Data Science Crew

Quora Question Pairs

Can You Identify Question Pairs That Have The Same Intent?

Gaurav Joshi
Gaurav.Joshi@iiitb.org
MT2018035

Tushar V Bharadwaj
Tushar.Bharadwaj@iiitb.org
MT2018126

Manpreet Singh Tuteja
Manpreetsingh.Tuteja@iiitb.org
MT2018058

***Abstract*—Quora is a community driven platform to ask questions and to connect with people who contribute their unique insights and quality opinion on topics of interest. This allows people to learn from each other. However, due to the large number of people using the platform there are a lot of questions that are repeatedly asked by its users.**

This paper is a report on our project to determine whether the intent of the given two questions is same or not. Our approach uses machine learning algorithms to resolve this problem by classifying these questions into their respective buckets with high accuracy.

***Index Terms*—TF-IDF, XGBoost, Data Visualization, NLP**

I. Introduction

The task is to determine if two questions are similar to each other. We're given a dataset of over 400k question pairs, and a test set of around 1k questions. Currently Quora uses Random Forest to identify duplicate questions, in this project we are using advanced text processing and machine learning algorithms to predict if the given question pairs have the same intent.

The pair of questions in our dataset will have a high probability to be similar if they have a lot of "rare terms" or nouns common to them, and their negatives will differ only by small semantic differences. However, this may not always be the case. There maybe two questions that may have different semantics but may mean the same. Example : Why did the number 45 win presidency in 2016? How did donald trump win 2016 presidential elections?

So, our main goal would be to design the model in such a way that it would be capable of determining similarity in questions even if they differ in words and phrases with same semantic meaning.

II. Approach Taken

Every sentence has a syntax and with it is associated a semantic. Since the deep learning alogrithms wasn't the scope of our solution it became very obvious that we had no control over the semantics of the questions, which leaves us with the syntax. We used text processing techniques to leverage the possibilty of feature extraction given two questions. The outcome of which was 43 features that would help our models predict solely on the basis of syntax of the two questions. We used these extracted features on different classification models and analysed their performance.To further alleviate the performance we used stacking on most stable classifiers for our data, which were Random Forest and XGB. Another potential level up we used were post processing techniques to reduce the log loss thereby increasing the accuracy.

III. Data Set

We're given a dataset of 400K+ questions, where in 148.9K questions were duplicates, and 254K questions were not, with about 11.06 words per question. Initially the data was skewed, having less than 40 pcnt of questions being duplicate, and the rest non-duplicates. Other than that, we found out that a good portion of questions were repeatedly asked. Some even having the frequency

of 120 and 180 in the dataset. On closer inspection it suggested that the questions which were repeated multiple number of times had a very high probability of being a duplicate question.

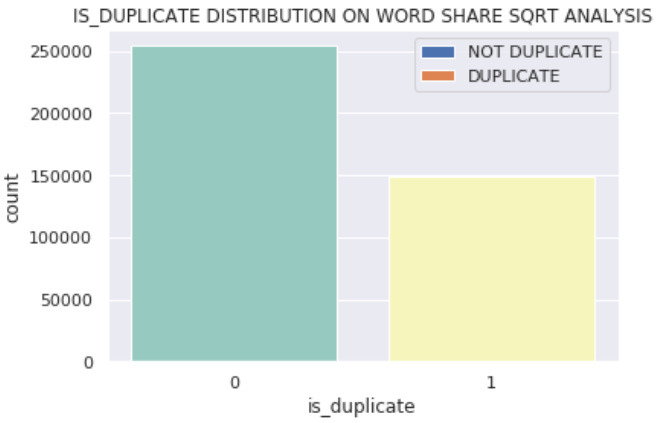

Fig. 1. Distribution of Duplicates and Non-Duplicates

Following attributes were avaialble in the training data set :

- Pair ID
- Question 1, Question 2 IDs
- Question 1, Question 2
- Is It Duplicate?

IV. Feature Engineering

It is rightly said that words are a powerful tool and our questions had enough of it. We extracted features out of these collection of words example character count, word differences,word densities, word weight and used the same to train our model.

A. Data Cleaning

Our training data, had redundant information, spelling mistakes, and stop words example: is, the etc which would eventually add up reducing the performance of the models in predicting the results accurately.
To address this we used the following data cleaning techniques.

- **Removal of stop words :** We created a dictionary of top 50 most frequent occurring words in the entire data set provided to us. This allowed us to create our own dictionary of stop words. To match the word count we did not consider these stop words as there was no real semantic associated with them. However on altering the number to more or less than 50 resulted in decreased performance of the model wrt to log loss.
- **Lemmatization :** Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. By lemmatizing we brought all the words to the same tense. Ex: Studying, Studied will be converted to Study.
- **Replacing null values :** After extracting the required features, our analysis suggested that some of these features had null values, however it meant that the calculated feature for that particular question was beyond norm or didn't add to the meaning. To address this we replaced them with default values depending on on the appropriate nature of the feature.
- **Over sampling:** Our dataset had more non duplicate values than duplicates , this suggested that our dataset was skewed and hence affected our results. We used oversampling techniques to fix this issue, We did SMOTE over sampling, which increased our data set to 508,774 rows. Which helped us improve our average log-loss by a very small amount however, it started producing non biased predictions.

B. Main Features

These features significantly impacted the resultant log loss of the model.

- **Graph Intersection Feature :**To understand the same consider a graph with (Q1, Q2) and (Q2,Q1) as edges, and then count all their intersections. The underlying idea is that, higher the frequency of occurrence of a question, more is the probability of that the question pair being duplicate, irrespective of the question it is paired with.
- **Question Frequency** Frequency count of questions resulted in a direct effect in decreasing the log-loss. It is obvious that the

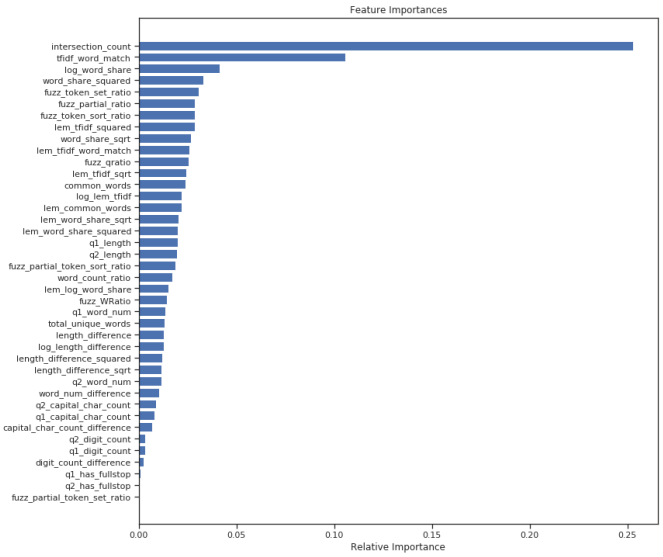

Fig. 2. Feature Importance

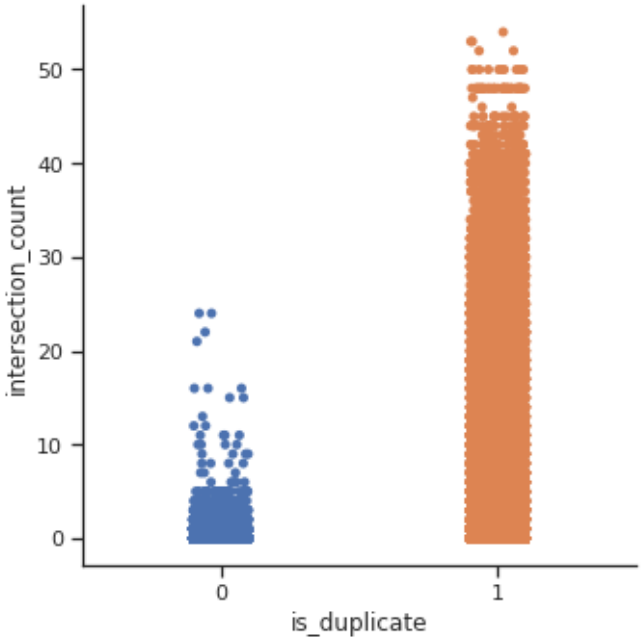

Fig. 3. Relevance Of Intersection Count

questions asked more frequently had a higher probability of being asked again.

- **TF-IDF :** Term frequency and inverse document frequency is a numerical statistic that reflects how important a particular word is in the whole document. TF-IDF based model paired with XG-Boost alone gave us a log-loss of 0.3 in public leaderboard. The additional help of term frequency was that it allowed us to determine the stop words easily which we could eliminate in future. We used two versions of TF-IDF features, ie tf-idf that was calculated on Lemmatized dataset and other on base data.

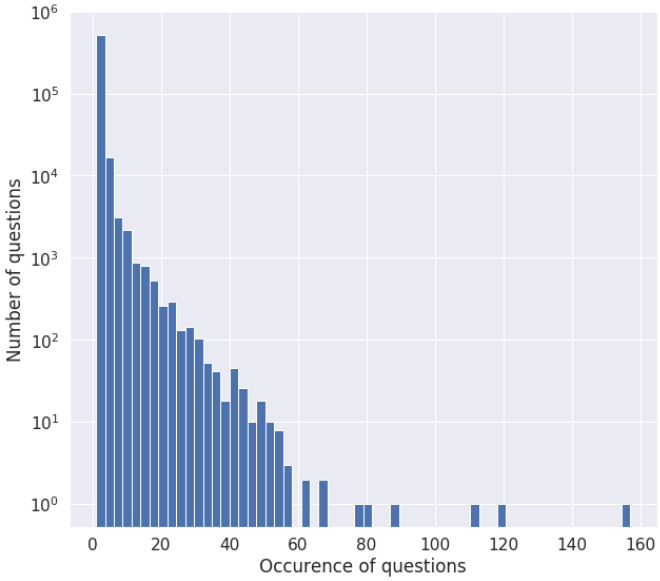

Fig. 4. Effect Of Question Frequency

$$\mathbf{tf}(t,d) = \frac{f_d(t)}{\max_{w \in d} f_d(w)}$$

$$\mathbf{idf}(t,D) = \ln\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

$$\mathbf{tfidf}(t,d,D) = \mathbf{tf}(t,d) \cdot \mathbf{idf}(t,D)$$

$$\mathbf{tfidf}'(t,d,D) = \frac{\mathbf{idf}(t,D)}{|D|} + \mathbf{tfidf}(t,d,D)$$

$f_d(t) :=$ frequency of term t in document d

$D :=$ corpus of documents

Fig. 5. Term Frequency and Inverse Document Frequency

- **log_word_share :** Log of number of shared words between two sentences. More the number of words shared between two sentences higher the probability of them being duplicate.

C. Other Features

Features that resulted in decreasing log loss.

- **Q#_word_num :** Number of words that are present in the Questions, since there is a higher probability of questions being similar to each other if the number of words present in both of them are within a permissible range. Though this might seem redundant since word num difference would provide us the range, however in practice this feature did seem to benefit us with improving the accuracy.
- **Word_num_difference :** This is the difference between the number of words in both the questions. The higher the difference in these meant a higher probability of the questions not being duplicate.
- **Q#_has_full_stop :** Checking wether either of the questions have fullstops.
- **Q#_digit_count :** The number of numeric characters that are present in each of the sentences.
- **Q#_digit_difference_count :** The difference in the number of numeric characters present in both the questions.
- **Q#_capital_char_count :** The number of capital characters in the sentence.
- **Q#_capital_char_count_difference :** Difference in the number of capital characters in both the sentences.
- **Variations of the above feature-set :** We've used variations of the above features and their combinations, example :- Log of all the above features, difference, squared, etc. Some of the features were skewed, by taking log or square of it, we were able to bring them closer on the same scale.
- **Total Unique Words :** While calculating TF-IDF, we were able to get a list of all term frequencies, we gathered a set of rare words, which behaved like "Nouns" because of their uniqueness with respect to their occurrence of themselves in the document. With this we could tell that even if few of these words are similar in both the questions then they are both talking about same topic or are atleast slightly co-related to each other.

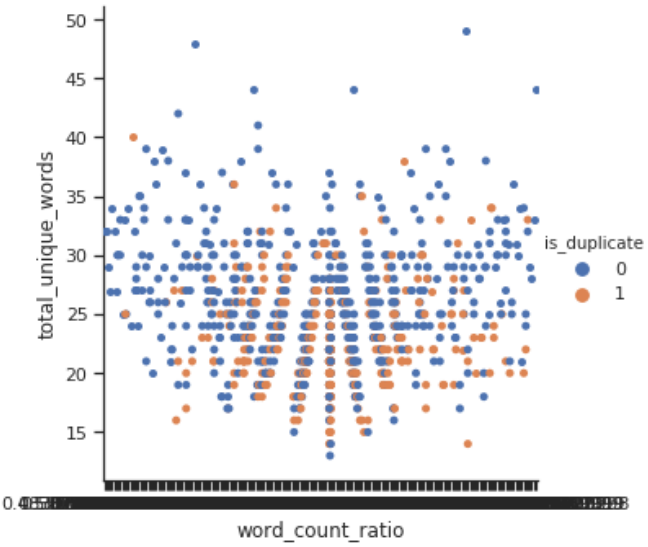

Fig. 6. Total Unique words, word count ratio and it's effect on is_duplicate

- **Fuzzy wuzzy Feature-set :** To account for the spelling errors in two strings, we took a similarity measure that uses Levenshtein distance to do the same task. If two strings are to be compared and they have a higher Levenshtein distance then they probably are the same terms. Fuzzy wuzzy is a string similarity measure, which gives a score out of 100 that tells us how close to each other are two strings based on the Levenshtein distance.
- **Cosine similarity :** Cosine matching is a way to determine how similar two strings are to each other. We are calculating the angle between the two feature vectors.

V. Models

A. Level 1 Model Training

Having gone over 200 iterations of constantly building and testing models throughout the project, we tried various models with different

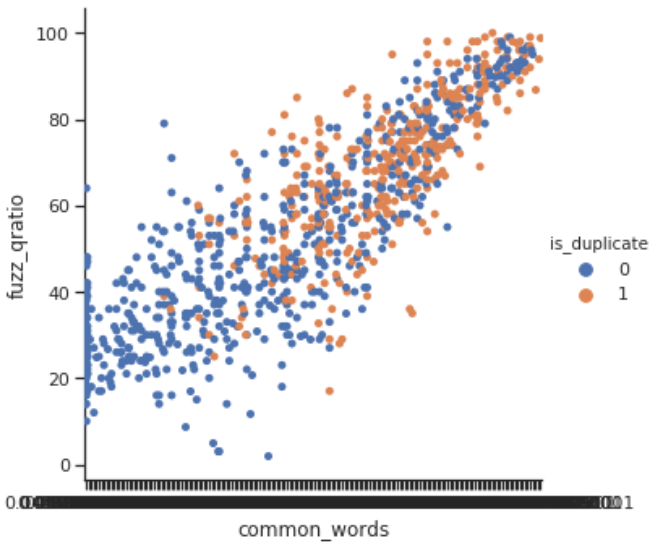

Fig. 7. Fuzzy Ratio

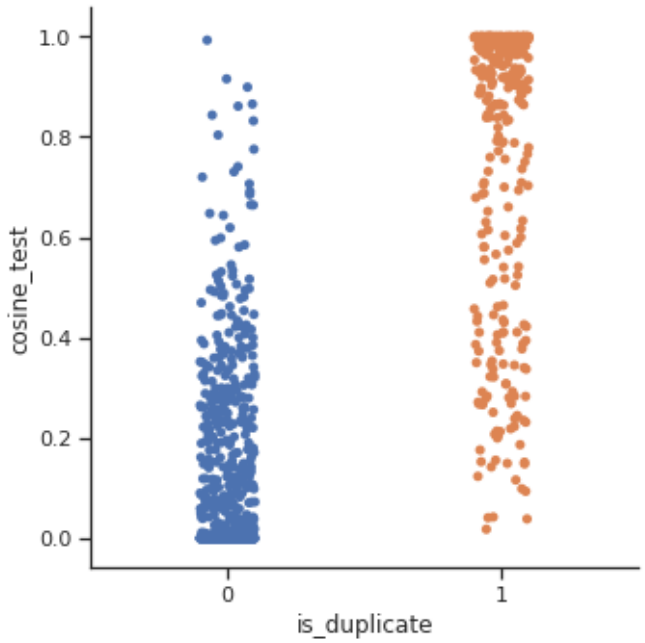

Fig. 8. Cosine Similarity

parameters and feature sets. We analysed that each of these models excelled in a particular type of feature set and some features which were detrimental to a model resulted in being a useful feature for the other.

- **Logistic Regression :** Log loss wrt to logistic regression was on a higher side compaed to other models. Our log-loss fluctuated around 0.37 to 0.32 even with best combinations of features. After analysing the same we uncovered that it was not able to effectively use 2 of our extremely important features i.e. Frequency count and Intersection count.
- **Light GBM Classifier :** Light GBM gave us a log loss of 0.29, this was the fastest model in terms of required time to execute among all the other models. However it could not compare close to our best models wrt to log loss.
- **SVM Classifier :** We used SVM classifier on our dataset and it gave us a log loss of 0.26, however it had quite a major drawback in terms of the time requirements, each trial taking up to 4-5 hrs. We restricted ourselves from this model for future iterations because of the time constraint.
- **Random Forest classifier :** Random forest even though was one of the models which had a lower individual log-loss, made a big difference during ensembling and stacking. (Details following)
- **XG-Boost :** XG Boost was the best model that consistently had a lower value of log loss. Having an average log loss at 0.23, which was close to our final log-loss. Parameter Tuning, and oversampling the XG Boost also improved the log-loss to some extent by shaving off another 0.01.

B. *Stacking*

- **Random Forest classifier and XG-Boost Stacked with XG-Boost :** This is the final model which we chose for our final submission since it gave us a considerable lower log loss. Though Random Forest had a log-loss of 0.36, on stacking it helped us fix few scenarios where our XG boost model used to predict incorrectly because our XG boost had a decrease in performance when the difference between the sentence lengths were too big, though their intent might be the same. The random forest based stacking allowed us to minimize the penalizing effect of being further away from answer by shifting the predicted values more towards the mean

which initially were on the either side in XGB.

- **Random Forest classifier and XG-Boost Stacked with Logistic Regression :** This combination gave us a logloss of 0.29.
- **Logistic Regression and XG-Boost Stacked with XG-Boost:** This model had a good performance, however it suffered same issue as the single XG-Boost model which failed to handle outliers. The log-loss of 0.26 was achieved using this model.
- **Light GBM and XG-Boost Stacked with XG-Boost:** This model gave us a log loss of 0.31 which was not comparable to both models individually.
- **Random Forest classifier and Logistic Regression with XG-Boost :** This model gave us a log loss of 0.28.
- **Stacking three models :** We combined multiple combinations, and also tried 3 layer stacking of Logistic Regression + Light GBM + Random Forst Classifier on top of XG Boost and Random Forest which was further stacked on a single XG Boost, however it led to overfitting of the data.

VI. Post Processing

A lot of different approaches were been tested throughout the course of this project, there were 3 major short comings that took us a while to overcome. After the initial 0.3 logloss mark, where we did text based feature engineering, adding any new feature, based on length, TF-IDF, fuzzy wuzzy etc. ie: string similarity features did not improve our log loss rather was detrimental in it's effect.

The first major breakthrough occured due to graph features, found on one of the stanford project reports. These graph features made a lot of sense, since the arrangement of data internally was based upon graphs. Each question is interlinked as a node with other questions similar to it, as per Quora's actual algorithm.

The final breakthrough that allowed us to break past the 0.2 barrier was post processing, though it did not occur to us that we could work on the results that are provided by our model to further improve our accuracies, we read on multiple discussion forums on how post processing played a key role in improving their scores in various different projects. We tried three approaches,

- **Min-Max :** By inspecting few samples manually, we noticed that if a question provided a value of 0.03 - 0.1, with an extremely high probability we could tell that it would be more likely to be zero, so we would process it as zero and vice versa. However, this method worked well initially but as our models got better, the only situations where our models failed were when a zero was predicted to be closer to one than a zero. This led to the min-max approach to penalize harshly. Since penalty of false prediction is high in log loss evaluation.
- **Balancing :** After analyzing that Random forest though gave us a high log loss on its own, kept predicting closer to ground truth. Example if a question were a duplicate, it would give us a value of over 0.5 most of the times. By analysing this trait about the stability of random forest. We used random forest and our stacked final model, and shifted our undecidable results more towards the extremes. This method kept giving us a small boost, from 0.25 to 0.245, a 0.005 gain, which was significant. However, this was replaced by our next transitivity based feature.
- **Transitivity Method :** Upon reading further and dwelling into how a lot of people approached the problem, it was evident that graph features made a big impact, such as intersection counts and the question frequency etc. For example, if question A is similar to question B, and if question B is similar to question C then we could say that the intent of the question A is same as Question C due to transitivity. because our dataset, contained a lot of repetetive questions asked in different forms of each other, one of the posts online on how this could help to resolve the situations where we've undecidable predictions example, if we ended up near 0.5 which makes the prediction useless, using the post processing technique, we could check if both

questions are linked to a common ancestor. If they're linked to a common ancestor, then they both will be same And we chose the best probability to be set for that question, between the two compared.

VII. Evaluation Metrics

The Quora Question Project uses Log Loss metric, which penalizes harshly for how wrong your answer is, example an answer of 1 will be penalized drastically more compared to an answer of 0.8 if the expected value was 0. So we had to be really careful whenever we wanted to perform any form of weighted averaging. However, our post processing approach took care of many of such question pairs.

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

Where,

- N *No of Rows in Test set*
- M *No of Fault Delivery Classes*
- Y_{ij} *1 if observation belongs to Class j; else 0*
- P_{ij} *Predicted Probability that observation belong to Class j*

Fig. 9. Log-loss Metric

A. Confusion Matrix and AUC

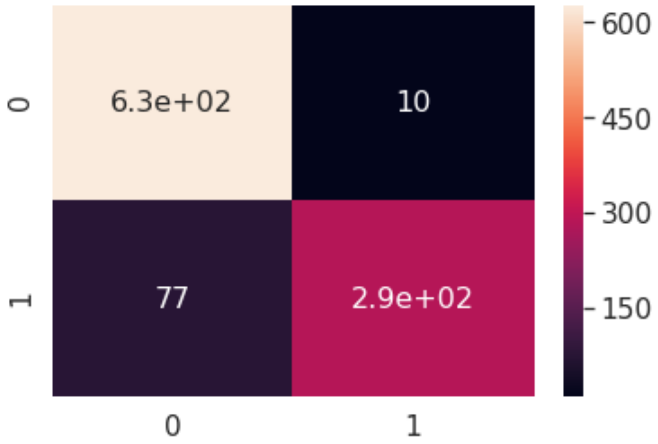

Fig. 10. Confusion Matrix

The Confusion Matrix on around 1000 sample test set tells us that we've a True negative of 630 , and true positive of 290 where as only 77 false negatives, and 10 false positives. Making our accuracy of nearly 91.3

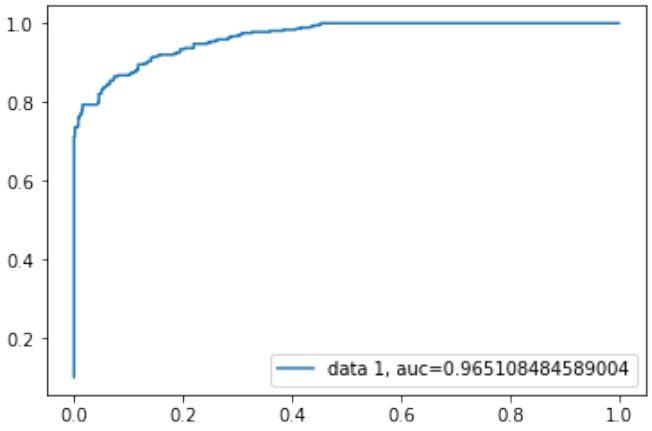

Fig. 11. Area Under The Curve

We obtained a area under the curve of 0.96

VIII. Conclusion

Although our problem required us to use semantic analysis of the question pairs to come to a conclusion on whether the intent is same or not we are convinced that powerful text processing techniques that we applied on our dataset was able to perform at par with many of the commonly used neural network algorithms. Mathematics behind the relation between these questions although could not reveal their individual semantics, was enough to provide us with the relation between these questions thereby being a reliable and quite an accurate solution.

IX. References

- https://towardsdatascience.com/
- https://pdfs.semanticscholar.org/4c19/2b8f45b1e913ee7da32624cd7559eccb0890.pdf
- https://web.stanford.edu/class/cs224n/reports/2761178.pdf
- https://web.stanford.edu/class/cs224n/reports/2759336.pdf