Latent Semantic Analysis

Tyler Forgione

December 30, 2025

Abstract

In this project, a Latent Semantic Analysis model was trained on Wikitext-103, a dataset of over 100 million tokens. The model only kept words that were used at least 25 times and was projected onto a 300 dimension latent space. It was then tested on numerous word-similarity and analogy tests and performed relatively well given the training corpus size as well as the minimum count set in the vocabulary. In the future, this model will be compared to simpler and more complex models in order to get a benchmark for semantic models.

Introduction

Word Co-occurrence

The most simplistic model one could use in order to model word learning is a word co-occurrence matrix. Simply, how many times a pair of words co-occur within some window (typically two words). For example, in a sentence like ”the dog bit the mailman”, the word ”bit” would co-occur with every other word in the sentence since the window extends two words on either side. However, the word ”mailman” would only co-occur with the two previous words.

A notable issue with the sentence above is that 40% of the words have no meaning; the word ”the” provides no information about any of the words in its window. Words that provide no information are essentially useless and can thus be removed. These words are known as stop-words and are usually extremely common words that occur with anything and everything. A word that co-occurs with everything is just as useful as a word that never occurs. With this in mind, a sentence like ”the dog bit the mailman” becomes ”dog bit mailman”. These two sentences are functionally the same, and with our two word window we now have that ”mailman” and ”dog” occur together. This adds some information to both words instead of wasting it on a stop-word.

This pure word co-occurence matrix can model semantic representations of words. However, it cannot work on homonyms or homographs and cannot account for the ability of humans to learn in ways other than by hearing/reading the word in a sentence (Bullinaria and Levy, 2007). For example, humans may learn the meanings of words visually or by reading the definition in a dictionary. These methods obviously cannot be represented in a co-occurrence matrix and would likely need other techniques.

Pointwise Mutual Information

In order to keep using the matrix previously described, there needs to be extra information added to each word pair. How often two words are used in the same context can be useful in certain situations. For example, predators will occur in the same window as words like ”claw” or ”fang” than prey animals.

The method used in this paper is called Pointwise Mutual Information (PMI). It is based on Mutual Information (MI), which is defined as

$$I(X;Y) = \sum_x \sum_y P_{(X,Y)}(x,y) \log\left(\frac{P_{(X,Y)}(x,y)}{P_X(x)P_Y(y)}\right)$$

This equation describes the amount of information obtained about one random variable while observing another (Shannon, 1948). Simply, it measures the difference between the joint behaviour of two variables from their behaviour if they were assumed to be independent. In the context of words, a closely related pair would have high mutual information because their joint probability would be far larger than if we assumed they were independent (Church and Hanks, 1990).

PMI is simply the part inside of the logarithm in the equation above. The definition is the same as for MI, except that MI is the average PMI over all events.

Notice that PMI requires probabilities. In Natural Language Processing, these are the probabilities of word occurrence or co-occurrence. This is why the co-occurrence matrix is used as a base for PMI. An important use of PMI is to disallow skew caused by frequency. If two words rarely occur together, there is very little information about that pair. PMI evaluates whether this pair is statistically meaningful by comparing the joint probability to the independent probability. This allows pairs of words with seemingly no information a chance to be highly valuable.

Singular Value Decomposition & Latent Semantic Analysis

When using word co-occurrence matrices, the result is an extremely large sparse matrix. This is expected, however it is unwanted. A sparse matrix, especially a very large one, will be unable to relate two words that should be related because their rows in the matrix will be too different. This is known as the data sparsity problem (Allison et al., 2006). The idea is simply that, in language, even when using extremely large corpora, there will be word pairs or word sequences that do not ever together. These sequences may very well be possible in the language in question but just not in the corpus.

Furthermore, by using a sparse matrix, and thus a high-dimension matrix, a phenomenon known as the curse of dimensionality occurs (Bellman, 1961). In the context of a sparse word co-occurrence matrix, this means two things. First, most of the word vectors will be almost orthogonal. That is, they will have nearly no relation to each other. Obviously, this is impossible, as at least a few of the words in the corpus must be related in some way. This has implications for cosine similarity, which would be near zero. Second, in high-dimensional sparse matrices, small changes in the corpus may have drastic effects on the model.

The most common way to fix the issues described above is to densify the matrix. This is done using Singular Value Decomposition (SVD). Simply, SVD is a factorization method of the form $M = U\Sigma V^T$. Intuitively, this performs three operations on a single matrix: a rotation/reflection, scaling, and another rotation/reflection. In the context of a sparse word co-occurrence matrix, SVD densifies the matrix and turns each word (a row in the original matrix) into a dense word vector. The components of this vector are latent semantic dimensions inferred by co-occurrence statistics. The idea of beginning with a sparse matrix and using SVD to truncate it is known as Latent Semantic Analysis (LSA) (Landauer et al., 1998).

SVD reducing dimensionality is a positive for language models because extra information is gained through latent semantics and models with less dimensions are typically faster. In this project, the model being tested is LSA. LSA is essentially this pathway: a word co-occurrence matrix with PMI applied to it and then truncated using SVD. Then, cosine similarity would be used during testing.

Wikitext-103

The dataset this model will be trained on is wikitext-103, a dataset of over 100 million tokens (Merity et al., 2016). This dataset was pulled from a number of Wikipedia articles and includes punctuation. This dataset was used to train the model in this paper simply because it blended size and breadth quite well. A larger dataset would take much longer to train, especially considering the lack of GPUs available. Further, certain datasets may be too specific to a certain topic (e.g.: science) and thus miss out on some common words.

Methods

This simple LSA model was created in three steps. First, the sparse word co-occurrence matrix was generated based on a vocabulary derived from the corpus. Next, PMI was applied to the sparse matrix. Finally, the matrix was truncated using SVD.

Sparse Matrix

In order to build the sparse word co-occurrence matrix, the corpus needs to be preprocessed and a vocabulary needs to be created. Preprocessing is quite simple and involves making all tokens lowercase, removing stop-words, and replacing any character that is not a lowercase letter or whitespace. The reason for making all the words lowercase is so that the same word is not mapped twice. Stop-words are removed due to their high usage and low information content. Finally, only lowercase letters and whitespaces are kept because the other characters are unnecessary for the model. Hyphenated words are the same to the model whether hyphenated or not.

Next, a vocabulary must be built before populating the matrix. The vocabulary is essentially the set of all the words kept in the corpus after some filtering is done. In this case, the filtering is simply a minimum number of uses. This vocabulary was built using a minimum usage of 25, meaning that all words used less than 25 times were discarded. This serves the purpose of decreasing the size of the matrix which increases speed during SVD. These discarded words would have likely had little information, however without testing, this cannot be confirmed. While time was not a massive issue, I like using my computer and do not want to spend two days performing singular value decomposition.

Applying PMI

This is implemented exactly as described in the introduction. Some math is done to assign a number to word pairs based on the probability that they occur together versus independently. This replaces the pure word co-occurrence number that was previously there.

Truncating with SVD

Singular Value Decomposition was performed using sklearn's TruncatedSVD model. For the purposes of this project, the number of components was 300. The number of components is the number of latent dimensions or singular vectors to keep after decomposition. Effectively, the data is being projected onto a 300-dimension latent space. The 300 components after using TruncatedSVD fit-transform method will be ordered by explained variance. Note that SVD and Piecewise Component Analysis (PCA) save for the fact that PCA requires a centred matrix.

The reason for choosing 300 components is that it is a good mix of quality and speed. Unfortunately, there is a limit to the number of cores on a processor and while the processor on my home computer is above-average and TruncatedSVD is multithreaded, there is only so much time the computer can have a 100% CPU load before I need to start using it again. Using any less than 300 components saves on memory and time, but leads to a worse model generally. Furthermore, using more than 300 components could lead to a better model, but this would require testing which is even more time performing SVD that is not available. Thus, 300 was chosen as a (arbitrary) happy middleground.

Testing

This LSA model was tested on numerous different tasks and datasets. First, the model was tasked with completing analogies with different themes. For example, in a capital cities theme, an analogy could be "Ottawa-Canada Berlin-?" and the model would have to choose Germany. Then, the model was tested on SimLex999, a dataset in which humans rate the similarity of words in a pair (Hill et al., 2015). The model did the same as the humans and judged word pair similarity and was then compared to human scores. Next, the model was tested on WordSim353, which works exactly as SimLex999 but with 353 word pairs instead of 999 (Finkelstein et al., 2002). Four more similar tests were used that function like the previous two. These are RG-65, MEN, MC-30, and MTURK-771. Each of these tests takes some number of word pairs and assigns a similarity to them. The way to assign a score to a model is to compute Spearman's ρ.

Results

Analogies

In the analogy dataset, the model performed quite well overall (Table 5). It performed far better on analogy sets in which fewer tokens were missing, which is completely understandable. For example, the model a terrible accuracy on the analogy set for currency, however 660 out of 866 analogies were impossible to complete. Thus, it can be deduced that the model likely did not have enough information relating to currency to properly complete those analogies, even if all the tokens were in the vocabulary. The model performed quite well on analogy sets with common terms such as capitals of common countries or nationality adjectives. However, even with few missing tokens, it performed very poorly on the city-in-state set. Perhaps a more US-focused dataset would have remedied this but this is a non-issue generally. The model was good enough for commonly used English words.

SimLex999

As mentioned previously, to get the model's score on the SimLex999 test, Spearman's ρ was calculated. This dataset is split into parts of speech (adjectives, nouns, and verbs). The model was able to identify 943 of the total word pairs in its vocabulary. As seen in table 1, the model performed noticeably worse on verbs than on either nouns or adjectives.

POS	Spearman ρ	Pairs Used
Adjective (A)	0.3264	111
Noun (N)	0.2928	663
Verb (V)	0.0382	219

Table 1: Spearman correlation by part of speech (POS).

WordSim353

In table 2 are the results for the model when tested on the WordSim353 dataset. The model performs extremely well on the similarity set and worse on the relatedness set. This is expected due to LSA being a count model, using association (similar words co-occurring) than true similarity.

Subset	**Spearman ρ**	**Pairs Used**
Similarity Set	0.6008	203
Relatedness Set	0.3768	252

Table 2: Performance on the WordSim-353 benchmark.

Rest

For the other four datasets the model will be tested on, the results are shown in table 3. Noteably, the model performs just as well as it did on WordSim353's similarity set. Since these datasets are also similarity (not relatedness) based, this outcome is expected.

Dataset	**Spearman ρ**	**Pairs Used**
RG-65	0.6567	64
MEN	0.5636	2983
MC-30	0.6785	30
MTurk-771	0.5237	769

Table 3: Spearman correlation on standard word similarity benchmarks.

Further, the MEN results may be split into parts of speech, just as SimLex999 is. These results are shown in table 4. Notably, the model performs far better on verbs than it did on SimLex999's verbs.

POS	**Spearman ρ**	**Pairs Used**
Noun (n)	0.6091	1991
Adjective (j)	0.5623	96
Verb (v)	0.5135	29

Table 4: Performance on the MEN dataset by part of speech (POS).

Discussion

The results for this LSA model are quite similar to most others in the literature. Unfortunately, lack of resources meant this model was not optimal (mainly in terms of the training corpus) but it did well nonetheless. Most of the results in the analogies dataset are respectable, and the ones that are not were generally in categories that the training corpus was not focused on.

The performance of the model on SimLex999 was as expected. Nouns and adjectives only have so many use-cases and contexts and thus are easier for LSA to deal with. Verbs, on the other hand, are versatile and have innumerable cases in which they may be viable. Specifically, context matter more with verbs than other parts of speech. The subject of the verb may be outside of its window but is undoubtedly required in order to gain knowledge of that verb. This is in complete contrast with nouns, which really only need nearby words as information.

On WordSim353, the performance on the similarity set is quite good and is far worse on the relatedness set. This is in line with the usual statistics from LSA models on this gold-standard benchmark.

Similarly, for the other four benchmarks used, performance is exactly as expected. The only caveat is that performance is much higher for verbs in the MEN set than on SimLex999. This may be due to the fact that there were only 29 pairs used from MEN. This low sample size may be skewing results in favour of the model when they shouldn't be. Another possibility is that the verb pairs used were extremely obvious for the model (e.g.: common verbs with few possible contexts).

Conclusion

This model performed quite admirably considering both its simplicity and speed in training. LSA is obviously just a mathematical model and exclusively models statistics, but it is a good representation of human word-learning through reading. Evidently, word learning comes in many forms and word co-occurrence is just one.

In the future, this model will be trained on a larger corpus of at least a billion tokens and will be compared to all models from least complex to most complex. This has already been done, however I'd like to have my own benchmark for when I make my own model or finetune some others.

This project served mostly as my foray into natural language processing and will serve as a basis for any future experimenting I do. Possibly, I'd like to combine a word co-occurrence model like this one with grounded representations of words in order to model human learning through vision as a whole (reading words and seeing objects).

Category	Accuracy	Correct / Total	Missing
capital-common-countries	0.4249	215 / 506	0
capital-world	0.1975	560 / 2835	1689
currency	0.0097	2 / 206	660
city-in-state	0.0794	185 / 2330	137
family	0.4079	155 / 380	126
gram1-adjective-to-adverb	0.0903	84 / 930	62
gram2-opposite	0.0860	65 / 756	56
gram3-comparative	0.4122	549 / 1332	0
gram4-superlative	0.1109	110 / 992	130
gram5-present-participle	0.4335	430 / 992	64
gram6-nationality-adjective	0.6789	981 / 1445	154
gram7-past-tense	0.2929	457 / 1560	0
gram8-plural	0.4504	536 / 1190	142
gram9-plural-verbs	0.2672	217 / 812	58

Table 5: Word analogy task accuracy by category.

References

Allison, B., Guthrie, D., & Guthrie, L. (2006). Another look at the data sparsity problem. In P. Sojka, I. Kopeček, & K. Pala (Eds.), *Text, speech and dialogue* (pp. 327–334). Springer Berlin Heidelberg.

Bellman, R. E. (1961). *Adaptive control processes: A guided tour*. Princeton University Press.

Bullinaria, J. A., & Levy, J. P. (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, *39*(3), 510–526. https://doi.org/10.3758/BF03193020

Church, K., & Hanks, P. (1990, January). Word association norms, mutual information, and lexicography. https://aclanthology.org/J90-1003/

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, *20*(1), 116–131.

Hill, F., Reichart, R., & Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, *41*(4), 665–695. https://doi.org/10.1162/COLI_a_00237

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to latent semantic analysis. *Discourse Processes*, *25*(2-3), 259–284.

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer sentinel mixture models. *CoRR*, *abs/1609.07843*. http://arxiv.org/abs/1609.07843

Shannon, C. E. (1948). A mathematical theory of communication [Reprinted with corrections (PDF) available at https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf]. *The Bell System Technical Journal*, *27*(3–4), 379–423,
bibrangessep 623–656.