ENGINEERING AT QUORA

Semantic Question Matching with Deep Learning

Nikhil Dandekar

Authors: Lili Jiang, Shuo Chang, and Nikhil Dandekar

In order to build a high-quality knowledge base, it's important that we ensure each unique question exists on Quora only once. Writers shouldn't have to write the same answer to multiple versions of the same question, and readers should be able to find a single canonical page with the question they're looking for. For example, we'd consider questions like “What are the best ways to lose weight?”, “How can a person reduce weight?”, and “What are effective weight loss plans?” to be *duplicate questions* because they all have the same intent. To prevent duplicate questions from existing on Quora, we've developed machine learning and natural language processing systems to automatically identify when questions with the same intent have been asked multiple times.

We recently released a public dataset of duplicate questions that can be used to train duplicate question detection models like the one we use at Quora. In this post, we'll give you a sense of what's possible with our duplicate question dataset by outlining a few deep learning explorations we pursued in a recent hack week.

We're also excited to announce a **meetup event for NLP and Machine Learning enthusiasts** that we'll be hosting at the Quora office in Mountain View, CA on the evening of **February 27**. We have a couple of exciting speakers lined up: Ben Hamner, co-founder and CTO of Kaggle will talk about “Kaggle Competitions and Reproducible Machine Learning”, and Xavier Amatriain, VP of Engineering at Quora, will be giving a talk entitled “Machine Learning and NLP at Quora”. If you're interested in attending, please apply to join here.

Problem Definition

More formally, the duplicate detection problem can be defined as follows: given a pair of questions *q1* and *q2*, train a model that learns the function:

$f(q1, q2) \rightarrow 0 \text{ or } 1$

where *1* represents that *q1* and *q2* have the same intent and *0* otherwise.

A simple duplicate detection approach is a word-based comparison. For example, we can use standard information retrieval measures like tfidf - or BM25 to find the word-based similarity between the two questions, then classify question pairs with similarity scores above a certain threshold as duplicates. But, if we want to detect duplicates at a more intent-based, semantic level, we need more intelligent approaches than these simple measures. In fact, there is active ongoing research in the Natural Language Processing community on this very problem [1-3].

Our Approaches

Our current production model for solving this problem is a random forest model with tens of handcrafted features, including the cosine similarity of the average of the word2vec embeddings of tokens, the number of common words, the number of common topics labeled on the questions, and the part-of-speech tags of the words.

experimented with three end-to-end deep learning solutions to the duplicate detection problem. Because each of these approaches was an end-to-end deep learning solution, we were able to save time by avoiding iterative, and often computationally complex feature engineering.

Our first approach was an in-house deep architecture that used Recurrent Neural Networks (RNNs). More specifically, we used the Long Short Term Memory network (LSTM) variant of RNNs, which are better at capturing long-term dependencies. We trained our own word embeddings using Quora's text corpus, combined them to generate question embeddings for the two questions, and then fed those question embeddings into a representation layer. We then concatenated the two vector representation outputs from the representation layers and fed the concatenated vector into a dense layer to produce the final classification result. Below is a visual representation of our first approach:

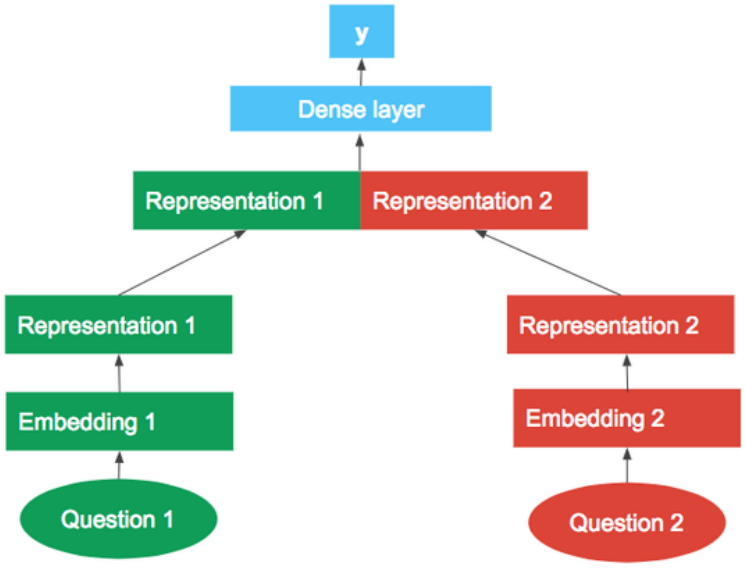

Figure 1: Architecture of approach 1, “LSTM with concatenation”

In our second exploration, we implemented the approach of Tai, Socher empirically-motivated handcrafted features: (1) the distance, calculated as the sum of squares of the difference of the two vectors, and (2) the “angle”, calculated as an element-wise multiplication of the two vector representations (denoted Θ). A neural network using the distance and angle as the two input neurons was then stacked on top, as shown below:two, and Manning [1] (footnote a). Similar to the previous approach, this too was an LSTM network, but rather than concatenating information from the two vector representations, this approach used

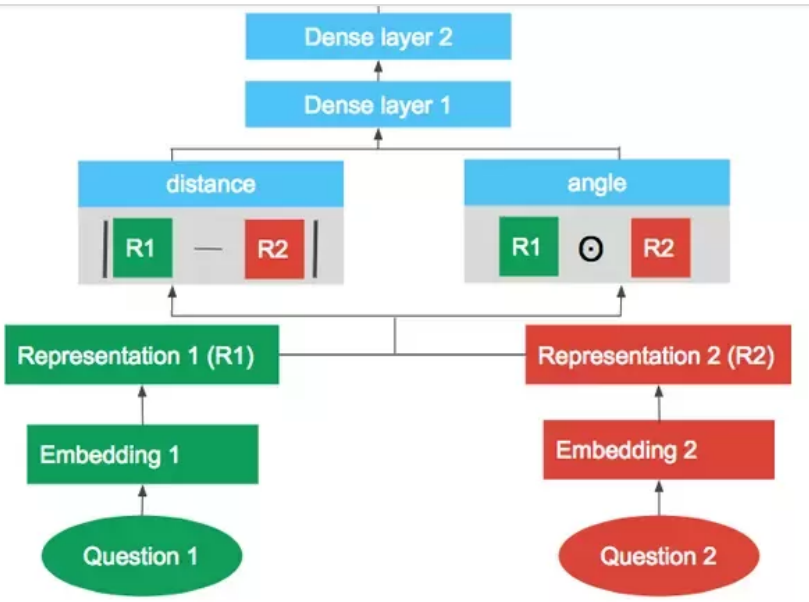

Figure 2: Architecture of approach 2, "LSTM with distance and angle"

Finally, we tried an attention-based approach from Google Research [4] that combined neural network attention with token alignment, commonly used in machine translation. The most prominent advantage of this approach, relative to other attention-based approaches, was the small number of parameters. Similar to the previous two approaches, this model represents each token from the question with a word embedding. The process, shown in Figure 3, trained an attention model (soft alignment) for all pairs of words in the two questions, compared aligned phrases, and finally aggregated comparisons to get classification result.

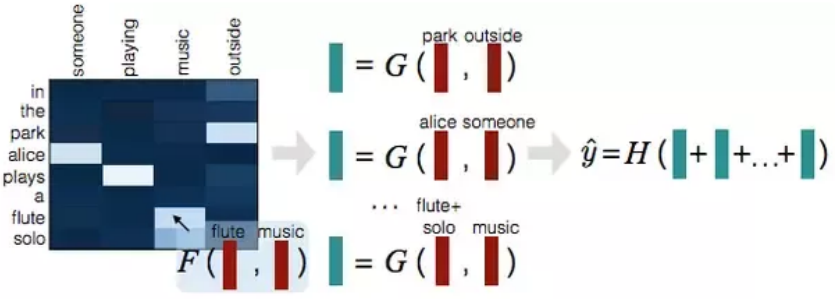

Figure 3: Overview of approach 3, "Decomposable attention" (from [4])

Here is how these three approaches compared to each other on our test data:

	LSTM with concatenation	LSTM with distance and angle	Decomposable attention
Accuracy	0.87	0.87	0.86
Precision	0.88	0.83	0.81
Recall	0.86	0.94	0.95
f1 score	0.87	0.88	0.87

In the future, we plan to experiment with more complex, deeper architectures as well as ensembles of different approaches, but we hope these examples illustrate several different approaches to the duplicate question detection problem.

We're excited to connect with fellow members of the NLP and ML communities at our meetup on February 27th. You can learn more and request an invite here.

Acknowledgements

This exploration was a collaboration between members of our engineering and data science teams: Hilfi Alkaff, Shuo Chang, Kornél Csernai, Nikhil Dandekar, and Lili Jiang.

References

[1] Juri Ganitkevitch, Benjamin Van Durme, Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*.
[2] Lushan Han, Abhay Kashyap, Tim Finin, James May- field, and Jonathan Weese. 2013. UMBC EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*.
[3] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.
[4] Ankur P Parikh, Oscar Täckström, Dipanjan Das, Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference, arXiv 2016

Footnote

a. The original paper proposes a tree-LSTM approach, where it parses the structure of input sentences based on POS tags. In practice, because of speed reasons, we didn't use the tree-LSTM approach but instead borrowed their architecture for a simple LSTM model.