n-gram

For applications in computational genomics, see k-mer. For Google phrase-usage graphs, see Google Ngram Viewer.
Not to be confused with Engram (disambiguation).

In the fields of computational linguistics and probability, an ***n*-gram** is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called **shingles**.[1]

An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram". Larger sizes are sometimes referred to by the value of n, e.g., "four-gram", "five-gram", and so on.

1 Applications

An ***n*-gram model** is a type of probabilistic language model for predicting the next item in such a sequence in the form of a $(n-1)$–order Markov model.[2] n-gram models are now widely used in probability, communication theory, computational linguistics (for instance, statistical natural language processing), computational biology (for instance, biological sequence analysis), and data compression. Two benefits of n-gram models (and algorithms that use them) are simplicity and scalability – with larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently.

2 Examples

Figure 1 shows several example sequences and the corresponding 1-gram, 2-gram and 3-gram sequences.

Here are further examples; these are word-level 3-grams and 4-grams (and counts of the number of times they appeared) from the Google n-gram corpus.[3]

- ceramics collectables collectibles (55)
- ceramics collectables fine (130)
- ceramics collected by (52)
- ceramics collectible pottery (50)
- ceramics collectibles cooking (45)

4-grams

- serve as the incoming (92)
- serve as the incubator (99)
- serve as the independent (794)
- serve as the index (223)
- serve as the indication (72)
- serve as the indicator (120)

3 *n*-gram models

An ***n*-gram model** models sequences, notably natural languages, using the statistical properties of n-grams.

This idea can be traced to an experiment by Claude Shannon's work in information theory. Shannon posed the question: given a sequence of letters (for example, the sequence "for ex"), what is the likelihood of the next letter? From training data, one can derive a probability distribution for the next letter given a history of size n : a = 0.4, b = 0.00001, c = 0, ...; where the probabilities of all possible "next-letters" sum to 1.0...

More concisely, an n-gram model predicts x_i based on $x_{i-(n-1)}, \ldots, x_{i-1}$. In probability terms, this is $P(x_i \mid x_{i-(n-1)}, \ldots, x_{i-1})$. When used for language modeling, independence assumptions are made so that each word depends only on the last $n-1$ words. This Markov model is used as an approximation of the true underlying language. This assumption is important because it massively simplifies the problem of learning the language model from data. In addition, because of the open nature of language, it is common to group words unknown to the language model together.

Note that in a simple n-gram language model, the probability of a word, conditioned on some number of previous words (one word in a bigram model, two words in a trigram model, etc.) can be described as following a categorical distribution (often imprecisely called a "multinomial distribution").

In practice, the probability distributions are smoothed by assigning non-zero probabilities to unseen words or n-grams; see smoothing techniques.

4 Applications and considerations

n-gram models are widely used in statistical natural language processing. In speech recognition, phonemes and sequences of phonemes are modeled using a *n*-gram distribution. For parsing, words are modeled such that each *n*-gram is composed of *n* words. For language identification, sequences of characters/graphemes (*e.g.*, letters of the alphabet) are modeled for different languages.[4] For sequences of characters, the 3-grams (sometimes referred to as "trigrams") that can be generated from "good morning" are "goo", "ood", "od ", "d m", " mo", "mor" and so forth (sometimes the beginning and end of a text are modeled explicitly, adding "__g", "_go", "ng_", and "g__"). For sequences of words, the trigrams that can be generated from "the dog smelled like a skunk" are "# the dog", "the dog smelled", "dog smelled like", "smelled like a", "like a skunk" and "a skunk #".

Practitioners more interested in multiple word terms might preprocess strings to remove spaces. Many simply collapse whitespace to a single space while preserving paragraph marks, because the whitespace is frequently either an element of writing style or introduces layout or presentation not required by the prediction and deduction methodology. Punctuation is also commonly reduced or removed by preprocessing and is frequently used to trigger functionality.

n-grams can also be used for sequences of words or almost any type of data. For example, they have been used for extracting features for clustering large sets of satellite earth images and for determining what part of the Earth a particular image came from.[5] They have also been very successful as the first pass in genetic sequence search and in the identification of the species from which short sequences of DNA originated.[6]

n-gram models are often criticized because they lack any explicit representation of long range dependency. (In fact, it was Chomsky's critique of Markov models in the late 1950s that caused their virtual disappearance from natural language processing, along with statistical methods in general, until well into the 1980s.) This is because the only explicit dependency range is $(n-1)$ tokens for an *n*-gram model, and since natural languages incorporate many cases of unbounded dependencies (such as wh-movement), this means that an *n*-gram model cannot in principle distinguish unbounded dependencies from noise (since long range correlations drop exponentially with distance for any Markov model). For this reason, *n*-gram models have not made much impact on linguistic theory, where part of the explicit goal is to model such dependencies.

Another criticism that has been made is that Markov models of language, including *n*-gram models, do not explicitly capture the performance/competence distinction discussed by Chomsky. This is because *n*-gram models are not designed to model linguistic knowledge as such, and make no claims to being (even potentially) complete models of linguistic knowledge; instead, they are used in practical applications.

In practice, *n*-gram models have been shown to be extremely effective in modeling language data, which is a core component in modern statistical language applications.

Most modern applications that rely on *n*-gram based models, such as machine translation applications, do not rely exclusively on such models; instead, they typically also incorporate Bayesian inference. Modern statistical models are typically made up of two parts, a prior distribution describing the inherent likelihood of a possible result and a likelihood function used to assess the compatibility of a possible result with observed data. When a language model is used, it is used as part of the prior distribution (e.g. to gauge the inherent "goodness" of a possible translation), and even then it is often not the only component in this distribution.

Handcrafted features of various sorts are also used, for example variables that represent the position of a word in a sentence or the general topic of discourse. In addition, features based on the structure of the potential result, such as syntactic considerations, are often used. Such features are also used as part of the likelihood function, which makes use of the observed data. Conventional linguistic theory can be incorporated in these features (although in practice, it is rare that features specific to generative or other particular theories of grammar are incorporated, as computational linguists tend to be "agnostic" towards individual theories of grammar).

4.1 Out-of-vocabulary words

Main article: Statistical machine translation

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.[7]

Nonetheless, it is essential in some cases to explicitly model the probability of out-of-vocabulary words by introducing a special token (e.g. *<unk>*) into the vocabulary. Out-of-vocabulary words in the corpus are effectively replaced with this special <unk> 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.[8]

5 *n*-grams for approximate matching

Main article: Approximate string matching

n-grams can also be used for efficient approximate matching. By converting a sequence of items to a set of *n*-grams, it can be embedded in a vector space, thus allowing the sequence to be compared to other sequences in an efficient manner. For example, if we convert strings with only letters in the English alphabet into single character 3-grams, we get a 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. For example, both the strings "abc" and "bca" give rise to exactly the same 2-gram "bc" (although {"ab", "bc"} is clearly not the same as {"bc", "ca"}). 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) may give better results for comparing alternative models.

Another method for approximate matching is signature files. The study reported in [9] shows that a bit-sliced signature file can be compressed to a smaller size than an inverted file which is the standard way of implementing a vector space approach. With a signature width less than half the number of unique n-grams, the signature file method is about as fast as the inverted file method, and significantly smaller.

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 can also be used for plagiarism detection.

6 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
- find likely candidates for the correct spelling of a misspelled word
- 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, OCR (optical character recognition), 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.

7 Bias-versus-variance trade-off

What goes into picking the *n* for the *n*-gram?

With *n*-gram models 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.

7.1 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 is 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 are 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

7.2 Skip-gram

In the field of computational linguistics, in particular language modeling, **skip-grams**[10] are a generalization of *n*-grams in which the components (typically words) need not be consecutive in the text under consideration, but may leave gaps that are *skipped* over.[11] They provide one way of overcoming the data sparsity problem found with conventional *n*-gram analysis.

Formally, an n-gram is a consecutive subsequence of length n of some sequence of tokens $w_1 \dots wn$. 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 rain in Spain falls mainly on the plain*

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

> *the in*, *rain Spain*, *in falls*, *Spain mainly*, *falls on*, *mainly the*, and *on plain*.

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.[12][13][14] 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.[12]

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.[15]

9 See also

- Collocation
- Hidden Markov model
- n-tuple
- String kernel
- MinHash
- Feature extraction
- Longest common substring problem

10 References

[1] Broder, Andrei Z.; Glassman, Steven C.; Manasse, Mark S.; Zweig, Geoffrey (1997). "Syntactic clustering of the web". *Computer Networks and ISDN Systems* **29** (8): 1157–1166. doi:10.1016/s0169-7552(97)00031-7.

[2] https://class.coursera.org/nlp/lecture/17

[3] Alex Franz and Thorsten Brants (2006). "All Our *N*-gram are Belong to You". *Google Research Blog*. Retrieved 2011-12-16.

[4] Ted Dunning (1994). "Statistical Identification of Language". New Mexico State University. Technical Report MCCS 94-273

[5] Soffer, A (1997). "Image categorization using texture features". *Proceedings of the Fourth International Conference on* **1** (233): 237. doi:10.1109/ICDAR.1997.619847.

[6] Tomović, Andrija; Janičić, Predrag; Kešelj, *Vlado (2006). "*n*-Gram-based classification and unsupervised hierarchical clustering of genome sequences".* Computer Methods and Programs in Biomedicine ***81*** *(2): 137–153. doi:10.1016/j.cmpb.2005.11.007.*

[7] Wołk, K.; Marasek, K.; Glinkowski, W. (2015). "Telemedicine as a special case of Machine Translation". *Computerized Medical Imaging and Graphics.*

[8] Wołk K., Marasek K. (2014). *Polish-English Speech Statistical Machine Translation Systems for the IWSLT 2014*. Proceedings of the 11th International Workshop on Spoken Language Translation. Tahoe Lake, USA.

[9] Carterette, Ben; Can, Fazli (2005). "Comparing inverted files and signature files for searching a large lexicon". *Information Processing and Management* **41** (3): 613–633. doi:10.1016/j.ipm.2003.12.003.

[10] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.1629

[11] David Guthrie; et al. (2006). "A Closer Look at Skip-gram Modelling" (PDF).

[12] Sidorov, Grigori; Velazquez, Francisco; Stamatatos, Efstathios; Gelbukh, Alexander; Chanona-Hernández, Liliana (2012). "Syntactic Dependency-based *n*-grams as Classification Features". *LNAI 7630*: 1–11.

[13] Sidorov, Grigori (2013). "Syntactic Dependency-Based *n*-grams in Rule Based Automatic English as Second Language Grammar Correction". *International Journal of Computational Linguistics and Applications* **4** (2): 169–188.

[14] Figueroa, Alejandro; Atkinson, John (2012). "Contextual Language Models For Ranking Answers To Natural Language Definition Questions". *Computational Intelligence* **28** (4): 528–548. doi:10.1111/j.1467-8640.2012.00426.x.

[15] Sidorov, Grigori; Velasquez, Francisco; Stamatatos, Efstathios; Gelbukh, Alexander; Chanona-Hernández, Liliana. "Syntactic *n*-Grams as Machine Learning Features for Natural Language Processing". *Expert Systems with Applications* **41** (3): 853–860. doi:10.1016/j.eswa.2013.08.015.

- Christopher D. Manning, Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press: 1999. ISBN 0-262-13360-1.
- White, Owen; Dunning, Ted; Sutton, Granger; Adams, Mark; Venter, J.Craig; Fields, Chris (1993). "A quality control algorithm for dna sequencing projects". *Nucleic Acids Research* **21** (16): 3829–3838. doi:10.1093/nar/21.16.3829.
- Frederick J. Damerau, *Markov Models and Linguistic Theory*. Mouton. The Hague, 1971.
- Figueroa, Alejandro; Atkinson, John (2012). "Contextual Language Models For Ranking Answers To Natural Language Definition Questions". *Computational Intelligence* **28** (4): 528–548. doi:10.1111/j.1467-8640.2012.00426.x.
- Brocardo, Marcelo Luiz; Issa Traore; Sherif Saad; Isaac Woungang (2013). *Authorship Verification for Short Messages Using Stylometry* (PDF). IEEE Intl. Conference on Computer, Information and Telecommunication Systems (CITS).

11 External links

- Google's Google Book *n*-gram viewer and Web *n*-grams database (September 2006)
- Microsoft's web *n*-grams service
- 1,000,000 most frequent 2,3,4,5-grams from the 425 million word Corpus of Contemporary American English
- Peachnote's music ngram viewer
- Stochastic Language Models (*n*-Gram) Specification (W3C)
- Michael Collin's notes on *n*-Gram Language Models

12 Text and image sources, contributors, and licenses

12.1 Text

- **N-gram** *Source:* https://en.wikipedia.org/wiki/N-gram?oldid=697947480 *Contributors:* Damian Yerrick, Zundark, The Anome, Michael Hardy, Kku, David Joffe, Benwing, Bernhard Bauer, Natch, Lysy, DavidCary, Bovlb, Macrakis, Khalid hassani, Sam Hocevar, Quota, Neumannkun, Dalebrearcliffe, Dupuy, Rajah, Jonsafari, Diego Moya, Ricky81682, Voxadam, Markaci, Stuartyeates, FrancisTyers, Sandius, SDC, Gwil, Qwertyus, Rjwilmsi, Mpo~enwiki, Phantomsteve, Ian Cheese, Doncram, Johndburger, H@r@ld, Canley, Allens, SmackBot, RedHouse18, Atomota, Jab843, Radagast83, Gnack, Daniel.Cardenas, Zovres, Nasz, Jergosh, Smith609, Xionbox, CmdrObot, Searchtools, Dgw, Mrebus, OmerMor, Cs california, Fyedernoggersnodden, Wikid77, Tdunning, Dawnseeker2000, Alphachimpbot, Dustinsmith, Kcho, Nono64, DrKay, Wumpus3000, TWCarlson, Xnuala, Soshial, Pleasantville, TXiKiBoT, Phileinsophia, Sapphic, Aarre, Nicksh, Xavivars, Firefly4342, Melcombe, Martarius, Schtickwriter, XLinkBot, DrakeMcSmooth, Lingprofe, Christian Kirchhoff~enwiki, Addbot, Fgnievinski, LaaknorBot, Yobot, AnomieBOT, DaveHorsman, GrouchoBot, Green Cardamom, Adler.fa, X7q, Jakesyl, Mundart, Chenopodiaceous, Redrose64, Jandalhandler, Lumengsimon, ПешСай, Ycchew, EmausBot, Tuankiet65, Ш, Dhemery, Dennis97519, Hhchen1105, BG19bot, FeralOink, BattyBot, Ryugecin, Bronzik, Iftekhar Naim, Monkbot, Mandudu123, Hgomezpy, Krz.wolk, User8178 and Anonymous: 108

12.2 Images

- **File:Text_document_with_red_question_mark.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a4/Text_document_with_red_question_mark.svg *License:* Public domain *Contributors:* Created by bdesham with Inkscape; based upon Text-x-generic.svg from the Tango project. *Original artist:* Benjamin D. Esham (bdesham)

12.3 Content license

- Creative Commons Attribution-Share Alike 3.0