Word Embeddings

Slava Mikhaylov
Professor of Public Policy and Data Science

Institute for Analytics and Data Science, Department of Computer Science
Department of Government
University of Essex

Outline

Word Embeddings

Word Embeddings

Word Embeddings

- A geometric way to capture the “meaning” of a word via a low-dimensional vector.
- Currently used a lot in Natural Language Processing (NLP) to answer search queries or translate from one language to another.
- Basically, a 300-dimensional vector is used to capture the nuances of word meaning.

Properties of Word Embeddings

- Cosine similarity: the similarity between two words (as rated by humans on a [1,1] scale) correlates with the cosine of the angle between their vectors.
- The cosine for ‘milk‘ and ‘cow‘ may be 0.6; for ‘milk‘ and ‘stone‘ 0.2.

Properties of Word Embeddings

- Word embeddings can solve analogy relationships via linear algebra.
- `man : woman ::king : ??` can be solved looking for word w such that $v_{king} - v_w$ is most similar to $v_{man} - v_{woman}$

$$min||v_w - v_{king} + v_{man} - v_{woman}||^2$$

Properties of Word Embeddings

- Solves about 75% of standard word analogy questions.
- Unsupervised method. The embeddings are constructed using big unannotated corpus.
- No analogy specific training.
- From fMRI brain imaging analysis similar to how the human brain encodes meaning (See Tom Mitchell et al. (2008). "Predicting Human Brain Activity Associated with the Meanings of Nouns." *Science*, 320, 1191.)

Computing Word embeddings

"You shall know a word by the company it keeps." (Firth, 1957)
`https://en.wikipedia.org/wiki/Distributional_semantics`

- Word vector is a succinct representation of the distribution of other words around this word.
- (cow, drink, babies, calcium: milk).

Computing Word embeddings

- Suppose the dictionary has N distinct words (in practice, $N = 100,000$).
- Take a very large text corpus (e.g., Wikipedia) and let $Count_5(w_1, w_2)$ be the number of times w_1 and w_2 occur within a distance 5 of each other in the corpus.
- Then the word embedding for a word w is a vector of dimension N, with one coordinate for each dictionary word.
- The coordinate corresponding to word w_2 is $Count_5(w, w_2)$.
- You can also extend it to cooccurence of w with ngrams.

Computing Word embeddings

- This embedding uses high-dimensional vectors (100,000-dimensional).
- We can reduce dimensionality by taking the rank-300 singular value decomposition (SVD).
- Method used directly in Latent Semantic Indexing (LSI). http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf
- We can improve the method by replacing the counts with their logs. That's Latent Semantic Analysis (LSA). http://lsa.colorado.edu/papers/plato/plato.annote.html.

Vector Space Models

- Insight from LSA: Dimension reduction via SVD improves quality of embedding. Level of average American school kid.
- Vector Space Models are modifications on the above.
- Embeddings also improve with various reweights: TF-IDF, PMI, log, etc.
- See overview of VSM in Turney and Pantel paper: `https://www.jair.org/media/2934/live-2934-4846-jair.pdf`.

Word2Vec

- Introduced in 2013 by Mikolov et al. at Google `https://code.google.com/archive/p/word2vec/`.
- Related to neural net models for language.
- Word embedding corresponds to the neural net's internal representation of the word. See here for more: `http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/`.

Word2Vec

$$Pr[w|w_1, w_2, \ldots, w_5] \propto exp(v_w \cdot (\frac{1}{5} \sum_i v_{w_i}))$$

- The left hand side gives the empirical probability that word w occurs in the text conditional on the last five words being w_1 through w_5.

Making sense of word2vec

- Original papers are not clear why they actually work.
- Levy and Goldberg (in a series of papers) explain word2vec: `https://levyomer.wordpress.com`.
- They show word2vec to be a modern version of older vector space models.

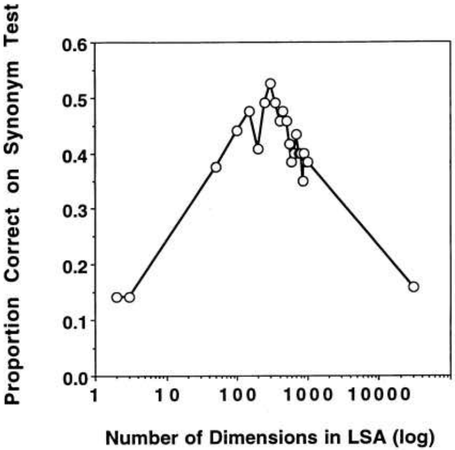
Making sense of word2vec

- Another approach, by Pennington et al. (Standord NLP group) suggest a GLoVe model (`http://nlp.stanford.edu/projects/glove/`).
- They use a weighted-SVD strategy to find word embeddings.
- They also provide an intuitive explanation why these embeddings solve word analogy tasks.

Making sense of word2vec

- For a more in-depth explanation of "black magic" see "RAND-WALK: A Latent Variable Model Approach to Word Embeddings" `https://arxiv.org/abs/1502.03520`.
- They provide a new generative model for text, and a clearer insight into the causative relationship between word meanings and the cooccurence probabilities.
- Intuitively, corpus generation is a dynamic process driven by the random walk of a *discourse* vector.
- Direction of this discourse vector represents what is being talked about.
- Each word is then related to this discourse vector through a time-invariant latent vector that captures its correlations with the discourse vector.

Low dimensional embeddings vs high dimensional embeddings



Performance curve of word embeddings

- Too few parameters make the model incapable of fitting to the signal;
- Too many parameters, and it starts overfitting (e.g., fitting to noise instead of the signal).
- Thus the dimension constraint acts as a regularizer for the optimization.

Performance curve of word embeddings

- Arora et al. "RAND-WALK" paper shows that relations correspond to directions (unlike Levy and Goldberg but closer to GLoVe).
- Particularly for semantic analogies.
- Using linear algebra and enough examples we can e.g. predict new leaders based on the list of current leaders.

Word embeddings in R

- text2vec `http://text2vec.org`
- textmineR `https://github.com/TommyJones`
- wordVectors `https://github.com/bmschmidt`