Language Models in NLP: An Overview and Comparison of N-gram and RNN Models

Language models are a core component in the study of Natural Language Processing (NLP.) A language model, in the simplest terms, is a probability distribution over a sequence of words. There are various uses for language models in NLP, including but not limited to, machine translation, speech recognition, classification, and data compression. Common applications such as Siri, Google Translate, and Google Suggest all incorporate the use of language models. Language models can also be broken down into different types: statistical language models are models that utilize traditional statistical techniques to learn the probability of words; neural language models are deep learning models that "imitate" how neurons process information and learn. This review will focus on one model of each type, specifically the N-gram statistical language model and the basic Recurrent Neural Network model.

An N-gram model is one of the simplest and most common statistical language models. The model predicts the next word in a sequence based on the n number of words previously seen. For example, in a bi-gram model, the sentence "My favorite animal is cats." will have bigrams <My, favorite>, <favorite, animal>, <animal,is>, etc. After the model is trained, the model will predict the current word based on the probabilities of these bigrams. In the sentence above, the word "favorite" is always followed by the word "animal" so the model will predict that given "favorite", the next work should be "animal." Other well-known n-gram models include the unigram model (one word each, independent probabilities) and tri-grams (predictions will be based on previous two words.) N-gram models hold the Markov assumption, which states that the generation of the current word will only depend on the words within the fixed context.

N-gram models are widely used for NLP as they train quickly, don't require manual annotation, and are incremental models. However, there are several shortcomings with N-gram models. For one, they are considered "sparse representations of language" - meaning that the model is completely trained on co-occurring words. In the event that the model encounters an unseen word, the probability for the entire sequence would be zero unless further enhancements are added (ie. smoothing.) Furthermore, to achieve better accuracy with an N-gram model, one will generally have to increase the number of N words. The more words in an n-gram, the more context the model will be able to capture between words that are far apart. However, achieving this will require increasing computational power and will start to negatively impact both RAM and memory.

Neural language models are based on the basic idea to represent each word in a sequence using distributed representations, or word embeddings. These embeddings are concatenated to form an input layer. The model then takes the input layer and feeds it into a hidden layer, where internal computations (usually a non-linear activation function) are performed and extracted into an output layer. As more hidden layers are added, the model is able to learn more feature information from the input and define more complex dependencies between each word. Basic recurrent neural networks (RNNs), also known as Elman Nets, expand upon the general feedforward neural network by feeding the hidden state of the previous time step into the hidden state at the current time step. Therefore, the output layer is not just influenced by the current input layer, but also by the previous hidden layers of all the previous time steps. This allows an RNN to capture the "history" of the entire sequence from beginning to end.

Generally speaking, neural language models are more suitable for temporal/sequential data for several reasons. To begin with, a neural model that uses embeddings will have the advantage of being able to take in larger inputs without increasing the model size (fewer parameters to learn.) The model would be able to better predict probabilities for words that appear in similar contexts. For example, in the scenario of unseen words, an n-gram model would assign zero probability whereas the neural model would be able to produce a better generalization of the current token. An n-gram model can only handle fixed input/output lengths and make independent predictions within a fixed context (n-gram window.) RNN-based models tend to perform better because they address these limitations, by accepting variable input/output lengths and retaining information over the previous input. Since the previous token impacts the current token prediction, this tends to produce a more accurate result. Additionally, when increasing the input, a non-linear RNN model will scale much better than the linear N-gram model. Lastly, RNN models will utilize and incorporate past memory, capturing dependencies across the whole sequence that traditional n-gram models would not be able to.

In conclusion, both statistical language models and neural language models are useful tools to perform natural language processing tasks. For simple datasets, statistical language models are quick and efficient. For more complex and large datasets, neural language models will often produce better results.

References

1. *Choosing neural networks over N-gram models for natural language ...* (n.d.). Retrieved November 5, 2022, from https://towardsdatascience.com/choosing-neural-networks-over-n-gram-models-for-natural-language-processing-156ea3a57fc
2. Sanad, M. (2022, June 14). *Language model in NLP: Build Language Model in Python*. Analytics Vidhya. Retrieved November 4, 2022, from https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-language-model-nlp-python-code/
3. *A short overview of statistical language models - jon.dehdari.org*. (n.d.). Retrieved November 5, 2022, from https://jon.dehdari.org/tutorials/lm_overview.pdf