion	0.022	No
	Abscess	0.019	No
	Ataxia	0.019	No
	Assault	0.017	Yes
	Antisemitism	0.017	Yes
	Adiabatic process	0.016	No
	Alcuin	0.016	No
	Agriculture	0.015	Yes
	Aphrodite	0.015	No

Improvement2() vs Model1 :

	Improvement2()	**Model1**
Precision	0.4	0.2

Mean Average Precision	=1/3*(1/1+2/5+3/6) = 0.63	1/2 * (1/1 + 2/3) = 0.83

FINAL CUSTOM SCORE	0.252	0.166

Final Custom Score = Mean Average Precision * Precision

2. QUERY: Viral Foreigner

2A. Query(**Model1**)	Top 10 documents	Score	Is the document relevant to the query?
Viral Foreigner	Acute disseminated encephalomyelitis	0.027	Yes
	Allegiance	0.021	No
	Mouthwash	0.016	Yes
	Hercule Poirot	0.013	No
	Aspirin	0.013	No
	Abortion	0.013	No
	Akira Kurosawa	0.01	No
	Anarchism	0.0	No
	Autism	0.0	No
	Albedo	0.0	No

2B. Query(**improvement2**())	Top 10 documents	Score	Is the document relevant to the query?
Viral Foreigner	Acute disseminated encephalomyelitis	0.05	Yes
	Abdominal surgery	0.033	Yes
	Allegiance	0.032	No
	Aspirin	0.026	No
	Mouthwash	0.023	Yes
	Abscess	0.019	No
	Abortion	0.019	No
	Hercule Poirot	0.015	No
	Americans with Disabilities Act of 1990	0.014	Yes
	Advanced Chemistry	0.013	No

Improvement2() vs Model1 :

	Improvement2()	**Model1**
Precision	0.4	0.2

Mean Average Precision	1/4 * (1/1+ 2/2+ 3/5+ 4/9) =0.76	1/2 * (1/1 + 2/3) = 0.83

FINAL CUSTOM SCORE	0.304	0.166

Final Custom Score = Mean Average Precision * Precision

3. QUERY: Enlighten Me!

1a. Query(**Model1**())	Top 10 documents	Score	Is the document relevant to the query?
Enlighten Me!	Advanced Chemistry	0.055	No
	Abner	0.047	No
	André-Marie Ampère	0.046	No
	Absalom	0.044	No
	Affidavit	0.044	No
	Augustin-Jean Fresnel	0.043	No
	Aaron	0.04	No
	The Alan Parsons Project	0.04	No
	A. A. Milne	0.037	No
	ABBA	0.037	No

1B. Query(**improvement2**())	Top 10 documents	Score	Is the document relevant to the query?
Enlighten Me!	Amos Bronson Alcott	0.025	Yes
	André Gide	0.021	No
	Adiabatic process	0.02	No
	Anna Kournikova	0.02	No
	Arminianism	0.018	Yes
	Akira Kurosawa	0.017	No
	AK-47	0.016	No
	Hercule Poirot	0.015	No
	Aristophanes	0.015	No
	Alan Turing	0.015	No

Improvement2() vs Model1 :

	Improvement2()	**Model1**
Precision	0.2	0

Mean Average Precision	=1/2 *(1/1 + 2/5) = 0.7	0

FINAL CUSTOM SCORE	0.14	0

Final Custom Score = Mean Average Precision * Precision

REFERENCES:

(n.d.). Retrieved November 23, 2020, from https://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). https://doi.org/10.3115/v1/d14-1162

Yuanhua Lv and ChengXiang Zhai. 2011. Lower-bounding term frequency normalization. In Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM '11). Association for Computing Machinery, New York, NY, USA, 7–16. DOI:https://doi.org/10.1145/2063576.2063584

Wikipedia contributors. (2020, October 20). Okapi BM25. In Wikipedia, The Free Encyclopedia. Retrieved 17:03, November 23, 2020, from https://en.wikipedia.org/w/index.php?title=Okapi_BM25&oldid=984459267

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

Rakhmanberdieva, N. (2018, December 14). Word Representation in Natural Language Processing Part II. Retrieved November 23, 2020, from https://towardsdatascience.com/word-representation-in-natural-language-processing-part-ii-1aee2094e08a