Intuitive Features for Identifying Questions of Semantic Coincidence

Tamir Bennatan
tamir.bennatan@mail.mcgill.ca

Jack Wei Xi Luo
jack.luo@mail.mcgill.ca

***Abstract*— In this paper, we address the problem of identifying if pairs of question have the same semantic intent. Previous work on this problem focuses on complex deep learning models, rather than careful feature engineering. In this paper, we propose three sets of features which are designed to capture our intuition of what characterizes semantic similarity between two questions. The first feature set uses Tf-Idf weights to model and compare the principal words in each question. The second set of features are predicted semantic categorizations of each question. Our final feature set is a series of knowledge-based similarity scores, calculated using the WordNet hierarchy. Through careful experimentation we quantify the relative importance of each of these feature sets for detecting questions of duplicate intent. We also a simple heuristic to rank the features we used in terms of predictiveness.**

I. INTRODUCTION

In January 2017, Quora - a popular question and answer website - released a dataset containing pairs of questions, with labels indicating if the questions have the same intent. A Kaggle competition was launched shortly thereafter to crowdsource the best models for predicting whether or not two questions are duplicates. Duplicate question detection is important for Quora, as duplicates make it harder to find the best answer for a particular question, and also limit the outreach of individual contributed answers.

Identifying duplicate questions is a special case of identifying semantic coincidence between short texts. This problem arises in many NLP tasks, including information retrieval (Jurafsky & Martin; 2009), automatic summarization (Lin & Hovy; 2003) and text classification (Li & Roth; 2002).

In this paper, we describe three feature sets designed to capture the semantic relationship between questions. We then evaluate the relative importance of these features for detecting duplicate questions. Each of these feature sets are inspired by different intuitions about what characterizes semantic similarity.

The first feature set is based on Term Frequency-Inverse Document Frequency (Tf-Idf) scores. Tf-Idf scores have been shown to be an effective way to model the relative importance of words within a document (Rajaraman & Ullman; 2011). As such, we propose a set of features which measure the similarity of two questions based on the similarity of the important words in each question.

The second feature set distinguishes between questions based a semantic categorization of their expected answers. Using a dataset of 6000 factoid questions, annotated with categorizations of their answers (Li & Roth; 2002), we fit a series of deep neural network models that predict the semantic category of the answer of a question. We then used these models to predict the answer categories of the questions in the Quora dataset, and used these predictions as features in our models.

The final feature set is a series of knowledge-based semantic similarity scores between paired questions. The similarity of two questions is calculated by composing the semantic similarities of the words in each question (Mihalcea et al; 2006), which can be computed in relation to a semantic network, such as WordNet (Miller; 1995). We also used neural word embeddings to measure word-to-word similarity.

We first trained a classifier on a baseline set of syntactic and neural features. We then augmented this baseline with different subsets of the features mentioned above, and retrained our classifiers with these new features. In this paper, we discuss the usefulness of our proposed feature sets in the duplicate question detection task.

II. RELATED WORK

To achieve high accuracy, many Kaggle competitors incorporated hundreds of features and stacked classifiers when making predictions. Most successful competitors incorporated long short-term memory (LSTM) networks in their submissions. Two LSTM variants proved especially useful: Siamese LSTMs and LSTMs with neural attention.

Siamese LSTMs are composed of two or more LSTM layers which have tied weights. When trained on paired examples, Siamese LSTMs are thought to construct sophisticated representations of semantic relationships between input texts, which makes them effective in semantic similarity and entailment tasks (Mueller et al; 2016). This property is applicable to detecting question deduplication, since at its core this task is one of detecting semantic coincidence between pairs of input sentences.

Neural attention, when coupled with LSTMs, is a mechanism which allows a LSTM to pay selective attention to the outputs of intermediate LSTM units. This technique often improves the performance of LSTMs in Natural Language Inference tasks (Rocktäschel et al.; 2016), and has proven effective on the Quora dataset.

Our work differs from that of the top Kaggle competitors in that our main objective was not to maximize the performance of our models, but rather to craft novel linguistic features and evaluate their predictiveness. Thus, we did not focus on building sophisticated deep learning models, for it is typically difficult to determine the relative importance of different text features to a deep learning neural network. Instead, we focused on using tree-based models to measure the marginal increase in performance of our models when trained on each of our proposed feature sets.

TABLE I

SAMPLE OF QUORA DATASET

QUESTION 1	QUESITON 2	DUPLICATE?
How can I be a good geologist?	What should I do to be a great geologist?	1
What is a web application?	What is a web application framework?	0
How can I access Torbox in India?	How can I access Google.com in India?	0

III. QUORA DATA SET

The dataset released by Quora consists of 404,290 pairs of questions posted to the site. 36.9% of pairs are labeled as duplicates. There are no missing values [Table 1].

Paired questions tend to be similar in that they are similarly worded, or in that they share words of low document frequency.

To gauge the ambiguity of the task and the noisiness of the labels, we asked 10 native English speakers (not the authors) to each classify 40 randomly sampled question pairs as duplicates or not. Of the 400 human responses collected, 81.75% coincided with the provided labels.

IV. METHOD

A. Baseline Features and Models

We built a baseline feature set and model so that we could measure the relative increase in performance that results from incorporating each new feature set.

Our baseline feature set consists of standard syntactic and string edit distance features[1]. We also noticed that many Kaggle competitors used neural word embeddings to construct "sentence vectors" for each question by averaging the embedding vectors of each word in a question. They then used various similarity functions[2] to compute the similarity between the sentence vectors of paired questions as features. We recreated these features using the fastText pre-trained word embeddings (Joulin et al.; 2016). Our initial features are summarized in Table 2.

We then split the dataset into training and test splits, which we kept consistent in all further experiments. We trained logistic regression and gradient boosted trees (XGB) models on the training set using the features described in Table 2, and used 3-fold cross validation to tune hyperparameters. These are our baseline features.

TABLE II

BASELINE FEATURES

Feature category	Feature
Syntactic features	Number of words in each question
	Number of words the two questions have in common
Character features	Number of characters in each question
	Difference in number of characters in each question
Edit distance variations	Partial string matches (with/without stopwords)
	Token-wise string matches
	Type-wise string matches
	Type-sorted string matches
Sentence embedding similarity	Vector similarities of sentence embeddings using Euclidean, Cosine, Cityblock, Bray-Curtis and Jaccard distances

To test whether logistic regression and XGB are adequate models for this task, we also implemented an LSTM classifier, since many Kaggle competitors demonstrated that these models perform well on the Quora dataset. With the use of a development set, we compared several LSTM architectures, and found that the Manhattan Siamese LSTM (MaLSTM) architecture (Mueller et al; 2016) performed best.

The MaLSTM is a Siamese neural network which forms a prediction by taking the Manhattan Similarity of of the vector representations of two inputs where the Manhattan similarity of two vectors v_1 and v_2 is defined:

$$ManhattanSim(v_1, v_2) = \exp(-||v_1 - v_2||_1) \quad (1)$$

We used fastText word embeddings of the words in each question as inputs to the MaLSTM. More details on our chosen architecture can be found in Appendix 1.

We found that the XGB model, when trained on the baseline features, had performance similar to that of the MaLSTM. Thus, we concluded that the XGB model has the capacity to achieve satisfactory results in the question duplicate detection task.

B. Tf-Idf Features

Suppose that we were only allowed to compare one word from each question in a pair to determine if the questions are duplicates, but we had a choice of which words to compare. Which words should we choose? We would reasonably want to choose the most *important* word from each question, or the words which are most particular to each question.

For example, in the fabricated question pair:

1) *Who is the president of the United States?*
2) *What is the highest paying government position?*

We would probably derive more insight into the similarity/dissimilarity of these questions by analyzing the words *president* and *government* than if we compared more commonly occurring words, like *who* and *highest*.

We capture this intuition by creating a set of features that incorporate the Tf-Idf scores of each word. We used Scikit-Learn's `TfidfTransformer` class (Pedregosa et al.; 2011) to compute the Tf-Idf of a word w in document d using the formula:

$$Tf\text{-}Idf(w, d) = Tf(w, d) * Idf(w) \quad (2)$$

Where $Tf(w, d)$ is the number of times word w appers in document d (term frequency), and $Idf(w)$ is defined:

$$Idf(w) = \log \frac{1 + n_d}{1 + df(w)} + 1 \quad (3)$$

[1]Edit distance variations were computed using the Python *fuzzywuzzy* package. More information on these metrics can be found here.

[2]Vector similarities were calculated using the Python *SciPy* package. More information on these metrics can be found here.

Where n_d is the number of documents[3] in the corpus, and $df(w)$ is the number of documents in the corpus that contain the word w.

The Tf-Idf weight of a word in a document will be high if: 1) the word appears many times in that document, and 2) the word has low document frequency. It serves as a natural heuristic for modeling the relative importance of a word within a document. Tf-Idf weights are useful for many NLP tasks; in fact, variations of this scoring scheme are used as term weights in nearly all vector space information retrieval models (Jurafsky & Martin; 2012).

Thus, we used Tf-Idf scores to craft four new features. The first two attempt to measure similarity of the "most important" word of each question in a pair. To do so, we find the word with the highest Tf-Idf weight in each question, then take the fastText word embedding for each of these words, and compute the distance between these vectors. We used Euclidean distance for one feature and Cosine distance for the second.

$$Feature_1 = ||Embed(w_1^*) - Embed(w_2^*)||_2 \quad (4)$$

$$Feature_2 = \frac{\langle Embed(w_1^*), Embed(w_2^*)\rangle}{||Embed(w_1^*)||_2||Embed(w_2^*)||_2} \quad (5)$$

Where

$$w_i^* = \arg\max_{w_i \in d_i}\{Tf\text{-}Idf(w_i, d_i)\} \qquad i \in \{1, 2\}$$

The third and fourth features measure the total Tf-Idf weight of the words two questions have in common, and the total weight of the words that they don't:

$$Feature_3 = \sum_{w \in \{d_1 \cap d_2\}} Tf\text{-}Idf(w, \{d_1 \cap d_2\}) \quad (6)$$

$$Feature_4 = \sum_{w \in \{d_1 \triangle d_2\}} \sum_{i=1}^{2} Tf\text{-}Idf(w, d_i)\mathbb{1}(w \in d_i) \quad (7)$$

C. Question Classifcation Features

IR-based question answering (QA) systems attempt to find short segments of text that answer a user's question by querying a database or searching the web (Jurafsky & Martin; 2017). These systems tend to perform better if they first constrain the answer candidates using a semantic categorization of the expected answer of an input question (Li & Roth; 2002).

For example, when considering the question:

Who is the president of the United States?

A QA system should limit its search to words that correspand to humans, as opposed to considering all noun-phrases.

Using the Text Retrieval Conference (TREC) question dataset (Voorhees; 2002), Roth and Li (2005) developed a hierarchical taxonomy for classifying the answer type of a question; this taxonomy consists of 6 coarse categories, and 50 fine sub-categories. They then annotated the 6000 questions in the TREC dataset using this taxonomy. The coarse answer categories are LOCATION, DESCRIPTION, ENTITY, NUMERIC, HUMAN, and ABBREVIATION, and some examples of fine categories are NUMERIC::Date and NUMERIC::Money. The task of predicting the answer type of a question is broadly termed *Question Classification*.

We hypothesized that knowing the answer types of the questions in the Quora dataset would be useful for the question duplicate detection problem, since two questions are more likely to have duplicate intent if they share the same answer type. Our next feature set tests this hypothesis.

We trained 5 LSTM models on the TREC dataset to predict the coarse categories of each question. These models each have slightly different archetectures, but they all consisted of three layers:

1) Embedding layer using pretrained fastText 300-dimensional embeddings
2) LSTM layer of output dimension 50
3) Output layer (of varying types) of output dimension 6.

The details of the different architectures are in Table 3. Note that we trained these LSTMs on the entire TREC dataset without the use of the development set, so we make no statement about the efficacy of these models for question classification.

TABLE III
ARCHETECTURE DETAILS OF LSTM QUESTION CLASSIFIERS

Model	Output Layer	Dropout	Recurrent Dropout	Embeddings Trainable?	Input Length
LSTM 1	Dense	20%	-	NO	10
LSTM 2	Dense	-	20%	NO	10
LSTM 3	LSTM	-	20%	NO	10
LSTM 4	LSTM	20%	20%	YES	10
LSTM 5	LSTM	-	20%	NO	6

Using these models, we predicted the answer type of the questions in the the Quora dataset. Then, considering each LSTM model separately, we added these predictions to our baseline features, and measured the increase in our baseline XGB model using 3-fold cross validation. We also tried an ensemble approach, where the predicted class of each question is the class that was predicted the most frequently amongst the 5 LSTM models.

We found that the ensembled predictions, as well as a binary feature which indicates if these predictions coincide, increased the performance of our baseline XGB model the most (in terms of reduction of log-loss). Thus, these are the features in the second feature set which we propose:

$$Feature_5 = \{\text{Question 1 Answer Type Prediction}\}$$

$$Feature_6 = \{\text{Question 2 Answer Type Prediction}\}$$

$$Feature_7 = \mathbb{1}(\text{Answer Type Predictions Agree?})$$

D. Sentence Semantic Similarity Scores

Our final feature set uses a framework for measuring the semantic similarity of short texts proposed by Mihalcea et al

[3]*Documents* and *texts* are questions, in this context.

(2006). This framework models the similarity of two texts as a function of the similarities of their component words and the specificity of each word. Word-to-word similarities are measured with respect to a semantic knowledge base (e.g. WordNet.)

Given a word-to-word similarity metric, $Sim(w_1, w_2)$, and two texts[3] T_1, T_2, the semantic similarity score of these two texts is computed by:

1) For each word $w \in T_1$, find the word $w^* \in T_2$ which maximizes $Sim(w, w^*)$.
2) Apply this same process to determine the words in T_1 that have the highest similarity to the words in T_2.
3) Sum up the similarities (weighted by a measure of specificity, such as Idf (3)), and normalize for the length of the sequences.

Note that words in one text are only compared to words in the other text *of the same part of speech*, as many knowledge-based similarity measures cannot be applied across parts of speech (Mihalcea et al; 2006).

$$SimText(T_1, T_2) = \frac{1}{2}\Big(\frac{\sum_{w \in T_1} \max_{w^* \in T_2} Sim(w, w^*) Idf(w)}{\sum_{w \in T_1} Idf(w)} + \frac{\sum_{w \in T_2} \max_{w^* \in T_1} Sim(w, w^*) Idf(w)}{\sum_{w \in T_2} Idf(w)}\Big) \quad (8)$$

This framework guarentees the convenient properties of symmetry: $SimText(T_1, T_2)$ = $SimText(T_2, T_1)$, and that the range of the text similarity scores is the same as that of the word-to-word similarity measure $Sim(w_1, w_2)$ used.

We used the python NLTK package for part-of-speech tagging (Bird et al; 2009). We implemented this framework using three similarity metrics available via NLTK's WordNet interface - path similarity, Leacock-Chodorow similarity (Leacock & Chodorow; 1998) and Wu-Palmer similarity (Wu & Palmer; 1994).

We also implemented a simplified version of this framework by removing the dependency on Idf scores in (8):

$$SimText_{simplified}(T_1, T_2) = \frac{1}{2}\Big(\frac{\sum_{w \in T_1} \max_{w^* \in T_2} Sim(w, w^*)}{|T_1|} + \frac{\sum_{w \in T_2} \max_{w^* \in T_1} Sim(w, w^*)}{|T_2|}\Big) \quad (9)$$

We made this simplification to reduce the correlation between these similarity scores and the Tf-Idf features proposed in section 4.B - which also depend on Idf scores.

We also extended this model by using a word-to-word similarity scores based neural embeddings. Using pre-trained GloVe word embeddings (Pennington et al.; 2014) available through the SpaCy Python package, we used the cosine similarity (5) of the embeddings of two words as the similarity function $Sim(w_1, w_2)$ in (8) and (9).

Using 3-fold cross validation, we added different subsets of these text-to-text similarity scores to our baseline feature set, and studied resulting boost in our baseline XGB model performance. Since the resulting text-to-text scores are highly correlated when using path similarity, Leacock-Chodorow similarity and Wu-Palmer similarity, we only tested one of these scoring functions at a time.

We found that including two semantic similarity scores, calculated using the simplified scheme (9) and using Leacock-Chodorow distance and embedding Cosine distance as $Sim(w_1, w_2)$, resulted in the largest increase in performance in our baseline model. These two similarity scores define the final feature set we propose.

$$Feature_8 = \{SimText_{Simplified}(T_1, T_2) \quad (10)$$
where $Sim(w_1, w_2) \equiv$ Leacock-Chodorow Similarity$\}$

$$Feature_9 = \{SimText_{Simplified}(T_1, T_2) \quad (11)$$
where $Sim(w_1, w_2) \equiv$ Cosine Distance (GloVe Embeddings)$\}$

V. RESULTS

To measure the importance of each of these feature sets, we trained logistic regression and XGB models on all the features described above. We then removed different subsets of the Tf-Idf, question classification and semantic scoring feature sets, and re-tuned the models' hyperparameters using 3-fold cross validation. We then used these models to predict the test-set labels. The prediction accuracy for the resulting models are summarized in Table 4, as well as those of our baseline classifiers described in section 4.A. We limit our analysis to the XGB model, as this performed much better than logistic regression.

Including all three feature sets results in the best performance, with a test set accuracy of 79.25% - 4.69% better than the baseline XGB model. Amongst the three feature sets we propose, removing the Tf-Idf features (4.B) results in the largest decrease in prediction accuracy relative to the full model - with a 3.01% decrease. Removing the question classification features from 4.C results in the smallest decrease in prediction accuracy, indicating that these features are the least meaningful of those we propose.

When using ensembled tree models such as XGB, one can use the number of times a variable is split upon as a measure of that variable's importance. This measure is called the *F-Score*. By plotting the F-Scores of each feature, we can gain insight into the importance of each variable used in a model, relative to the other variables present (Gareth et al; 2017).

A variable importance chart of the full model corroborates the result that the Tf-Idf features are the most meaningful, as these features are amongst those with the highest F-Scores [figure 1]. The semantic similarity scores also have high F-Scores. Interestingly, when the Tf-Idf features are removed, the sematnic similarity scores become much more important relative to the remaining features, evident in Figure 2. This suggests that the Tf-Idf features and semantic similarity scores explain some of the same variance, and once the Tf-Idf

TABLE IV

TEST SET ACCURACY SCORES, WITH DIFFERENT FEATURE SETS OMITTED

XGB AND LOGISTIC REGRESSION, WITH FEATURE SETS OMITTED								
Feature sets ommited	∅	{1}	{2}	{3}	{1,2}	{1,3}	{2,3}	{1,2,3}
XGB Test Set Accuracy	**.7925**	.7624 (.0301)	.7861 (.0064)	.7825 (.01)	.7585 (.034)	.7504 (.0421)	.7794 (.0131)	.7456 (.0469)
Logistic Regression Test Set Accuracy	**.6962**	.6838 (.0124)	.6951 (.0011)	.6837 (.0125)	.6836 (.0126)	.6743 (.0219)	.6827 (.0125)	.6728 (.0234)
BASELINE MODELS								
Manhattan LSTM	.7588	Human (400 Responses)			.8175	Random Guessing		.5015

Here, {1} are the Tf-Idf features from 4.B, {2} are the question classification features from 4.C, and {3} are the sematantic similarity scores from 4.D. Cells contain each model's prediction accuracy, as well as the difference between prediction accuracy and the best model's prediction accuracy.

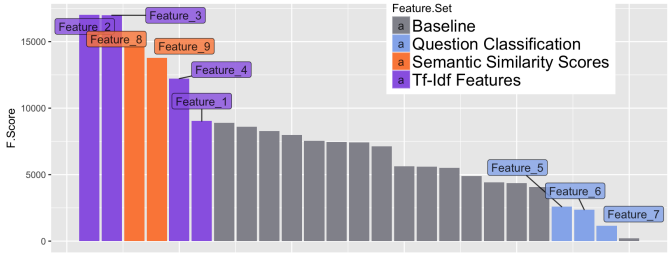

Fig. 1. Variable importance chart of model trained on all features.

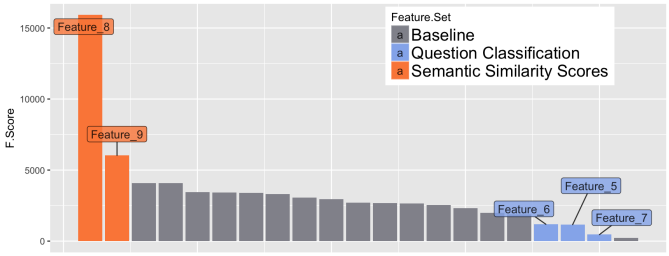

Fig. 2. Variable importance chart of model that excludes the Tf-Idf features.

features are removed the model must rely more heavily on the semantic similarity scores. Indeed, the features in these to feature sets are strongly correlated; the correlation between Feature 9 and Features 6 and 7 are 68.28% and -73.91%, respectively.

The variable importance charts also suggest that the question classification features are not meaningful. In fact, these three features are amongst four features with the lowest F-Score in the complete model.

VI. DISCUSSION AND CONCLUSIONS

Using the three sets of features we describe in this paper, we were able to improve the prediciton accuracy of our baseline model from 74.56% to 79.25%. Of the three feature sets, those computed using Tf-Idf scores were the most impactful. This confirms our hypothesis that measures of word importance and specificity are useful in comparing the semantic similarity between two questions.

We also applied and extended a framework for modeling semantic similarities between short texts using knowledge-based word similarity scores, proposed by Mihalcea et al (2006). We found that these scores were useful in our classification task. These features were highly correlated with those calculated using Tf-Idf scores, however, and so it may be that these scores are capturing the specificity of the words in each question, rather than the semantic similarity of these words.

Finally, we found that our answer type predictions were not useful for for detecting duplicate questions. We suspect that this is because we used overly complex models to predict the answer type of a question. The TREC dataset has only 6,000 observations, but the number of parameters in our LSTM classifiers ranged from 70,506 to 28,141,268. Though we can't know for sure, it's probable that these classifiers were overfit to the TREC dataset, and did not generalize to the Quora dataset.

In future work, we would try using simpler models for predicting a question's answer type - such as Naïve Bayes or a Support Vector Machine. We could also try to incoroprate predictions for the fine answer categories in the annotated TREC dataset, as apposed to the coarse categories we used.

We also think that an interesting extension to this work would be to extract features that concern the grammatical functions of the words in each question. For example, one could construct a dependency parse tree for each question, and compare the subjects direct objects, head-words of each question.

VII. STATEMENT OF CONTRIBUTION

Tamir engineered the features described, worked on the classifiers built, and conducted model experiments. Jack contributed to the classifiers and paper, and experimented with advanced techniques for feature extraction, such as named entity recognition and headword extraction.

APPENDIX

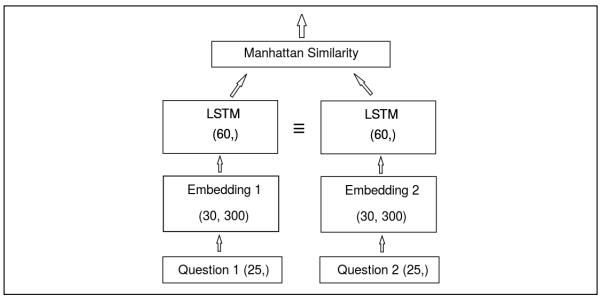

Fig. 3. Manhattan Siamese LSTM archetecture, annotated with output dimension at each layer. Weights of each LSTM layer are tied (Siamese).

REFERENCES

[1] Daniel Jurafsky and James H. Martin. 2009. Speech and Language Processing (2nd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[2] Daniel Jurafsky and James H. Martin. 2017 Speech and Language Processing. (3rd Edition Draft)

[3] Lin, C., and Hovy, E. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In Proceedings of Human Language Technology Conference .

[4] Xin Li, Dan Roth, Learning Question Classifiers. COLING'02, Aug., 2002.

[5] Rajaraman, A.; Ullman, J.D. (2011). "Data Mining". Mining of Massive Datasets. pp. 1–17.

[6] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In AAAI'06, July 2006.

[7] George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

[8] Jonas Mueller and Aditya Thyagarajan. Siamese Recurrent Architectures for Learning Sentence Similarity. In AAAI, 2016

[9] Rocktäschel, Grefenstette, Hermann, Kočiský and Blunsom. Reasoning about Entailment with Neural Attention. in: International Conference on Learning Representations (ICLR). 2016

[10] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of Tricks for Efficient Text Classification

[11] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[12] Xin Li and Dan Roth, (2005). Learning question classifiers: The role of semantic information. Journal of Natural Language Engineering, 11(4).

[13] Bird, Steven, Edward Loper and Ewan Klein (2009). Natural Language Processing with Python. O'Reilly Media Inc.

[14] Wu, Z., and Palmer, M. 1994. Verb semantics and lexical selection. In Proceedings of the Annual Meeting of the Association for Computational Linguistics.

[15] Leacock, C., and Chodorow, M. 1998. Combining local context and WordNet sense similarity for word sense identification. In WordNet, An Electronic Lexical Database. The MIT Press.

[16] James, Gareth, et al. "Bagging, Random Forests, Boosting." An Introduction to Statistical Learning: with Applications in R, Springer, 2017.