Question & answering system based on retrieval

Ziyi Fan[1, †], Mingkan Jiang[2, †], Wei Li[3, †], Zihan Weng[4,5, †]

[1]Computer Science, University of California, Irvine, Irvine, California, 92612, U.S.
[2]Engineering, University of New South Wales, Sydney, NSW, 2052, Australia
[3]Computer Science, Xidian University, Xi'an, Shaanxi, 710126, China
[4]Mathematics, Georgia Institute of Technology, GA, 30332, U.S.

[5]zweng40@gatech.edu
[†]These authors contributed equally.

Abstract. The principle of the question and answering system is nowadays one big branch of natural language processing. When a user enters a query, the system should find the question that is most similar to the entered question stored in the datasets. Then directly return the answer corresponding to the found question. Two models are implemented to better calculate the similarity between entered questions and questions in the corpus: TF-IDF and Word2Vec. After one hundred experiments to test the accuracy of these two models, the result shows that the Word2Vec has 81% accuracy and TF-IDF has only 49% accuracy. It is concluded that Word2Vec has a better accurate performance than TF-IDF. There is also an optimized storage method by using the inverted list for the corpus to improve the model's efficiency.

Keywords: Machine learning, question and answering system, TF-IDF, Word2Vec.

1. Introduction

Industries use Natural language processing (NLP) equipped tools more and more to gain insights from data, run automated routine tasks, and help everyone to live a more convenient lifestyle. Considering the increasing needs, more attention is needed to the question and answering (QA) system, a critical NLP problem.

NLP makes it possible for machines to understand human language. It is nowadays considered one of the hottest and essential branches of Artificial Intelligence (AI). It aims to simulate the human brain by building neural networks to parse text and speech based on deep learning [1]. It is commonly applied to machine translation, speech recognition, email detection, etc. NLP has progressed rapidly, but there are still many problems. The structure and logic of human language are very complex. Hence dealing with these uncertainties and limitations is one of the important research directions. Also, NLP requires a huge database, and the classification and accuracy of this database is a challenge that must be faced. For example, misinformation brought by modern society's online news causes AI to have a lot of difficulty assessing the credibility of the information [2].

Based on functions provided by NLP principles, the QA system allows users to ask questions in human language and give immediate responses as brief answers. QA systems are now extensively utilized in search engines, conversational User Interface mobile, and virtual assistants. It is believed by most that QA systems have relatively good performance at giving answers to simple pieces of information [3]. With more complex questions presented to them, however, users should expect to get possible candidate answers [4] and then find the answer to the question on their end by browsing through the given list. These existing problems show people the room for improvement on this topic, which leads to the situation that more people are putting their eyes on these issues and trying to resolve them.

The concept and application of word embedding models, or distributed representations, have recently been the most spectacular development in NLP. GloVe is one of the most salient algorithms in NLP. GloVe, Global Vectors for Word Representation, is a word representation based on word count and statistics. Some advantages of the GloVe model can be quick training, the capability of extending to the large corpus, and good performance on small corpus and small vectors [5]. GloVe utilizes the co-occurrence matrix and considers both the local and global information of the word. Word co-occurrence is the number of words i in one word j. Usually, GloVe chooses the interval that includes 21 words and is centered on j given in the texts. Our experiment uses two other embedding algorithms similar to GloVe: TF-IDF and Word2Vec.

2. Method

2.1. TF-IDF

First, we tried to use TF-IDF (term frequency-inverse document frequency) as our algorithm to solve the QA system. TF-IDF is a statistical method for information retrieval and data mining. It is a fundamental document representation method in the NLP field [6]. TF-IDF can evaluate a word's importance in a document's dataset. In particular, the term frequency (TF) is referred as the local term weight, while the inverse document frequency (IDF) is referred as the global term weight [7]. The more frequent a word occurs, the more important it is. However, the importance decreases inversely proportional to its coveted frequency in the corpus. For example, suppose the TF of a word in an article is high, and it rarely appears in other articles, and the word can be used for class distinction.

TF-IDF is a reliable estimate of information [8]. In this algorithm, the given corpus is "train-v2.0.json". This corpus contains groups of paired questions and answers in the JSON format. First, we wrote the unprocessed question and the unprocessed answer into two lists. During this process, we ensured the index of the question and the answer in the list be the same. Then we counted the number of corpora to understand the corpus better. Next, other statistics were collected: the total number of characters in the question, the answer list, and the number of different characters. The frequencies of occurrence of each character were then sorted. We also applied the drawing function to show the image of the statistics. Finally, the question list was subjected to a series of natural language preprocessing operations. The procedures are following:

Step 1. Convert words to lower case.

Step 2. Stemm words by using porter stemming.

Step 3. Process the numbers: numbers such as 44 and 415 are seen as words, and new words are defined as "#number."

Step 4. Filter stop words.

Step 5. Remove words that appear very infrequently by using the Zipf's law.

The Zipf's law was also applied to process the text. The frequency of words is inversely proportional to their frequency of registration [9]. According to the Zipf's law, it constitutes an astonishing amount of regularity in the use of natural language processing [10]. The occurrences of the n^{th} common word is $1/n$ of occurrences of the most common word. When n is large enough, the accumulation of the series, or the total number of words in the text, is $1/n$, which is approximately equal to $ln(n)$. Here is our pseudocode for preprocessing text:

After preprocessing the words in the question list, each text was converted into a TF-IDF vector. Then the converted results were stored in a X matrix. The size of X matrix is equal to N times D. Here N is the number of questions, and D is the size of the dictionary library. According to the user input question, the three most likely answers were found based on the following operation requirements:

Step 1. Perform a series of preprocessing for questions.
Step 2. Convert the texts into TF-IDF vectors by using the vectorizer.
Step 3. Calculate the similarity of the questions in each library.
Step 4. Find the top 3 answers with the highest similarity.

Simplicity and high-speed are the advantages of TF-IDF. However, TF-IDF only applies word frequency to measure how important the words are. It makes the words independent of each other, and the sequence of the information is reflected badly. TF-IDF is also susceptible to skewed datasets, leading to the underestimation of IDF when many documents are in a certain category. Based on the TF-IDF formula, the IDF will be higher for some rare words, which may often be mistaken for keywords. Moreover, this algorithm will calculate and compare the similarity to all the other questions in the library. This could be a very inefficient approach if there are many questions in the library.

Algorithm 1 Preprocessing Raw Text

```
Input: Text of a question from the question list
Output: Processed Text
for all letter in text of a question do
    letter ← letter.lowercase
    letter ← stem(letter)
    if letter $\in$ number then
        letter ← '#number'
    end if
    if letter $\notin$ stop words then
        Add letter in word list
    end if
end for
Result: word list
```

2.2. Word2Vec

Based on that, the algorithm was changed from TF-IDF to Word2Vec. Word2Vec is one of the most utilized techniques since it was published in 2013. It takes texts as the training data for a neural network. Word2vec has a set of related models that can generate word embeddings. These models are shallow two-layer neural networks that can be trained to reconstruct the linguistic context of words [11]. The resulting embedding captures whether words appear in similar contexts or not. Even though Word2Vec is not the first or the last algorithm to discuss vector spaces, embeddings, analogies, and similarity metrics, the advantages of Word2vec are undeniable. Word2Vec is simple and flexible to perform such that anyone can download the code and use it [12]. From the perspective of accuracy, Word2Vec performs better than TF-IDF. The results of the Word2Vec have similar contexts to similar cosine embeddings, which fits JR Firth's distributional assumption [13]. It needs a little preprocessing, which means little memory is needed. It can capture contextual and semantic relationships between different words. Also, the human effort in labeling the data is less than other algorithms since the training is unsupervised.

Word2Vec has two main models: the Continuous Bag-of-Word Model (CBOW) and the Skip-grams algorithm. Word2Vec uses these two models to generate a distribution of the word representation. The concept of skip-grams is to take a word as the central word in each iteration and try to predict the context words within its range. The CBOW algorithm is similar to the Skip-grams algorithm. Whereas, in the

CBOW model, the contextual words before and after the contents determine the central word. Hence, CBOW is faster, while Skip-grams does a better job with less commonly used words [14].

In this algorithm, each embedded word that is formed in a matrix is preprocessed into the corpus. It uses the Glove2Word2Vec function in the Gensim module to convert the GloVe word vector into a Word2Vec vector in another file. The input GloVe word vector file in this code is "./glove.6B.300d.txt". It stores and queries word vectors in the output file named "./glove2word2vec.6B.100d.txt".

Then it read each embedded word and converted it into a matrix. This is a matrix with the dimension of D times H. Here, D represents the dictionary size, and H represents the word vector's size. If a given word does not appear in the dictionary, it neglects this given word. Hence, we can get the sentence vector for each word straightforwardly:

$$Sentence\ Vector = the\ Average\ of\ Word\ Vectors \tag{1}$$

After preprocessing the data, the system can provide the answers in response to users' questions. We can do the following operations to get the results. Instead of the above TF-IDF's method, we used an inverted list to find the problem in the library that was similar to the question input. Then, the cosine similarity was calculated and compared for these problems. The requirements were the following:

Step 1. Define a simple inverted list by using *defaultdict*.
Step 2. Given the user-entered questions, take the union of all documents containing these words and the inversion list to screen candidates.
Step 3. Convert the user-entered question into sentence vectors.
Step 4. Calculate the cosine similarity between questions in the corpus.
Step 5. Use the priority queue to find three indexes of answers with the greatest similarity.
Step 6. Return answers to the top three questions with the highest similarity.

Here is our pseudocode for finding the top three answers:

Algorithm 2 Calculating the similarity

```
Input: user-entered question
Output: three answers with the highest cosine similarity
  preprocessing the question according to Algorithm 1
  candidates ← set()
  for all word in the preprocessed question:
      candidates ← candidates ∪ inverted list{word}
  end for
question vector ← Average Word Vectors according to (1)
question vector ← question vector / L2 norm vector of question vector
cosine similarity ← L2 norm vector of one @ question vector
Result: three answers with the highest cosine similarity.
```

3. Experiment

3.1. Dataset

3.1.1. train-v2.0.json. train-v2.0.json contains pairs of questions and answers in JSON format. We need to write the parser to extract questions and answers into lists. The link of train-v2.0.json is https://github.com/chrischute/squad/blob/master/data/train-v2.0.json

3.1.2. glove.6B. GloVe is an unsupervised learning algorithm that obtains word vector representations. The aggregated global word-to-word co-occurrence statistics from the corpus are trained. The resulting representation exhibits the linear substructure of the word vector space. The link to glove.6B is https://nlp.stanford.edu/projects/glove/.

3.2. Result

We used experiments to test and validate our model performance. Our test set had two representations with the same meaning for each question. Among them, each of the first expression was from the question library, which had a standard answer. Each second expression was a synonymous translation to the corresponding first expression. We can get the model accuracy by calculating the outputting results from the returning answer to the same question with different expressions.

In this experiment, we input the same 50 sets of questions to both models, and the final accuracies were 49% and 81% respectively. Meanwhile, accuracies of two models were different such that the latter was higher than the former, which indicated that the performance of the model has been improved.

The following two tables are results of 10 random sample questions in the 50 sets of questions from different models. In this example of 10 sets of data, accuracies of two models were 55% and 70% respectively. In the second, third, and eighth sets of examples, Word2Vec was able to compensate for questions and answers that were completely unrecognized by the TF-IDF. However, in sets 4 and 5, Word2Vec appeared to have inferior cases compared to TF-IDF. In general, the Word2Vec had more improvements on accuracy compared to TF-IDF.

Table 1. TF-IDF and Word2Vec implementations' results for 10 sets of examples.

Num	Question	TF-IDF Answer	Valid	Word2Vec Answer	Valid
1.1	Which airport was shut down?	Chengdu Shuangliu International Airport	1	Chengdu Shuangliu International Airport	1
1.2	Which airport is closed?	Newburgh, New York	0	Plymouth City Airport	0
2.1	What government blocked aid after Cyclone Nargis?	British protectorate	0	Myanmar	1
2.2	Which government stopped aid after Hurricane Nargis?	1954	0	Myanmar	1
3.1	What areas did Beyonce compete in when she was growing up?	Warsaw	0	singing and dancing	1
3.2	What areas did Beyonce do?	large shopping areas.	0	singing and dancing	1
4.1	What percentage of Houston's population is African-American?	25%	1	Kathmandu Metropolitan City	0
4.2	What is the percentage of Houston's population that is African American?	25%	1	37.9	0

Table 1. (continued).

Num	Question	TF-IDF Answer	Valid	Word2Vec Answer	Valid
5.1	What federal law did the United States pass in 1997, in response to the LaMacchia Loophole?	No Electronic Theft Act	1	Kathmandu Metropolitan City	0
5.2	What federal law was passed in 1997 in response to the LaMacchia Loophole?	No Electronic Theft Act	1	The United States Code	0
6.1	Which book of the Bible talks about the use of wine during Jewish feasts?	Deuteronomy	1	Deuteronomy	1
6.2	Which book of the Bible talks about using wine in the Jewish holiday?	Deuteronomy	1	Deuteronomy	1
7.1	In what month was the preview of Spectre released in movie theaters?	July	1	July	1
7.2	In what month did the Spectre trailer hit theaters?	July 2015	0	six	0
8.1	Who also preached that one was not a member of his or her city?	Mack Newton	0	Augustine	1
8.2	Who preached that he was not a member of his or her city?	Mack Newton	0	Augustine	1
9.1	Which social media site was used for complaining about the Indian censoring?	Twitter.	1	Twitter.	1
9.2	Which social media website is used to complain about India's censorship?	her "continuance of hysterics" over a "miserable trifle"	0	Twitter.	1
10.1	When will work being on the follow-up to Spectre?	spring 2016	1	spring 2016	1
10.2	When will the follow-up work on Spectre take place?	spring 2016	1	spring 2016	1

Table 2. Accuracies of TF-IDF and Word2Vec models under 50 sets of questions.

	TF-IDF	Word2Vec
Accuracy	49%	81%

4. Conclusion

Two models were implemented to solve our QA system throughout the system design process. The first model was the TF-IDF model, which was inefficient and easily susceptible to skewed datasets. Based on that, it was altered to the Word2Vec model. After improving the model performance, the experiment attested to this: the TF-IDF model only performs accuracy with 49%, whereas the Word2Vec model performs accuracy with 81%. Although Word2Vec has shown a much better result in consideration of the accuracy, the efficiency and accuracy performance can still be improved in many ways. In the future, researchers can add other algorithms to the neural network in the model, including back propagation, Gradient calculation, and Gradient inspection to reduce loss function.

References

[1] Nadkarni PM, Ohno-Machado L, and Chapman WW. (2011). *Natural language processing: an introduction.* (J Am Med Inform Assoc).

[2] Przybyła P, and Soto A. (2021). *When classification accuracy is not enough: Explaining news credibility assessment*. Information Processing & Management, Volume 58, Issue 5, Article 102653, ISSN 0306-4573.

[3] Gomes J, de Mello RC, Ströele V, and de Souza JF. (2022). *A study of approaches to answering complex questions over knowledge bases.* (Knowl Inf Syst).

[4] Mendes A, and Coheur L. (2013). *When the answer comes into question in question-answering: Survey and open issues.* Natural Language Engineering, Volume 19, Issue 1, Page 1-32, ISSN 1351-3249.

[5] Pennington J, Socher R, and Manning C. (2014). *GloVe: Global Vectors for Word Representation.* Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Page 1532–1543. (Association for Computational Linguistics).

[6] Kim D, Seo D, Cho S, and Kang P. (2019). *Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec.* Information Sciences, Volume 477, Page 15–29, ISSN 0020-0255.

[7] Mohammed M, and Omar N. (2019). *Question classification based on Bloom's taxonomy cognitive domain using modified TF-IDF and word2vec*. PloS One, Volume 15, Issue 3, e0230442.

[8] Arroyo-Fernández I, Méndez-Cruz C-F, Sierra G, Torres-Moreno J-M, and Sidorov G. (2019). *Unsupervised sentence representations as word information series: Revisiting TF–IDF.* Computer Speech & Language, Volume 56, Page 107–129, ISSN 0885-2308.

[9] Lestrade S. (2017). *Unzipping Zipf's law.* PloS One, Volume 12, Issue 8, e0181987.

[10] Moreno-Sánchez I, Font-Clos F, and Corral Á. (2016). *Large-scale analysis of Zipf's law in English texts*. PloS One, Volume 11, Issue 1, e0147073.

[11] Mikolov T, Chen, K, Corrado G, Dean J. (2013). *Efficient Estimation of Word Representations in Vector Space*. (ArXiv).

[12] Church K. (2017). Word2Vec. Natural Language Engineering, Volume 23, Issue 1, Page 155-162, ISSN 1351-3249.

[13] Goldberg Y, and Levy O. (2014). *word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method*. (ArXiv).

[14] Rong X. (2014). *word2vec Parameter Learning Explained*. (ArXiv).