
A word n -gram language model is a purely statistical model of language. It has been superseded by recurrent neural network –based models, which have been superseded by large language models . [1] It is based on an assumption that the probability of the next word in a sequence depends only on a fixed size window of previous words. If only one previous word is considered, it is called a bigram model; if two words, a trigram model; if n − 1 words, an n -gram model. [2] Special tokens are introduced to denote the start and end of a sentence ■ s ■ {\displaystyle \langle s\rangle } and ■ / s ■ {\displaystyle \langle /s\rangle } .

To prevent a zero probability being assigned to unseen words, the probability of each seen word is slightly lowered to make room for the unseen words in a given corpus . To achieve this, various smoothing methods are used, from simple "add-one" smoothing (assigning a count of 1 to unseen n -grams, as an uninformative prior) to more sophisticated techniques, such as Good–Turing discounting or back-off models .

Unigram model

A special case, where n = 1, is called a unigram model. Probability of each word in a sequence is independent from probabilities of other word in the sequence. Each word's probability in the sequence is equal to the word's probability in an entire document.

P uni (t 1 t 2 t 3) = P (t 1) P (t 2) P (t 3) . {\displaystyle P_{\text{uni}}(t_{1}t_{2}t_{3})=P(t_{1})P(t_{2})P(t_{3}).}

The model consists of units, each treated as one-state finite automata . [3] Words with their probabilities in a document can be illustrated as follows.

Total mass of word probabilities distributed across the document's vocabulary, is 1.

∑ word in doc P (word) = 1 {\displaystyle \sum _{\text{word in doc}}P({\text{word}})=1}

The probability generated for a specific query is calculated as

P (query) = ∏ word in query P (word) {\displaystyle P({\text{query}})=\prod _{\text{word in query}}P({\text{word}})}

Unigram models of different documents have different probabilities of words in it. The probability distributions from different documents are used to generate hit probabilities for each query. Documents can be ranked for a query according to the probabilities. Example of unigram models of two documents:

Bigram model

In a bigram word (n = 2) language model, the probability of the sentence I saw the red house is approximated as

P (I, saw, the, red, house) ≈ P (I ■ ■ s ■) P (saw ■ I) P (the ■ saw) P (red ■ the) P (house ■ red) P (■ / s ■ ■ house) {\displaystyle P({\text{I, saw, the, red, house}})\approx P({\text{I}}\mid \langle s\rangle)P({\text{saw}}\mid {\text{I}})P({\text{the}}\mid {\text{saw}})P({\text{red}}\mid {\text{the}})P({\text{house}}\mid {\text{red}})P(\langle /s\rangle \mid {\text{house}})}

Trigram model

In a trigram (n = 3) language model, the approximation is

P (I, saw, the, red, house) ≈ P (I ■ ■ s ■ , ■ s ■) P (saw ■ ■ s ■ , I) P (the ■ I, saw) P (red ■ saw, the) P (house ■ the, red) P (■ / s ■ ■ red, house) {\displaystyle P({\text{I, saw, the, red, house}})\approx P({\text{I}}\mid \langle s\rangle ,\langle s\rangle)P({\text{saw}}\mid \langle s\rangle ,I)P({\text{the}}\mid {\text{I, saw}})P({\text{red}}\mid {\text{saw, the}})P({\text{house}}\mid {\text{the, red}})P(\langle /s\rangle \mid {\text{red, house}})}

Note that the context of the first n – 1 n -grams is filled with start-of-sentence markers, typically denoted .

Additionally, without an end-of-sentence marker, the probability of an ungrammatical sequence *I saw the would always be higher than that of the longer sentence I saw the red house.

Approximation method

The approximation method calculates the probability P (w 1 , … , w m) {\displaystyle P(w_{1},\ldots ,w_{m})} of observing the sentence w 1 , … , w m {\displaystyle w_{1},\ldots ,w_{m}}

P (w 1 , … , w m) = ∏ i = 1 m P (w i ■ w 1 , … , w i – 1) ≈ ∏ i = 2 m P (w i ■ w i – (n – 1) , … , w i – 1) {\displaystyle P(w_{1},\ldots ,w_{m})=\prod _{i=1}^{m}P(w_{i}\mid w_{1},\ldots ,w_{i-1})\approx \prod _{i=2}^{m}P(w_{i}\mid w_{i-(n-1)},\ldots ,w_{i-1})}

It is assumed that the probability of observing the i th word w i (in the context window consisting of the preceding i – 1 words) can be approximated by the probability of observing it in the shortened context window consisting of the preceding n – 1 words (n th -order Markov property). To clarify, for n = 3 and i = 2 we have P (w i ■ w i – (n – 1) , … , w i – 1) = P (w 2 ■ w 1) {\displaystyle P(w_{i}\mid w_{i-(n-1)},\ldots ,w_{i-1})=P(w_{2}\mid w_{1})} .

The conditional probability can be calculated from n -gram model frequency counts:

P (w i ■ w i – (n – 1) , … , w i – 1) = c o u n t (w i – (n – 1) , … , w i – 1 , w i) c o u n t (w i – (n – 1) , … , w i – 1) {\displaystyle P(w_{i}\mid w_{i-(n-1)},\ldots ,w_{i-1})={\frac {\mathrm {count} (w_{i-(n-1)},\ldots ,w_{i-1},w_{i})}{\mathrm {count} (w_{i-(n-1)},\ldots ,w_{i-1})}}}

Out-of-vocabulary words

An issue when using n -gram language models are out-of-vocabulary (OOV) words. They are encountered in computational linguistics and natural language processing when the input includes words which were not present in a system's dictionary or database during its preparation. By default, when a language model is estimated, the entire observed vocabulary is used. In some cases, it may be necessary to estimate the language model with a specific fixed vocabulary. In such a scenario, the n -grams in the corpus that contain an out-of-vocabulary word are ignored. The n -gram probabilities are smoothed over all the words in the vocabulary even if they were not observed. [4]

Nonetheless, it is essential in some cases to explicitly model the probability of out-of-vocabulary words by introducing a special token (e.g.) into the vocabulary. Out-of-vocabulary words in the corpus are effectively replaced with this special token before n -grams counts are cumulated. With this option, it is possible to estimate the transition probabilities of n -grams involving out-of-vocabulary words. [5]

n -grams for approximate matching

n -grams were also used for approximate matching. If we convert strings (with only letters in the English alphabet) into character 3-grams, we get a 26 3 {\displaystyle 26^{3}} -dimensional space (the first dimension measures the number of occurrences of "aaa", the second "aab", and so forth for all possible combinations of three letters). Using this representation, we lose information about the string. However, we know empirically that if two strings of real text have a similar vector representation (as measured by cosine distance) then they are likely to be similar. Other metrics have also been applied to vectors of n -grams with varying, sometimes better, results. For example, z-scores have been used to compare documents by examining how many standard deviations each n -gram differs from its mean occurrence in a large collection, or text corpus , of documents (which form the "background" vector). In the event of small counts, the g-score (also known as g-test) gave better results.

It is also possible to take a more principled approach to the statistics of n -grams, modeling similarity as the likelihood that two strings came from the same source directly in terms of a problem in Bayesian inference .

n -gram-based searching was also used for plagiarism detection .

Bias–variance tradeoff

To choose a value for n in an n -gram model, it is necessary to find the right trade-off between the stability of the estimate against its appropriateness. This means that trigram (i.e. triplets of words) is a common choice with large training corpora (millions of words), whereas a bigram is often used with smaller ones.

Smoothing techniques

There are problems of balance weight between infrequent grams (for example, if a proper name appeared in the training data) and frequent grams . Also, items not seen in the training data will be given a probability of 0.0 without smoothing . For unseen but plausible data from a sample, one can introduce pseudocounts . Pseudocounts are generally motivated on Bayesian grounds.

In practice it was necessary to smooth the probability distributions by also assigning non-zero probabilities to unseen words or n -grams. The reason is that models derived directly from the n -gram frequency counts have severe problems when confronted with any n -grams that have not explicitly been seen before – the zero-frequency problem . Various smoothing methods were used, from simple "add-one" (Laplace) smoothing (assign a count of 1 to unseen n -grams; see Rule of succession) to more sophisticated models, such as Good–Turing discounting or back-off models . Some of these methods are equivalent to assigning a prior distribution to the probabilities of the n -grams and using Bayesian inference to compute the resulting posterior n -gram probabilities. However, the more sophisticated smoothing models were typically not derived in this fashion, but instead through independent considerations.

Linear interpolation (e.g., taking the weighted mean of the unigram, bigram, and trigram)

Good–Turing discounting

Witten–Bell discounting

Lidstone's smoothing

Katz's back-off model (trigram)

Kneser–Ney smoothing

Skip-gram language model

Skip-gram language model is an attempt at overcoming the data sparsity problem that the preceding model (i.e. word n -gram language model) faced. Words represented in an embedding vector were not necessarily consecutive anymore, but could leave gaps that are skipped over (thus the name "skip-gram"). [6]

Formally, a k -skip- n -gram is a length- n subsequence where the components occur at distance at most k from each other.

For example, in the input text:

the set of 1-skip-2-grams includes all the bigrams (2-grams), and in addition the subsequences

In skip-gram model, semantic relations between words are represented by linear combinations , capturing a form of compositionality . For example, in some such models, if v is the function that maps a word w to its n -d vector representation, then

v (k i n g) − v (m a l e) + v (f e m a l e) ≈ v (q u e e n) {\displaystyle v(\mathrm {king})-v(\mathrm {male})+v(\mathrm {female})\approx v(\mathrm {queen})}

where ≈ is made precise by stipulating that its right-hand side must be the nearest neighbor of the value of the left-hand side. [7] [8]

Syntactic n -grams

Syntactic n -grams are n -grams defined by paths in syntactic dependency or constituent trees rather than the linear structure of the text. [9] [10] [11] For example, the sentence "economic news has little effect on financial markets" can be transformed to syntactic n -grams following the tree structure of its dependency relations : news-economic, effect-little, effect-on-markets-financial. [9]

Syntactic n -grams are intended to reflect syntactic structure more faithfully than linear n -grams, and have many of the same applications, especially as features in a vector space model . Syntactic n -grams for certain tasks gives better results than the use of standard n -grams, for example, for authorship attribution. [12]

Another type of syntactic n -grams are part-of-speech n -grams, defined as fixed-length contiguous overlapping subsequences that are extracted from part-of-speech sequences of text. Part-of-speech n -grams have several applications, most commonly in information retrieval. [13]

Other applications

n -grams find use in several areas of computer science, computational linguistics , and applied mathematics.

They have been used to:

design kernels that allow machine learning algorithms such as support vector machines to learn from string data [citation needed]

find likely candidates for the correct spelling of a misspelled word [14]

improve compression in compression algorithms where a small area of data requires n -grams of greater length

assess the probability of a given word sequence appearing in text of a language of interest in pattern recognition systems, speech recognition , optical character recognition (OCR), intelligent character recognition (ICR), machine translation and similar applications

improve retrieval in information retrieval systems when it is hoped to find similar "documents" (a term for which the conventional meaning is sometimes stretched, depending on the data set) given a single query document and a database of reference documents

improve retrieval performance in genetic sequence analysis as in the BLAST family of programs

identify the language a text is in or the species a small sequence of DNA was taken from

predict letters or words at random in order to create text, as in the dissociated press algorithm

cryptanalysis [citation needed]

See also

Collocation

Feature engineering

Hidden Markov model

Longest common substring

MinHash

n-tuple

String kernel

References