Preparation of Papers for IEEE Sponsored Conferences & Symposia*

Tamir Bennatan
tamir.bennatan@mail.mcgill.ca

Pradeep Misra** pmisra@cs.wright.edu

Abstract— **This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.This is a bunch of filler. It is here so that I don't screw myself over while formatting.**

I. INTRODUCTION

In January of 2017, Quora a popular question and answer website, released a dataset consisting of pairs of questions from their site, and labels indicating if the questions have the same intent. Shortly thereafter, they used this dataset to start a Kaggle competition, offering $25,000 to the competitors who build the best models for predicting whether a pair of questions are duplicates, in terms of log-loss error.

The ability to identify duplicate questions is important for question-answer sites like Quora; duplicate questions make it harder to find the best answer for a particular question, and limit the outreach of each individual answer contributed to the site.

Key to the problem of duplicate question detection is the ability to model semantic similarities between pairs of text. This is an important component of many Natural Language Processing (NLP) applications, 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 the duplicate question detection task. Each of these feature sets are inspired by different intuitions about what characterizes semantic similarity, and are engineered using techniques used in wide ranging NLP tasks.

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 attempts to distinguish between questions based on broad categorizations of their expected answer type - a process called question classification. Using a dataset of 5500 factoid questions, annotated with their answer type (Lin & Roth; 2002), we fit a series of deep neural network models that predict the answer type of a question. We then used the predictions of these models on the Quora dataset as features for the question deduplication task.

The final feature set uses a framework for creating knowledge-based semantic similarity scores between sentences (Mihalcea et al; 2006). These scores are calculated using semantic similarity metrics of the words in each sentence, where these metrics are defined in relation to a semantic network, such as WordNet (Miller; 1995). We implement and test the utility of these scores, as well as extend this framework by incorporating neural word embeddings.

We first trained a baseline classifier on standard syntactic and neural features. We then augmented these baseline features with different subsets of the features mentioned above, and re-trained 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 competitors in the Kaggle competition incorporated hundreds of features and stacked many classifiers when making predictions. Most successful competitors incorporated Long Short Term Memory Recurrent Neural Networks (LSTM) 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 have been shown to be effective in semantic similarity and entailment tasks, for they construct sophisticated representations of semantic relationships between input examples (Mueller et al; 2016). This property is applicable to detecting question deduplication, since at its core the duplicate question detection task is one 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 has proven to improve the performance of LSTMs in various Natural Language Inference tasks (Rocktschel et al.; 2016).

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
Why do girls want to be friends with the guy they reject?	How do guys feel after rejecting a girl?	0
How can I access Torbox in India?	How can I access Google.com in India?	0

Our work differs from that of the top 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 using a deep learning model. Instead, we drew inspiration from techniques used in other NLP tasks build features that capture our intuition about what it means for two texts to be semantically similar.

III. QUORA DATA SET

The dataset released by Quora consists of 404290 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 guage the ambiguity of the problem and the noisiness of the labels, we asked 8 native English speakers (not the authors) to each classify 40 randomly sampled question pairs as duplicates or not. Of the 320 human responses collected, 80.93% coincided with the provided labels.

IV. METHOD

A. Baseline Features and Models

To determine if a new feature is useful, we needed a baseline feature set and model, so that we could measure the increase in performance that results from incorporating each new feature.

Thus, we started by creating a baseline feature set. These were typical syntactic and string distance features. 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 the question. They then used various similarity functions to use 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

[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.

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 character in each question
	Number of character in each question
Edit distance variations [1]	Partial string matches (with/without stopwords)
	Token-wise string matches
	Type-wise string matches
	Type-sorted string matches
Sentence embedding similarity [2]	Vector similarities of sentence embeddings using Euclidean, Cosine, Cityblock, Bray-Curtis and Jaccard distances

trained Logistic Regression and Extremely Boosted Decision 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 models.

To test whether logistic regression and XGB models are adequate choices 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 the Sequence-to-Sequence LSTM the Manhattan Siamese LSTM (MaLSTM) archetectures, and found that the MaLSTM performed better. (Mueller et al; 2016).

The MaLSTM is a siamese neural network which emits a prediction by taking the Manhattan similarity of of the vector representations of two inputs (Mueller et al; 2016), where the Manhattan similarity of two vectors v_1 and v_2 is defined:

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

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

We found that the XGB model, when trained on the baseline featurees, had performance similar to that of the MaLSTM. Thus, we concluded that the XGB model has the capacity to adequately perform 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 two questions by analyzing the words *president* and *government*, instead of 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-Learns `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:

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

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 to as term weights in nearly all vector space information retrieval models (Jurafsky & Martin; 2012).

The Tf-Idf score 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 extract 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, extract fastText word embedding vector of each of these words, and compute the distance between these vectors. We used cosine distance for one feature and embedding distance for the second. I.e:

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

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

Where

$$(w_1^*, w_2^*) = \underset{w_1 \in d_1, w_2 \in d_2}{\arg\max} \big(Tf\text{-}Idf(w_1,d_1), Tf\text{-}Idf(w_2,d_2)\big)$$

The third and fourth features measure the total Tf-Idf weight of the words the questions in a pair share, and the total weight of the words that they dont share:

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

[3]Documents correspond with questions, in this context.

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

C. Question Classifcation Features

IR-based question answering (QA) systems attempt to find short segments of text that adequately answer a users 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 is likely to return noisy responses if it considers every noun-phrase in a knowledge base. Instead, it should only consider noun-phrases correspond to humans.

Using the Text Retrieval Conference (TREC) question dataset (Voorhees; 2002), Dan Roth and Li Xin (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 built 5 LSTM models, and trained them on the TREC dataset to predict the coarse categories of each question. The models differed in their use of dropout between layers, recurrent droput within LSTM cells, the type of the output layer they used their trainable parameters, 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 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 performance of these models in the question classification problem.

We used the models to predict 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 was the class that was predicted the most frequently amongst the 5 LSTM models.

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

We found that the ensemble predictions of each question in a pair, 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). These are the features in the second feature set which we studied.

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

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

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

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 (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 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 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 has 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 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 NLTKs WordNet interface - path similarity, Leacock-Chodorow similarity (Leacock & Chodorow; 1998) and Wu-Palmer similarity (Wu & Palmer; 1994).

We implemented one implication of this framework. First, we removed the dependency on Idf scores in (4), and normalized by taking the average of the word-to-word similarities:

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

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 score based neural embeddings. Using pre-trained GloVe word embeddings (Pennington et al.; 2014) available through the SpaCy python library, 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 text-to-text scores are highly correlated when using path similarity, Leacock-Chodorow similarity and Wu-Palmer similarity as $Sim(w_1, w_2)$, we only tested one of these scores at a time.

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

$$Feature_{10} = \{SimText_{Simplified}(Q_1, Q_2) \text{ where } Sim(w_1, w_2) \equiv \text{Leacock-Chodorow Similarity}\} \quad (10)$$

$$Feature_{11} = \{SimText_{Simplified}(Q_1, Q_2) \text{ where } Sim(w_1, w_2) \equiv \text{Cosine Distance (GloVe Embeddings)}\} \quad (11)$$

E. Some Common Mistakes

- The word data is plural, not singular.
- The subscript for the permeability of vacuum ?0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter o.
- In American English, commas, semi-/colons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear

outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an inset, not an insert. The word alternatively is preferred to the word alternately (unless you really mean something that alternates).
- Do not use the word essentially to mean approximately or effectively.
- In your paper title, if the words that uses can accurately replace the word using, capitalize the u; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones affect and effect, complement and compliment, discreet and discrete, principal and principle.
- Do not confuse imply and infer.
- The prefix non is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the et in the Latin abbreviation et al..
- The abbreviation i.e. means that is, and the abbreviation e.g. means for example.

V. USING THE TEMPLATE

Use this sample document as your LaTeX source file to create your document. Save this file as **root.tex**. You have to make sure to use the cls file that came with this distribution. If you use a different style file, you cannot expect to get required margins. Note also that when you are creating your out PDF file, the source file is only part of the equation. *Your TEX → PDF filter determines the output file size. Even if you make all the specifications to output a letter file in the source - if you filter is set to produce A4, you will only get A4 output.*

It is impossible to account for all possible situation, one would encounter using TEX. If you are using multiple TEX files you must make sure that the "MAIN" source file is called root.tex - this is particularly important if your conference is using PaperPlaza's built in TEX to PDF conversion tool.

A. Headings, etc

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named Heading 1, Heading 2, Heading 3, and Heading 4 are prescribed.

B. Figures and Tables

Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation Fig. 1, even at the beginning of a sentence.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity Magnetization, or Magnetization, M, not just M. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write Magnetization (A/m) or Magnetization A[m(1)], not just A/m. Do not label axes with a ratio of quantities and units. For example, write Temperature (K), not Temperature/K.

VI. CONCLUSIONS

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

APPENDIX

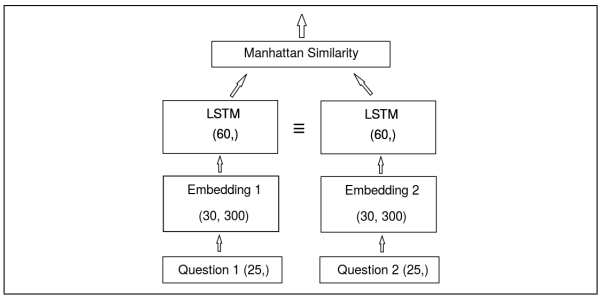

Fig. 1. Manhattan Siamese LSTM archetecture, annotated with output dimension at each layer. Weights of each LSTM layer are tied (Siamese).

STATEMENT OF CONTRIBUTION

The preferred spelling of the word acknowledgment in America is without an e after the g. Avoid the stilted expression, One of us (R. B. G.) thanks . . . Instead, try R. B. G. thanks. Put sponsor acknowledgments in the unnumbered footnote on the first page.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

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. 117.
[6] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In AAAI06, 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] Rocktschel, Grefenstette, Hermann, Koisk 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]
[14]
[15]