Word Embedding - Attention

E. Scornet

Fall 2020

Generic point of view

Definition: Word embedding

- Aims at mapping words or phrases from the vocabulary to real-valued vectors.
- Involves a mathematical embedding from a space with one dimension per word to a continuous vector space with a much lower dimension.

Many different techniques:

- Latent semantic analysis
- Word2Vec
- Glove
- FastText
- ...

Outline

Occurrence Matrix

Need for a **term-document matrix** which describes the occurrences of terms in documents; it is a sparse matrix whose rows correspond to terms and whose columns correspond to documents

	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Word Vector (Passage Vector)

Document Vector

One way to compute an occurrence matrix is TF-IDF (term frequency - inverse document frequency) most used technique in 2015, see e.g., Beel et al. 2016

TF-IDF

Term frequency

First use by Luhn 1957: *The weight of a term that occurs in a document is simply proportional to the term frequency.*

Simplest choice: $tf(t, d) = f_{t,d}$ (number of times the term t occurs in document d)

Inverse document frequency

First use by Sparck Jones 1972: *The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs*

Simplest choice:

$$idf(t, d) = \log\left(\frac{|D|}{1 + |\{d \in D, t \in d\}|}\right).$$

TF-IDF weights

$$w(t, d) = tf(t, d) idf(t, d).$$

Post-processing TF-IDF

Low rank space - Latent semantic analysis (LSA)

Once the occurrence matrix is computed, you can use a dimension reduction technique (SVD for example) to lower the number of variables describing documents.

Ranking using matching score

For a new document d_{new} which is simply a set of words, the matching score of a document d in the corpus to d_{new} is

$$\mathrm{Score}(d_{\mathrm{new}}, d) = \sum_{t \in d_{\mathrm{new}}} w(t, d).$$

Warning : This solution is biased towards long documents where more of your terms will appear.

Post-processing TF-IDF

Ranking using cosine similarity

Cosine similarity between two vectors d_1, d_2 is

$$\mathrm{Cosine}(d_1, d_2) = \frac{\langle d_1, d_2 \rangle}{\|d_1\| \|d_2\|}$$

Usually, cosine similarity is computed between vectors from the TF-IDF matrix. It takes into account the document length and thus the number of times a term is repeated.

Distance based on whether words occur or not

Given two documents d_1 and d_2, one can compute

$$\mathrm{Jaccard}(d_1, d_2) = \frac{|t, t \in d_1, t \in d_2|}{|t \in d_1 \text{ or } t \in d_2|}$$

It counts the number of common words in the two documents, without taking into account repetitions.

Applications

- Compare the documents in the low-dimensional space (data clustering, document classification).

- Find similar documents across languages, after analyzing a base set of translated documents (cross language retrieval).

- Find relations between terms

Outline

Word2Vec

["Efficient estimation of word representations in vector space", Mikolov, Chen, et al. 2013]

Two different versions:

- Continuous Bag Of Words (CBOW)
  Predict a word given the surrounding words in a sentence

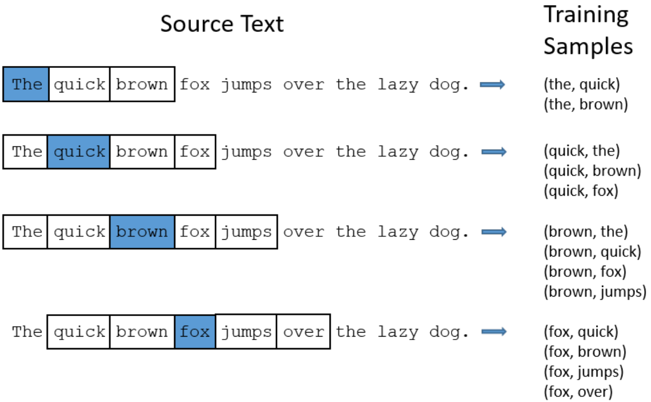
- Skip-Gram
  Predict the surrounding words of a given word in a sentence

  In each case, we are not interested by the prediction but by the hidden layer of the resulting neural network.

Word2Vec: Skip Gram

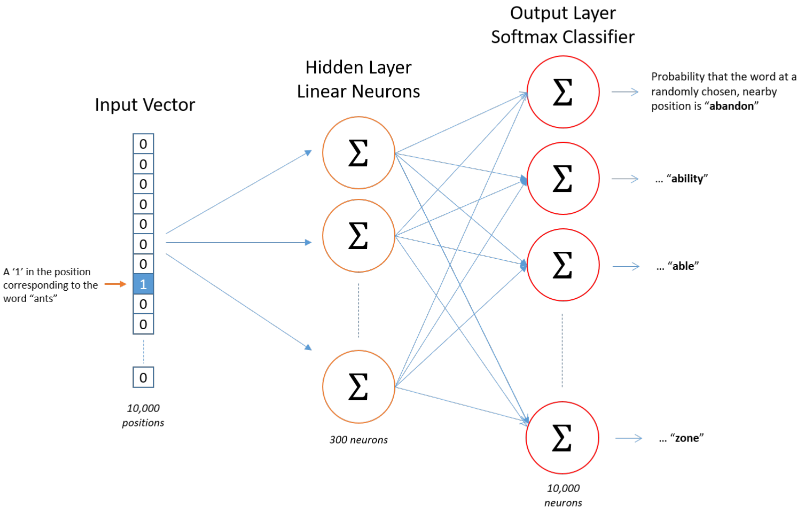
Given a word (in blue) in a sentence, try to guess which words are just before or after the blue word.

Feed the network with pairs of words: input (blue word) / output (one word close to the input in the sentence). The closeness is determined by a *window size* (here equal to 2).



Reference:
`http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/`

Word2Vec: Skip Gram - Neural Network



The Hidden layer gives a representation for each word.

https://medium.com/@vishwasbhanawat/the-architecture-of-word2vec-78659ceb6638

Negative sampling

Reference : `https://cs224d.stanford.edu/lecture_notes/notes1.pdf`

Softmax involves every probability of the output layer: too slow to compute

Instead of saying :

- One neuron should be close to 1, the other close to zero

Say

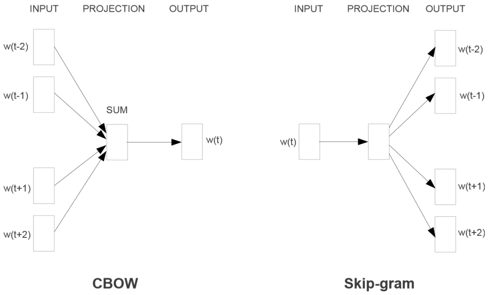
- One neuron should be close to 1, and k others (typically $k = 5$) should be close to zero.

The k words are called negative words (words whose probability to be associated with the input is low) and are sampled based on the original corpus with a unigram distribution to power 3/4 (unigram distribution being the probability of occurrence of each word in the corpus).

Since the power is less than 1, it **emphasizes the words with small probability** compared to the original unigram distribution.

Improvements

["Distributed representations of words and phrases and their compositionality", Mikolov, Sutskever, et al. 2013]



- **Best window size**: 10 for Skip-gram Model and 4 for CBOW
- **Subsampling**: Delete each word i in the training set with the probability

$$P_{\mathrm{delete}}(i) = 1 - \sqrt{\frac{\varepsilon}{f_i}},$$

where f_i is the frequency of the word in the document, and $\varepsilon \simeq 10^{-5}$.

Outline

GloVe (Global Vector)

["Glove: Global vectors for word representation", Pennington et al. 2014]: Leverage information on the whole corpus

Let X be the matrix of co-occurrences, that is the element X_{ij} is defined as the number of times word$j$ occurs in the context of word i, that is when word i and j are distant of less than `Windows size`=10 words in the sentence.

The algorithm learns two sets of representations $w_1, \dots, w_V$ and $\tilde{w}_1, \dots, \tilde{w}_V$ where w_j and $\tilde{w}_j$ are latent representation of word j (space of dimension 300 typically).

The objective function to minimize in order to find the best weights $\mathbf{w}, \tilde{\mathbf{w}}$ is

$$J(\mathbf{w}, \tilde{\mathbf{w}}) = \sum_{i,j=1}^{V} f(X_{ij}) (w_i^T \tilde{w}_j - \log X_{ij})^2,$$

where f is a weighting function chosen as

$$f(x) = \begin{cases} (x/x_{max})^\alpha & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases}$$

with $\alpha = 3/4$ and $x_{max} = 100$.

Weights $\mathbf{w}, \tilde{\mathbf{w}}$ are randomly initialized and a gradient descent type procedure (AdaGrad) is used. The algorithm outputs $\mathbf{w} + \tilde{\mathbf{w}}$ as proposed representation (typical dimension is 300).

Outline

FastText

["Bag of tricks for efficient text classification", Joulin et al. 2016]

["Enriching word vectors with subword information", Bojanowski et al. 2017]

["Learning word vectors for 157 languages", Grave et al. 2018]

For $n = 3$, the word "where" is represented by the set of its 3-grams:

$$<\text{wh, whe, her, ere, re}>$$

and the word itself, that is the sequence $<$where$>$.

For any word w, denote by $\mathcal{G}_w$ the set of n-grams (up to 6-grams) appearing in w.

A word w is then represented by a weighted sum of all n-grams it contains, that is the scoring function

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^T \mathbf{v}_c,$$

where $\mathbf{v}_c$ is the context vector.

Memory concern: Only a fixed number of n-grams is allowed namely $K = 2.10^6$.

Main idea: Use n-grams instead of words to improve accuracy. Very fast implementation.

Outline

ELMo (Embeddings from Language Models)

["Deep contextualized word representations", Peters et al. 2018]

Change of objective: modelling the probability of having a word given the previous ones:

$$p(w_k | w_{k-1}, \ldots, w_1)$$

Use of a **bidirectional two-layers LSTM** structure to model a word in a sentence given the previous ones and the past ones.

Taking into account the possible change of meaning of a word given its context: not a single vector per word!

For supervised tasks, run the previous network and extract all layer representations. Then learn a linear combination of these layers.

Outline

To go further

- Attention Mechanism
  https://jalammar.github.io/
  visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with
- Fundamental paper on attention
  [“Generating sequences with recurrent neural networks”, Graves 2013]
  [“Neural machine translation by jointly learning to align and translate”, Bahdanau et al. 2014]
- Transformers
  [“Attention is all you need”, Vaswani et al. 2017]
  http://jalammar.github.io/illustrated-transformer/
- BERT (state of the art)
  [“Bert: Pre-training of deep bidirectional transformers for language understanding”, Devlin et al. 2018]
  http://jalammar.github.io/illustrated-bert/

Let us play with Word2Vec:

http://projector.tensorflow.org/

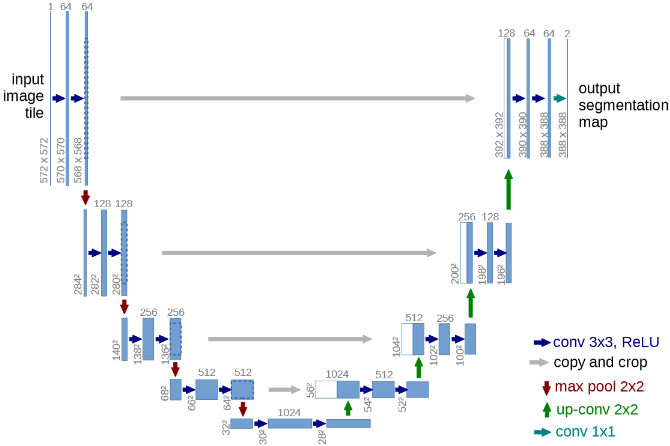
More references:

[“Universal language model fine-tuning for text classification”, Howard and Ruder 2018]

[“Improving language understanding by generative pre-training”, Radford et al. 2018]

Embeddings for images

["U-net: Convolutional networks for biomedical image segmentation", Ronneberger et al. 2015]



U-net

- Data augmentation : elastic deformation
- Upsampling: repeat each entry 4 times to double the size of the image
- Loss function: weighted pixel-wise cross-entropy computed on the last layer (softmax)

$$L(\theta) = \sum_{x \in \Omega} w(x) \log(p_{\ell(x),\theta}(x)),$$

where $\ell(x)$ is the true label of the pixel x and w is the weight function defined as

$$w(x) = w_c(x) + w_0 \exp\Big(- \frac{(d_1(x) + d_2(x))^2}{2\sigma^2} \Big),$$

with w_c is the weight map to balance the class frequencies, d_1 is the distance to the boarder of the nearest cell and d_2 the distance to the boarder of the second nearest cell ($w_0 = 10, \sigma = 5$).

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

Joeran Beel et al. "paper recommender systems: a literature survey". In: *International Journal on Digital Libraries* 17.4 (2016), pp. 305–338.

Piotr Bojanowski et al. "Enriching word vectors with subword information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

Edouard Grave et al. "Learning word vectors for 157 languages". In: *arXiv preprint arXiv:1802.06893* (2018).

Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

Jeremy Howard and Sebastian Ruder. "Universal language model fine-tuning for text classification". In: *arXiv preprint arXiv:1801.06146* (2018).

Armand Joulin et al. "Bag of tricks for efficient text classification". In: *arXiv preprint arXiv:1607.01759* (2016).

Hans Peter Luhn. "A statistical approach to mechanized encoding and searching of literary information". In: *IBM Journal of research and development* 1.4 (1957), pp. 309–317.

Tomas Mikolov, Kai Chen, et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

Tomas Mikolov, Ilya Sutskever, et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

Matthew E Peters et al. "Deep contextualized word representations". In: *arXiv preprint arXiv:1802.05365* (2018).

Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

Alec Radford et al. "Improving language understanding by generative pre-training". In: *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf* (2018).

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

Karen Sparck Jones. "A statistical interpretation of term specificity and its application in retrieval". In: *Journal of documentation* 28.1 (1972), pp. 11–21.

Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.