Natural Language Processing: Project Background
CS 4980 003 – NLP
Fass, Grant; Kaja, Nicholas;

Primary Report

We are pursuing a project in natural language sentence completion for our project. Natural language sentence completion involves determining what word best fits in a blank present in a sentence. These types of questions are usually found on the Scholastic Aptitude Test (SAT). This type of problem is useful because it measures language model performance on problems that educational experts deem important. Models that perform well on this type of problem are likely to form better constructed sentences during sentence generation.

Our research found that there is primarily one method used to solve this task. This method does, however, have several variations. The primary method for forming a sentence completion model is to compute the probability of each one of the possible sentences and choose the most probable option. This probability computation can be done in a variety of ways. The most standard way would be to use n-grams and compute their probabilities. Using Latent Semantic Analysis is also possible. Another method used syntactic dependency trees. These methods of computing probabilities are sometimes optionally combined with methods of preserving long-range dependencies in sentences.

N-grams is likely one of the most common starting points for sentence completion models. This is likely due to the ease of their implementation and understanding. N-grams have multiple variations on their implementation that can lead to differences in model performance. For example, the Computation Approaches to Sentence Completion paper attempted to use n-

grams by themselves, maximum-entropy class-based n-grams, as well as composite approaches with recurrent neural networks for long-range dependencies (Zweig et al., 2012).

Latent Semantic Analysis is one of the approaches that differs in a more major way from using n-grams. Latent Semantic Analysis involves forming a term-frequency matrix for each word in an input document. This matrix is kept small by only using documents that include one of the possible options to complete the sentence. Once this matrix is constructed, they can compute the similarity scores of each of the potential answers and pick the highest (Zweig et al., 2012).

Another option is the use of syntactic dependency trees. This was the approach taken in the paper Dependency Language Models for Sentence Completion. This paper attempted a few variations in their approach such as an unlabeled dependency language model, and a labeled approach. The difference with the labeled approach was that it could better differentiate between the grammatical relations of words in the input documents. The paper ended up implementing this in a manner like n-grams. They applied extra terms such as 'ROOT' to the n-grams to help preserve some of the information at the start of the sentences (Gubbins & Vlachos, 2013).

Existing research in the field appears to be testing their models on SAT questions (Zweig et al., 2012), or on the Microsoft Research Sentence Completion Challenge dataset (Gubbins & Vlachos, 2013). These datasets consist of sentences and several imposter sentences. An imposter sentence is defined as "a sentence in which a single (fixed) word in the original sentence has been replaced by an imposter word with similar occurrence statistics." (Zweig & Chris, 2011)

For our project we plan to create several simple models. We will likely end up using various N-gram models as well as possible syntactic dependency trees. These models will be

constructed from Khan Academy lecture transcript data that we scraped using beautiful soup. We will then evaluate the performance of these models using either the SAT questions or Microsoft Research Sentence Completion Challenge Dataset. We will then use the best performing model to try and generate random sentences. This is done to get a feel for how coherent and legible the sentences our model forms are.

The model that we are going to be building will differ in several ways from the models that the researchers used. For starters, our models will be much simpler. This is because our primary goal is the investigation of this task and how it works. Additionally, we plan to make several changes to our models to see what effect they have. This includes features such as arbitrary or mixed n-gram sizes, unknown word tokens, paragraph tags, Laplacian smoothing, and named entity recognition token combination.

Another difference between our implementation and other implementations is the type of corpus used to train the model. Many of the models in the research were trained on a very large natural language domain such as newspaper articles (Zweig et al., 2012). Instead, our model will be trained on a smaller, more focused, domain of Khan Academy transcripts. The difference in the dataset we train the model on will likely affect the results of the model. This is part of the reason for the paper about combining n-gram language models with neural network language models. This is to allow for efficient domain adaptation, though we likely will not make it this far due to time constraints (Li et al., 2022).

We plan to use popular open-source tools and frameworks to assist us in this task. Some of the frameworks that we will be using are the Python Natural Language Tool Kit (NLTK) and

SpaCy. Additionally, we plan to leverage PyTorch and TensorFlow if we decide to implement a neural language model for long-range dependency preservation.

More Information About Example Systems & Their Implementations.

One example is described in this paper: https://arxiv.org/pdf/2210.14431.pdf

- Their code can be found in this repo: https://github.com/ghrua/NgramRes

In this paper, the authors aim to combine an n-gram language model with a neural language model. They claim that n-gram models (in this case 5-gram) can achieve decent performance, comparable in some test cases to models like GPT-2 in metrics like perplexity. So, they use a 5-gram language model as a base alongside a neural net language model with residual learning. They claim that the neural LM will approximate the "information gap" not learned or captured by the n-gram LM. This allows them to swap out the underlying n-gram model without retraining the entire neural net component. This allows for both cheaper models in terms of training and very quick domain adaptation.

The authors tested this model on 3 standard language tasks: language modeling, translation, and summarization. The model outperformed some popular baseline models in metrics like perplexity for language modeling, BLEU for translation, and ROUGE for summarization. The authors note that switching out the underlying n-gram model for a domain-specific one can decently improve performance. Improvements that they claim are comparable to fine-tuning the entire model on a domain-specific corpus.

Another example is described in the paper Computational Approaches to Sentence Completion. This paper is about using 'sentence-level semantic coherence' to answer questions like those that would be on SAT sentence completion questions. They attempted to solve this problem through using n-grams as well as through LSA. The n-gram model would theoretically record the information contained in the words around the sentence to allow it to complete better.

The Latent Semantic Analysis version attempts to evaluate the global coherence of the sentence. They attempted to use a few variants of N-gram models including a maximum-entropy class-based model. They also tried the recurrent neural network language model. They achieved the best results by combining the Latent Semantic Analysis based method with local information. They mentioned that other models related to this task "find a set of potential replacement words, and then they rank them." Their n-gram model was trained using trigrams on 1.1 billion words present in newspapers. Bigrams that occurred at least twice were retained as well. They constrained their vocabulary to all words that occur at least 100 times in the data. Thus leaving 124 thousand words for the vocabulary and 59 million n-grams. They implemented a recurrent neural network model to help retain the long-range dependencies between words. For running LSA they formed a term frequency matrix. They then computed the total similarity of potential answers with the rest of the words in the sentence and chose the most related option.

The paper Dependency Language Models for Sentence Completion is about utilizing syntactic information to improve the performance of sentence completion models. They aimed to solve this by creating two language models. They estimated the probability of sentences using syntactic dependency trees. They tested this on the Microsoft Research Sentence Completion Challenge. They cited improved performance over other similarly sized models. "These models are similar to standard n-gram language models, but instead of using the linear ordering of the words in the sentence, they generate words along paths in the dependency tree of the sentence."

The paper Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing is about prompt-based learning. It may be useful in the future but is not as directly related to our current direction for our project.

Sources:

Gubbins, J., & Vlachos, A. (2013, October). Dependency language models for sentence completion. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1405-1410). https://aclanthology.org/D13-1143.pdf

Li, H., Cai, D., Xu, J., & Watanabe, T. (2022). N-gram Is Back: Residual Learning of Neural Text Generation with n-gram Language Model. ArXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/2210.14431

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, *55*(9), 1-35. https://dl.acm.org/doi/full/10.1145/3560815

Zweig, G., Platt, J. C., Meek, C., Burges, C. J., Yessenalina, A., & Liu, Q. (2012, July). Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 601-610). https://aclanthology.org/P12-1063.pdf

Zweig, Geoffrey, and Chris J.C. Burges. *The Microsoft Research Sentence Completion Challenge*, Microsoft, Dec. 2011, www.microsoft.com/en-us/research/publication/the-microsoft-research-sentence-completion-challenge/.