Natural Language Processing

Data Science master's degree

Delivery 1: **Quora question pairs**

Report

Qijun Jin

Challenge

The goal of this competition is to predict which of the provided pairs of questions contains two questions with the same meaning. The ground truth is the set of labels that have been supplied by human experts. The ground truth labels are inherently subjective, as the true meaning of sentences can never be known with certainty. Human labeling is also a 'noisy' process, and reasonable people will disagree. As a result, the ground truth labels on this dataset should be taken to be 'informed' but not 100% accurate and may include incorrect labeling. The labels, overall, are to represent a reasonable consensus, but this may often not be true on a case-by-case basis for individual items in the dataset.

Questions

- What problems/limitations do you think the model has? What type of errors do you get?

  The default model of CountVectorizer has some parameters that can be interesting for our main purpose of extracting features from the doc. We can provide the basic conversion like strip_accents and lowercase, the basic conversion like token_pattern. Moreover, we can also provide specialized methods to filter our doc with a preprocessor and tokenizer.

  By directly applying the default model, we could face the following problems:

  1. Rarely seen words and tags: By nature, the language written by users can contain several words or tags which are not in the standard vocabulary. These undesired words could be HTML tags and mathematical symbols. For most of the queries, these elements are so unusual to appear in the Natural Language, therefore, if we remove these kinds of words, it will help the generalization of the model.
  2. Stopwords: Conversely, the stopwords are the set of very frequent words that can appear in most queries. By its popularity, this kind of word is meaningless to the model as it will worsen the model.
  3. Lemmatization: Generally, words like verbs, adjectives, have their primitive form. This could be a limitation for the model if this type of feature is preserved redundantly on it, meaning that several words have the same meaning.
  4. Stemming: Another type of limitation that could happen is that certain words contain prefixes and suffixes, that can be found in an inflected word. This is also redundant for the model.

5. Negative sense (no implemented): The tokenizer by pattern could have removed the negative sense of the word. The phrase in negative form does not have the same meaning as the one in positive form.
6. Numbers (no implemented): The model by itself cannot differentiate arabic numerals and textual numbers. Therefore, this could pose the problem of duplication of words with the same meaning.

- What type of features can you build to improve the basic naive solution?
  1. Tokenization by pattern: The process of filtering the sentences is important since people could have written some undesired words like HTML tags, and mathematical symbols. If we subtract these rarely seen words, it would be easier for our model to generalize.
  2. Stopwords treatment: In general, sentences have many stopwords which do not provide significant meaning to the sentence. These types of words can be noisy for the generalization of the model.
  3. Lemmatize: Once we have subtracted the stopwords, if we look at the significant words, several words could have the same meaning. So, lemmatization takes into consideration the morphological analysis of the words, which is also beneficial to reduce the dimension of our vocabulary.
  4. Stemming: Since the words are lemmatized, we could still face the problem in which two similar words have the same stem. This would also synthesize the sentences.
  5. Negative treatment (no implemented): This process would be good for the sentence which contains negative tense. If we convert earlier words like “doesn´t” to “does not” and maintain the “not” skipping from the stopwords filtering, we could get the negative tense of sentences that can benefit this type of comparison.
  6. Numerical to textual (no implemented): In the question queries, there are 2 types of numbers: Arabic numerals and textual numbers. In this sense, if we convert the Arabic numerals to textual numbers, it will help the model to learn.

Casting

For each query in the document, we need to cast all the types into a single type to ‘str’ as the feature extraction requires that all the input be the same type, therefore words can be vectorized. The casting method implemented is called *cast_list_as_strings*, it takes as an input a list of strings.

Feature vectorizer

1. SimpleCountVectorizer

   This is a customized CountVectorizer which is provided in the notebook of the second session. It requires a function of doc_clean and tokenizer for this SCV. On the doc_clean side, it is a preprocessor that filters the queries words by removing undesired characters in the queries. On the tokenizer side, it is a method that filters words into more synthesized and shortened primitive words. By this, each query will only contain a significant word. After all, it will convert each query as a vector that has the size of the vocabulary and is represented as sparse.
2. TD-IDF Vectorizer

   Unlike SimpleCountVectorizer, which provides the number of frequencies with respect to index of vocabulary, TD-IDF takes into account the overall documents of weights of words. Therefore, it helps us in dealing with most frequent words which are penalized. In detail, it weights the words counts by a measure of how often they appear in the documents.

Model

1. Logistic Regression

   This is the given classification model that serves as the baseline model. It is a process of modeling the probability of a discrete outcome given an input variable and the model produces a binary outcome.
2. XGBoost

   Nowadays, XGBoost is one of the most widely used supervised Machine Learning algorithms. This is due to its ease of implementation, its good results and it is an optimized distributed gradient boosting that is designed to be highly efficient, flexible, and portable.

Experiments and results

Parameter settings

- SimpleCountVectorizer – LogisticRegression

	ngram	stopwords	lemmatizer	stemmer
V1	(1,1)	None	WordNet	Porter
V2	(1,1)	nltk.stopwords	WordNet	Porter
V3	(1,2)	None	WordNet	Lancaster
V4	(1,2)	None	WordNet	Porter

V5	(1,3)	None	WordNet	Porter

- TFIDFVectorizer – LogisticRegression

	ngram	stopwords	lemmatizer	stemmer
V6	(1,3)	None	WordNet	Porter
V7	(1,3)	nltk.stopwords	WordNet	Porter

- SimpleCountVectorizer – XGBoost

	ngram	stopwords	lemmatizer	stemmer
V8	(1,3)	None	WordNet	Porter
V9	(1,4)	None	WordNet	Porter

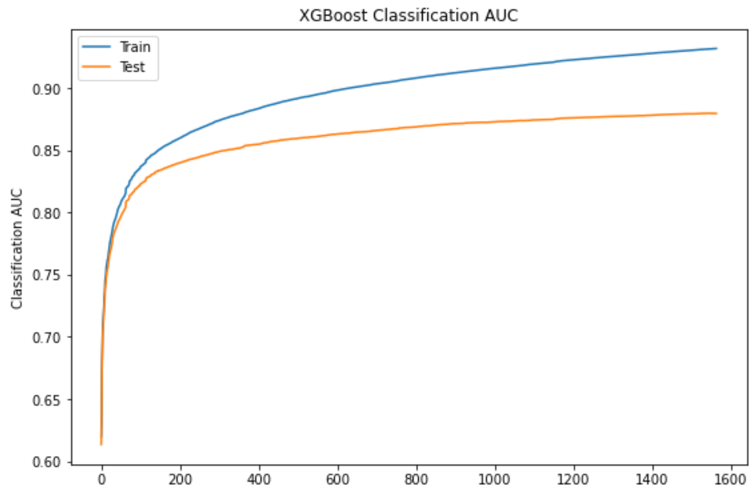
- TFIDFVectorizer – XGBoost

	ngram	stopwords	lemmatizer	stemmer
V10	(1,4)	None	WordNet	Porter

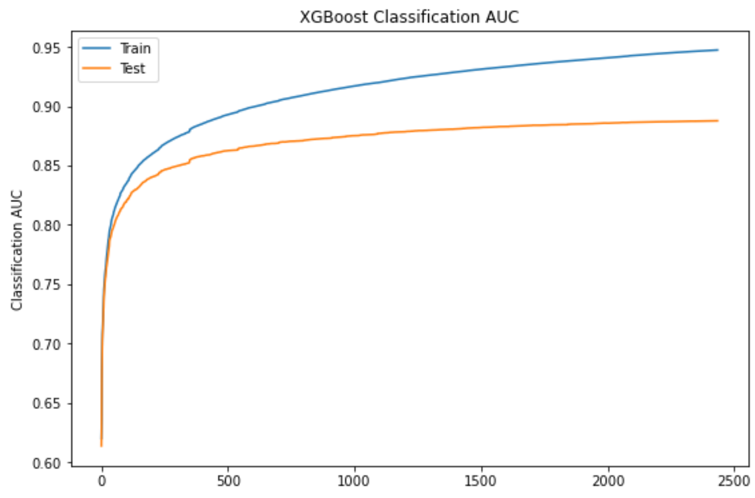
Results (roc_auc_score)

	Train	Validation	Test
V1	0.765	0.715	0.724
V2	0.759	0.705	0.715
V3	0.970	0.767	0.776
V4	0.972	0.771	0.777
V5	0.997	0.787	0.787
V6	0.877	0.764	0.765
V7	0.869	0.749	0.758
V8	0.817	0.768	0.769
V9	0.843	0.780	0.775
V10	0.811	0.744	0,741

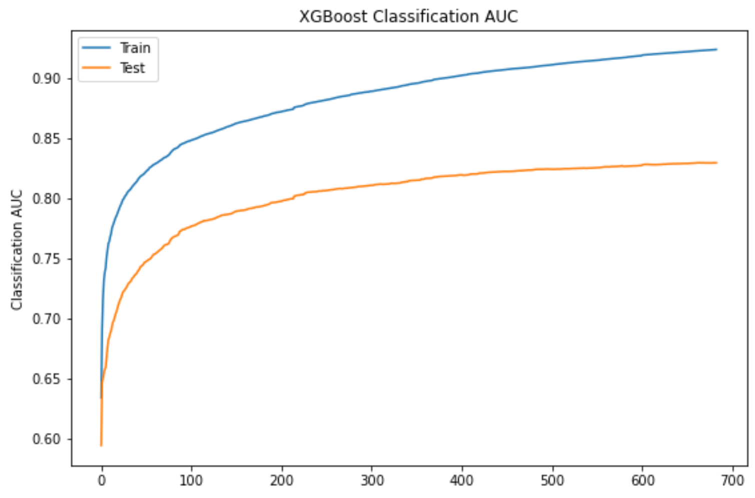
V8 - auc_score



V9 - auc_score



V10 - auc_score



Conclusion

To sum up, this practice is very interesting from the perspective that by using Machine Learning model, we can predict the similarity of two queries of string. As a workflow, there are several procedures that has been done in order to convert the original query into vectorized query. After vectorizing strings, we have used model to learn from the vectorized features. Overall, the highest result obtained is 0.787 from the logistic model. With the XGBoost model, the auc score from the graphics are higher than the one using the predicted values with true labels. It is maybe due to that the state of the best result saved from the fit is not conserved.