Draft: Question-Question Mapping

Eric Gibbons

1 Review

Just as a quick review, my project is to develop a method to sentence to sentences. The idea behind this is approach is that if we can match some given input sentence to one of our 1.6 million questions we already have a correct answer for that question. Thus we are spared the tedium (and very difficult task) of a question-answer situation.

2 Nuances

This approach is fairly nuanced and rather challenging. Computing sentence similarity would require matching the grammatical model of the sentence and equivalent structures. For example, suppose we have

"He walked to the store yesterday"

This is equivalent to

"Yesterday, he walked to the store"

though the structure is slightly different.

3 Method

The method I have been trying to trying to implement uses the skip-gram `word2vec` method. The details of this algorithm are found here[1]. A lower level description of word embedding can be found in the CS224D course notes and videos as well. The basic idea here is that we are able to place our dictionary of words in our corpus (all of the unique words in the body of text we are working with) in some N-dimensional space by assigning them some vector based on the relative context in each occurance of the word in the text. Thus, given a dictionary of M words, we can assign word i to some vector $x_i \in \mathbb{R}^N$ where N is the length of the vector. If we could visualize this (and you can use algorithms to reduce the dimensionality of the vector), we would see that each related words might cluster together and so forth. Again, this is a very simplistic explanation, but that is the gist of the method.

The question now is how to extend this to sentences. The simplest, and the one I am using, allows us to find the the word vectors x_i for each sentence. We can do a point-wise average of the vectors to create a sentence vector v_j.

$$v_j = \frac{1}{L}\sum_{i=1}^{L} x_i$$

Now, suppose the user inputs a sentence. We can then convert this sentence to a sentence vector $\hat{v}$. We can then do a comparison against all of the other sentence vectors from the known questions. An appropriate comparison could be the ℓ_2-norm, but perhaps other checks could also be used. This would take the form

$$j = \underset{j}{\text{argmin}} \left\| \hat{v} - v_j \right\|_2$$

where j is the index corresponding to the closest matching sentence in the known question-answer pairs.

4 Progress

This is coded up already. I cannot speak as to how well it works. The bottlneck is the computation. I can generate the word embeddings easily. Most of the heavy lifting there is done on the GPUs via TensorFlow. However, the averaging and matching were both coded in python/numpy, which are slow for these choices.

To give some numbers, our dictionary size is the 50,000 most common words (after the stop words are left out). We have around 1.6 million pairs with each sentence being around, say, 15 words. Each vector is $N = 128$ long. That is a lot of matching and averaging to create the sentence embeddings. Past that, we have to compare a test sentence to each one of the 1.6 million sentences. That is unreasonably slow.

My goals in the future are:

- My goal next month is to speed this up. A lot of speed can be gained by doing this in a lower language such as C++ and wrapping it in python/numpy, which is essentially what Caffe and TensorFlow do. That should make the process more manageable.
- I will also have to spend time trying to tune the word2vec to get proper embedding.
- I also want to see if running an automatic spell-checker would improve these embeddings. These were scraped from the internet the spelling is atrocious, and since there could be several spellings from many words this could throw off the embedding. I am unsure what performance gain this would give us, but it would help.

To the last point, if we could get the real clinical QA data that would make the biggest difference. Reading some of the crawled QA data, it is easy to see that the use of the English language is horrendous with bad spelling, grammar, and abuse of abbreviations. Clinical data would (hopefully) be more uniform and consequently would lead to more accurate responses.

References

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.