Finishing Sentences

Andrew Savage and Bhushan Suwal

1 Project Overview

Many of the technologies that exist in everyday life use techniques to assist in sentence creation: messenger applications often have suggestions for the next word to use, Gmail has both automatic response suggestions and auto-complete for email writing, and Google search has plenty of suggestions for queries when you begin typing. These are only a few examples of a field of very many—sentence prediction is clearly a large area of work.

Our project will take an approach to the work of sentence completion with natural language processing by looking at sentence completion under certain contexts.

Partial sentence completion is certainly a hard problem alone. Partial sentences can vary widely (of course) and suggest very different kinds of completions based on itself. Some partial sentences can have a huge number of possibilities for completion ("This is a...") while some have very few possibilities ("Hi, my name is..."). This makes the process for completing a sentence very inconsistent.

Currently, most of the work done with sentence completion is in syntax formation. The organization of words in a sentence can be hard due to the fact that in different sentences words can be organized in very different ways.

Adding on context to the problem of sentence completion drastically narrows down the space of reasonable possibilities for the end of the sentence. Taking the previous open ended example of "This is a..." we narrow the range of feasible possibilities if we instead have two sentences, the first providing context for the second: "The weather is really lovely! This is a..." gives the implication that the second sentence should conclude something about the weather, as in "This is a beautiful day."

- Aim 1: Use partial sentences to attain complete sentences that are at least decently syntactically correct.
- Aim 2: Given a context with the partial sentence, attain a complete sentence which fits under that context.

2 Background and Related Work

There has been significant work done on the topic of sentence completion under a certain context, which Google's blog [3] explains nicely.

Because we also use neural networks in our example (even though the actual implementation is built in) we need to do some feature selection. Sutton and Barto [2] give a description of neural networks, and we have another source for feature selection and tuning.

3 Problem Formulation and Technical Approach

Given a partial sentence, learning how to pick the next word in the context of reinforcement learning can be a hard problem to formulate: the previous sequence of words should of course give some information as to what the state of the agent is in, but it is not clear how to represent that state. Similarly, how do we define how "good" a complete sentence is? A sentence may be grammatically correct but completely nonsensical, as in often the case with natural language processing applications. So how do we reward the agent?

Word vectors appear to be one solution to these problems. Word vectors are a way of quantifying words: they keep track of the syntax of the word within a sentence, to some (variable) degree. More specifically, for each word in a piece of text, the vectors tracks some number of previous and some number of future words within a sentence. The word vectors track when each word appears and does not appear within a given specified syntax, tracking which syntaxes are usual for that word and which are unusual or do not appear at all. It is also common to perform principal component analysis on the word vectors and prune the resulting space based on the steps between eigenvalues produced, so one can operate in a smaller vector space (generally by an order of magnitude) than one would otherwise.

We will use some combination of word vectors from the context clause and the partial previous sentence to form a point in vector space (testing different functions to combine the word vectors to see what works the best). We will also pull sentences from some English sentence database to incorporate a reward function: this reward will be determined by the distance that the completed sentence produced by the agent has from its nearest neighbor in the existing word space consisting of sentences from the database, that is the closer the agent's produced sentence to the sentence in the word space, the higher its score. We will also use context as a reward metric: if the nearest neighbor to the agent's sentence has a similar context to the context given to the agent, it will result in a higher reward.

Actions are simpler—the agent just needs to pick which word will be next. This will be done by feeding the vector consisting of the context and the existing partial sentence into a recurrent neural network, likely as described in [1]. After a word is fed in, the output will be a word from the English dictionary (though this will likely be pruned to some extent to limit the amount of possibilities the agent can run into).

We will approach testing in a variety of manners. The first testing, smoke/regression testing, will be done manually. We will input a sentence and a partial sentence manually and see if the result is reasonable to a degree. This is simply to make

sure that the algorithm actually produces a sentence, not garbled nonsense.

Next, we will begin to train it: this will likely involve some manual interaction to adjust variables within either the properties of the neural network or something else, but will mostly involve automated training of the agent with episodes of creating sentences and rewards of inverse distance as described above.

One issue we may run into is that back-propagation in the neural network likely cannot happen at each time step once we do not test the method manually: while we test it manually, we are able to tell the neural network when a result is good or bad in terms of sentence completion, but once we leave the picture and testing becomes automated, we are not able to do so. Instead, reward is based on the distance from the nearest neighbor in word vector space from the database we decide to use—but this may not be a reasonable reward for every time step besides the last as sentences may be very different from how they will be completed. The agent needs to produce a combination of words to complete a sentence, and will not necessarily approach the best end result in a monotonically increasing manner.

4 Evaluation and Expected Outcomes

We wish our agent to produce some weird sentences, working sporadically. We will know we've succeeded when we're getting sentences similar to the ones in the space (i.e. rewards are maximized and distances are minimized).

If we manage to achieve this base goal, we will consider tackling quantifying the correctness of long sentences. This problem is suitable for reinforcement learning because the state can be modeled as an MDP because of the smaller state space and completing long sentences is a task where reward is obtained only much later than where a state might be. For now, we shelve this as a reach goal.

References

[1] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[2] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[3] Yonghui Wu. Smart compose: Using neural networks to help write emails. *Google AI Blog*, 2018.