# The Roots of Statistical Language Modeling

Understanding papers on the topic, compiled by D. Gueorguiev 3/19/2024

## Introductory Notes

A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of *the curse of dimensionality*: a word sequence on which the model will be tested is likely to be different from all the word seen during training. Traditional but very successful approaches based on n-grams obtain generalization by concatenating very short overlapping sequences seen in the training set. Bengio et al ([[1]](https://github.com/dimitarpg13/large_language_models/blob/main/articles/A_Neural_Probabilistic_Language_Model_bengio03a.pdf)) proposed to fight the curse of dimensionality by ***learning a distributed representation for words*** which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously (1) a distributed representation of each work along with (2) the probability function for word sequences, expressed in terms of these representations.

Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of works that are similar (in the sense of having a nearby representation) to words forming an already seen sentence. Training such large models within reasonable time is itself a significant challenge. Report is presented on experiments using neural networks for the probability function, showing on two text corpora that the proposed approach significantly improves on state-of-art *n-gram* models, and that the proposed approach allows to take advantage of longer contexts.

It was mentioned in the previous paragraph that language modeling is difficult due to the *curse of dimensionality*.

It is obvious in case one wants to model the joint distribution between many discrete random variables and has been discussed widely in the literature. For instance, words in a sentence or discrete attributes in a data-mining task are represented by such joint distribution and fall into this category of many random variables.

*Example*: if we want to model the joint distribution of 10 consecutive words in NLP with a vocabulary of size, there are potentially free parameters. When modeling continuous variables, we obtain generalization more easily (e.g. with smooth classes of functions like multi-layer networks or Gaussian mixture models) because the function to be learned can be expected to have some local smoothness properties. For discrete spaces, the generalization structure is not as obvious: any change of these discrete variables may have a drastic impact on the value of the function to be estimated, and when the number of values that each discrete variable can take is large, most observed objects are almost maximally far from each other in hamming distance.

A useful way to visualize how different learning algorithms generalize, inspired from the view of non-parametric density estimation, is to think of how probability mass that is initially concentrated on the training points (e.g. training sentences) is distributed in a larger volume, usually in some form of neighborhood around the training points. In high dimensions, it is crucial to distribute probability mass where it matters rather than uniformly in all directions around each training point. Bengio et al. show in [[1]](https://github.com/dimitarpg13/large_language_models/blob/main/articles/A_Neural_Probabilistic_Language_Model_bengio03a.pdf) a new approach for statistical learning which is more suitable for generalization is fundamentally different than the previous *n-gram*-based techniques.

**Definition**: *Statistical language model*

Such language model where the conditional probability of the next word given all previous ones can be expressed as:

(1)

where is the -th word and writing sub-sequence .

When building statistical models of natural language, one considerably reduces the difficulty of this modeling problem by taking advantage of word order, and the fact that temporally closer words in the word sequence are statistically more dependent. Thus, n-gram models construct tables of conditional probabilities for the next word, for each one of large number of *contexts*, i.e. combinations of the last words:

(2)

We only consider those combinations of successive words that actually occur in the training corpus, or that occur frequently enough. What happens when a new combination of words appears that was not seen in the training corpus? We do not want to assign zero probability to such cases, because such new combinations are likely to occur, and they will occur even more frequently for larger context sizes. A simple answer is to look at the probability predicted using a smaller context size, as done in *back-off trigram models* or in *smoothed (or interpolated) trigram models*. So, in such models, how is generalization basically obtained from sequences of words seen in the training corpus to new sequences of words? A way to understand how this happens is to think about a generative model corresponding to these *interpolated* or *backoff -gram* models. Essentially, a new sequence of words is generated by “gluing” very short and overlapping pieces of length or up to words that have been seen frequently in the training data. The rules for obtaining the probability of the next piece are implicit in the particulars of the back-off or interpolated -*gram* algorithm. Typically, trigrams have been used in the past (). Obviously, there is much more information in the sequence that immediately precedes the word to predict than just the identity of the previous couple of words. Bengio et al in [[1]](https://github.com/dimitarpg13/large_language_models/blob/main/articles/A_Neural_Probabilistic_Language_Model_bengio03a.pdf) make the following improvements on the following disadvantages on the -*gram*-derived approaches described above:

*disadvantage 1*: the n-gram based approaches are not taking into account contexts farther than 1 or 2 words

*disadvantage 2*: the n-gram based approaches are not taking account the *semantic similarity* between words

For example: we have the following sentence in the training corpus: “*The cat is walking in the bedroom*”. We would expect the latter to be helpful for generalization as “*A dog was running in a room*” due to semantic and grammatical similarity.

The proposed approach by Bengio et al is summarized as follows:

1. Associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in )
2. Express the joint *probability function* of word sequences in terms of the feature vectors of these words in the sequence,

and

1. Learn simultaneously the *word feature vectors* and the parameters of that *probability function*

The feature vector represents different aspects of the word – each word is associated with a point in a vector space. The number of features (e.g. , or in the experiments) is much smaller than the size of the vocabulary (say ). The probability function is expressed as a product of conditional probabilities of the

## Literature

[1] [A Neural Probabilistic Language Model, Yoshua Bengio et al, Université de Montréal, Québec, 2003](https://github.com/dimitarpg13/large_language_models/blob/main/articles/A_Neural_Probabilistic_Language_Model_bengio03a.pdf)