Drawing a "map of words": word embeddings and applications to machine translation and sentiment analysis

Mostapha Benhenda

Artificial Intelligence Club, Kyiv

mostaphabenhenda@gmail.com

January 28, 2015

Overview

Introduction

We want to compute a "map of words", i.e. a representation:

$$R : \text{Words} = \{w_1, ..., w_N\} \to \text{Vectors} = \{R(w_1), ..., R(w_N)\} \subset \mathbb{R}^d$$

such that:

$$w_i \approx w_j \qquad \text{(meaning of words)}$$

is equivalent to:

$$R(w_i) \approx R(w_j) \qquad \text{(distance of vectors)}$$

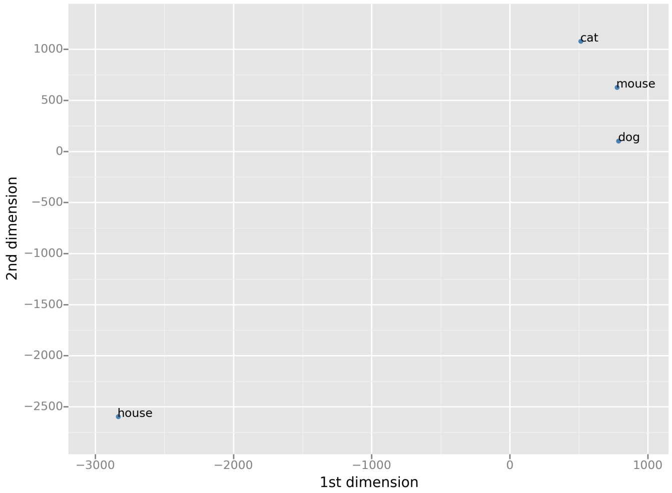
These vectors are interesting because:

- the computer can "grasp" the meaning of words, just by looking at the distance between vectors.
- We can feed prediction algorithms (linear regression,...) with these vectors, and hopefully get good accuracy, because the representation is "faithful" to the meaning of words.

For example, if we have the list of words:

cat, dog, mouse, house

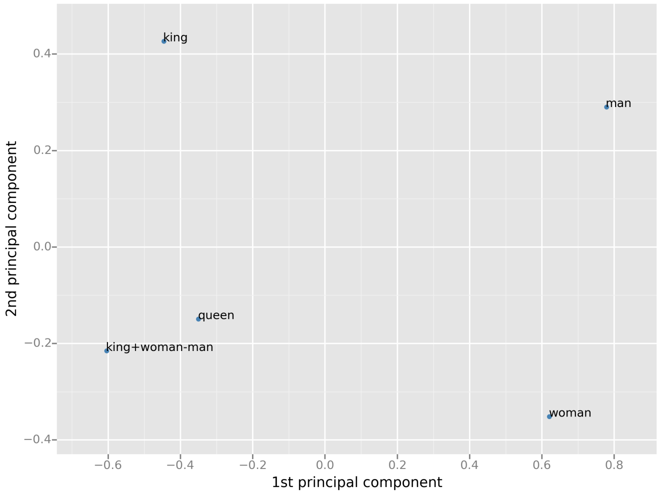
We expect the most distant word to be: ?



(this is a 2-dimensional visualization of a 300-dimensional space, using t-SNE, a visualization technique that preserves clusters of points)

Even better, we can sometimes make additions and substractions of vectors, and draw parallelisms. For example,

$$\text{king} + \text{woman - man} \approx ?$$



(2-dimensional visualization of a 300-dimensional space, using PCA, a visualization technique that preserves parallelisms)

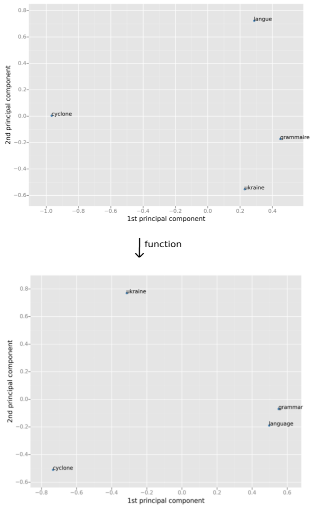
- Idea behind all algorithms (Word2vec, GloVe,...): "You shall know a word by the company it keeps" (John R. Firth, 1957)
- The more often 2 words are near each other in a text (the training data), the closer their vectors will be.
- We hope that 2 words have close meanings if, statistically, they are often near each other.

So we need quite big datasets:

- Google News English: 100 Billion words
- Wikipedia French or Spanish: $\simeq$ 100 Million words
- MT 11 French: 200 Million words
- MT 11 Spanish: 84 Million words

Application to machine translation

Idea: we compute maps of all English words, all French words, and we "superpose" the 2 maps, and we should get an English/French translator.



- [Mikolov et al. 2013] (from Google), made an English $\rightarrow$ Spanish translator (among other languages).
- I tried to reproduce their results for French $\rightarrow$ English, and French $\rightarrow$ Spanish.
- Using their algorithm Word2vec, they trained their vectors on the dataset MT 11, and on Google News.
- I did the same on MT 11 and Wikipedias for French and Spanish (trained with Gensim package in Python), and I used their Google News-trained vectors for English.

- Then they took the list of the 5000 most frequent English words, and their Google translation in Spanish.
- They train a linear transformation W that approximates the English $\rightarrow$ Spanish translation, i.e. they take W that minimizes:

$$\sum_{i=1}^{5000} \|W(v_i) - G(v_i)\|^2$$

where $G(v_i)$ is the Google-translation of v_i.

- They test the accuracy of W on the next 1000 most common words of the list (1-accuracy). They also test the accuracy up to 5 attempts, i.e. they test if $G(v_i)$ belongs to the 5 nearest neighbors of $W(v_i)$ (5-accuracy).
- I did (almost) the same for French $\rightarrow$ English, and French $\rightarrow$ Spanish.

- code available here: https://drive.google.com/open?id=0B86WKpvkt66BY09TSHJoekRqZjg&authuser=0
- computer-intensive task, I recommend using Amazon Web Services, or similar.

Results

- Mikolov, English → Spanish:

Training data	**1-accuracy**	**5-accuracy**
Google News	50%	75%
MT 11	33%	51%

- Me, French → Spanish:

Training data	**1-accuracy**	**5-accuracy**
Wikipedia	na	<10%
MT 11	25%	37%

Results

- Mikolov, English $\rightarrow$ Spanish:

Training data	1-accuracy	5-accuracy
Google News	50%	75%
MT 11	33%	51%

- Me, French $\rightarrow$ Spanish:

Training data	1-accuracy	5-accuracy
Wikipedia	na	<10%
MT 11	25%	37%

French (Wikipedia) $\rightarrow$ English (Google News) 10-accuracy: $< 10\%$.

Results

- Mikolov, English → Spanish:

Training data	1-accuracy	5-accuracy
Google News	50%	75%
MT 11	33%	51%

- Me, French → Spanish:

Training data	1-accuracy	5-accuracy
Wikipedia	na	<10%
MT 11	25%	37%

French (Wikipedia) → English (Google News) 10-accuracy: $< 10\%$.

My conclusion: Wikipedia = cauchemar !

Sentiment analysis of movie reviews

- Goal: we want to determine if a movie review is positive or negative.
- Toy problem in machine learning: reviews are ambiguous, emotional, full of sarcasm,...
- Long-term goal: computer can understand emotions.
- Commercial applications of sentiment analysis:
    - marketing: customer satisfaction,...
    - finance: predict market trends,...
- Example of review: "This movie contains everything you'd expect, but nothing more".

Vector averaging (Kaggle tutorial)

We work on the IMDB dataset. To predict the sentiment (+ or -) of a review:

- let the vector of a review be the average of the vectors of its words
- We use these vectors as input to a supervised learning algorithm (e.g. SVM, random forests...)
- We get 83% accuracy (not the best method on this Kaggle challenge, easier methods work better)
- Limitation of the method: the order of words is lost, because addition is a commutative operation.

Convolutional Neural Networks (deep learning) (work in progress)

- CNN are biology-inspired neural network models, initially introduced for image processing.
- they preserve spatial structure: the input is a $n \times m$ matrix (the pixels of the image)
- but here, the input is a sentence (with its "spatial structure" preserved):

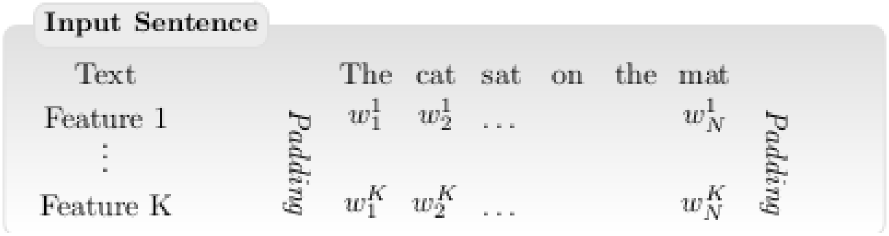


Figure : source: [Collobert et al. 2011]

Other applications of word embeddings

- Innovative search engine (ThisPlusThat), where we can subtract queries, for example:

$$\text{pizza} + \text{Japan -Italy} \rightarrow \text{sushi}$$

- Recommendation systems for online shops

Example of word embedding algorithm: GloVe

GloVe: "Global Vectors" algorithm made in Stanford by [Pennington, Socher, Manning, 2014]. There are 2 steps:

1. Build the co-occurrence matrix X from the training text
2. Factorize the matrix X to get vectors

1. Build the co-occurrence matrix X:
For the first step, we apply the principle "you shall know a word by the company it keeps":

- The *context window* $C(w)$ of size 2 of the word $w=$ Ukraine (for example), is given by:

The national flag of Ukraine is yellow and blue.

(in practice, we take the context window size around 10)

- The number of times 2 words i and j lie in the same context window is denoted by $X_{i,j}$.
- The symmetric matrix $X = (X_{i,j})_{1\leq i,j\leq N}$ is the *co-occurrence matrix*.

2. Factorize the co-occurrence matrix X:

- To extract vectors from a symmetric matrix, we can write:

$$X_{i,j} = < v_i, v_j > \qquad v_i \in \mathbb{R}^d,$$

where d is an integer fixed by us a priori (a hyperparameter). i.e. we can write:

$$X = \mathrm{Gram}(v_1, ..., v_N)$$

- This formula does not give a good empirical performance, but: for any scalar function f, $(f(X_{i,j}))_{1 \le i,j \le N}$ is still a symmetric matrix.
- Let's find f that works!

- Let i, j be 2 words, for example: $i=$ fruit, $j=$house. The third word $k=$apple has a meaning closer to fruit than to house, so $X_{i,k}/X_{j,k}$ is small.
- If $k=$room (closer to "house" than to "apple"), then $X_{i,k}/X_{j,k}$ is large.
- If $k=$ sky (far from both "house" and "apple"), then $X_{i,k}/X_{j,k} \simeq 1$.

$\rightarrow$ the ratio of co-occurrences $X_{i,k}/X_{j,k}$ is important to capture meaning of words. So we should look at $f(X_{i,k}/X_{j,k})$.

- On the other hand, if we want to combine 3 vectors and the scalar product in a "natural" way, we do not have much choice:

- Let i, j be 2 words, for example: $i=$ fruit, $j=$house. The third word $k=$apple has a meaning closer to fruit than to house, so $X_{i,k}/X_{j,k}$ is small.
- If $k=$room (closer to "house" than to "apple"), then $X_{i,k}/X_{j,k}$ is large.
- If $k=$ sky (far from both "house" and "apple"), then $X_{i,k}/X_{j,k} \simeq 1$.

$\rightarrow$ the ratio of co-occurrences $X_{i,k}/X_{j,k}$ is important to capture meaning of words. So we should look at $f(X_{i,k}/X_{j,k})$.

- On the other hand, if we want to combine 3 vectors and the scalar product in a "natural" way, we do not have much choice:

$$< v_i - v_j, v_k >= f(X_{i,k}/X_{j,k})$$

- Let i, j be 2 words, for example: $i=$ fruit, $j=$house. The third word $k=$apple has a meaning closer to fruit than to house, so $X_{i,k}/X_{j,k}$ is small.
- If $k=$room (closer to "house" than to "apple"), then $X_{i,k}/X_{j,k}$ is large.
- If $k=$ sky (far from both "house" and "apple"), then $X_{i,k}/X_{j,k} \simeq 1$.

$\rightarrow$ the ratio of co-occurrences $X_{i,k}/X_{j,k}$ is important to capture meaning of words. So we should look at $f(X_{i,k}/X_{j,k})$.

- On the other hand, if we want to combine 3 vectors and the scalar product in a "natural" way, we do not have much choice:

$$< v_i - v_j, v_k >= f(X_{i,k}/X_{j,k})$$

- We also have:

$$< v_i, v_k > - < v_j, v_k >= f(X_{i,k}) - f(X_{j,k})$$

- So $f = \log$

- We cannot factorize the matrix $\log X$ explicitly, i.e. we cannot directly compute v_i, v_j such that:

$$< v_i, v_j >= \log X_{i,j}$$

- but we compute an approximation, by minimizing a cost function J:

$$\min_{v_1,\ldots,v_N} J(v_1, ..., v_N) \sim \sum_{i,j=1}^{N} (< v_i, v_j > - \log X_{i,j})^2$$

- To do that, we use gradient descent (standard method).

Looks like cooking, but:

- good empirical performance (at least similar to Word2vec)
- cost function J similar to Word2vec
- GloVe model is analogous to Latent Semantic Analysis (LSA). In LSA:
  - co-occurrence matrix is a word-document matrix X (In GloVe: word-word matrix)
  - we factorize $\sim \log(1 + X_{i,j})$ (SVD, not Gram matrix)

Conclusion: GloVe model is worth to study!

Future work

- Doc2vec [Mikolov et al. 2014]
- Deep learning (Convolutional Neural Networks): Natural Language Processing (almost) from Scratch [Collobert et al. 2011]

Future work

- Doc2vec [Mikolov et al. 2014]
- Deep learning (Convolutional Neural Networks): Natural Language Processing (almost) from Scratch [Collobert et al. 2011]
- Suggestion: have a deep learning study group in Kyiv!
- Applications of deep learning: NLP, images, videos, speech,...
  $\rightarrow$ money!!

Future work

- Doc2vec [Mikolov et al. 2014]
- Deep learning (Convolutional Neural Networks): Natural Language Processing (almost) from Scratch [Collobert et al. 2011]
- Suggestion: have a deep learning study group in Kyiv!
- Applications of deep learning: NLP, images, videos, speech,...
  $\rightarrow$ money!!

Pre-requisites: a little of:

1. coding (Python or Matlab or Java...)
2. linear algebra (matrix multiplication)
3. calculus (chain rule)
4. enthusiasm!

Register at: http://deeplearningkyiv.doorkeeper.jp/

References

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., ... & Bengio, Y. (2010)
Theano: a CPU and GPU math expression compiler
Proceedings of the Python for scientific computing conference (SciPy), (Vol. 4, p. 3)

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011)
Natural language processing (almost) from scratch
The Journal of Machine Learning Research, 12, 2493-2537

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013)
Distributed representations of words and phrases and their compositionality
Advances in Neural Information Processing Systems pp. 3111-3119

Mikolov, T., Quoc V. Le, Sutskever, I. (2013)
Exploiting similarities among languages for machine translation
arXiv preprint arXiv:1309.4168.

References

Mikolov, T., Quoc V. Le (2014)
Distributed representations of sentences and documents
arXiv preprint arXiv:1405.4053.

Pennington, J., Socher, R., & Manning, C. D. (2014)
Glove: Global vectors for word representation
Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 12.

Řehůřek, R., & Sojka, P. (2010)
Software framework for topic modelling with large corpora
Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014), 12.