Static semantic words/documents models

Nicolas.Hernandez@univ-nantes.fr

ATAL - 31 octobre 2024

Nantes Université

LS2N Laboratoire des Sciences du Numérique de Nantes

TALN

Overview

1. Semantic space and similarity measures
2. Sparse models to represent words and documents semantics
3. Dense models and static representations

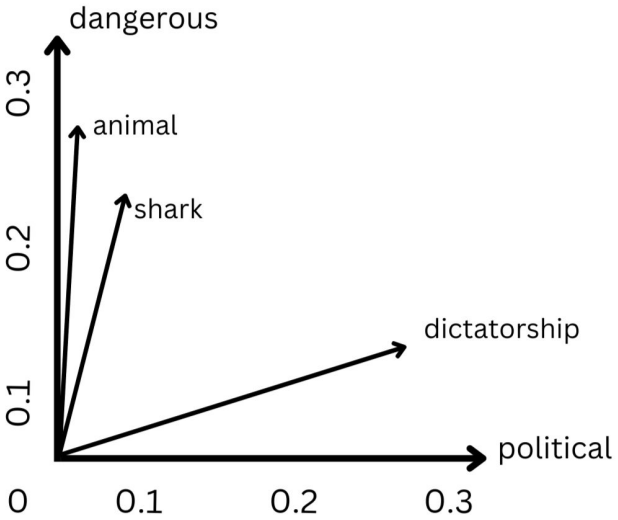
Semantic spaces and similarity measures

Semantic space

Mathematically represented as a **vector space**

With words, sentences, documents as **vector semantics** also called **embeddings***

Allow to **calculate semantic similarity** between a pair of words, sentences or documents embeddings **through the cosine metric**



*sometimes more strictly applied only to dense vectors

Measuring similarity: dot product

Given two vectors, v and w with same dimensionality |V|

dot product operator from linear algebra (also called **scalar product** or **inner product**) can be used to measure vectors similarity:

$$\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

But favors longer vectors and with higher values in each dimension

Measuring similarity: normalized dot product aka cosine

Normalized dot product using the product of the vector lengths to normalize

Similar to the cosine of the angle between the two vectors

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta$$

$$\frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = \cos\theta$$

Can be computed as

$$\text{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

Raw frequency values are non-negative, the cosine ranges from 0–1, from 1 for vectors pointing in the same direction, through 0 for orthogonal vectors

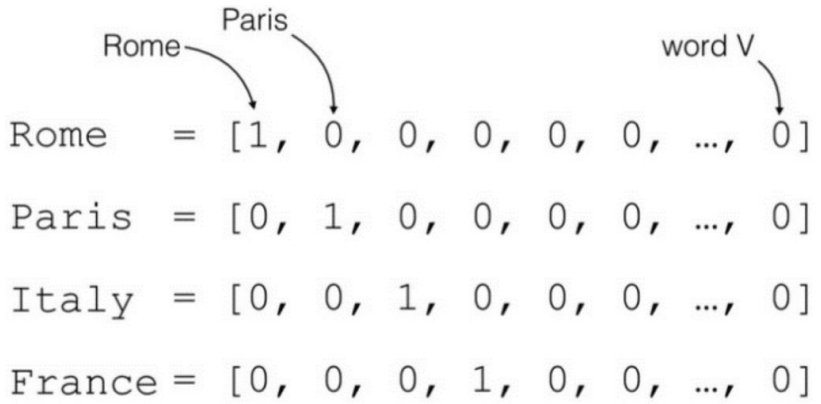
Words as vectors: word dimensions

One-hot encoding

Approach: describing words through vectors, one dimension per vocabulary entry (word), **each word with 1 in its corresponding column else 0**

Allow to differentiate words

But same similarity distance between any pair of words (zero)



Distributional Hypothesis

Basis for **Statistical Semantics** coming from linguists

- "*a word is characterized by the company it keeps*" (Firth, 1957)
- "*words that occur in the same contexts tend to have similar meanings*" (Harris, 1954)
- Implicit in (Weaver, 1955)

Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015) with colors added for explanation.

Co-occurrence vectors weighted by number of occurrence

Commonly called the **term-term** matrix or the **term-context** matrix of dimensionality |V|x|V| with V as Vocabulary

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

(Jurasky & Martin 2021)

Co-occurrence vectors weighted by the mutual information

Positive Pointwise Mutual Information (PPMI) ~ to determine whether two words, w_1 and w_2, co-occurs more often than "by chance" in a corpus (Church and Hanks, 1990); also called *association score*

Probability that w occurs, P(w), can be calculated by maximum likelihood estimation

Negative score means "which occurs less than chance » ; "positive" PMI suppresses this effect

It is possible to artificially increase the number of occurrences to solve bias of extremely rare words with high scores cf. (Levy et al. 2015), Laplace smoothing

$$PMI(w_1, w_2) = \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

$$P(w) = \frac{count(w)}{totalWordCount}$$

$$P(w_1, w_2) = \frac{count_{cooccurence}(w_1, w_2)}{totalWordCount}$$

$$PPMI = \begin{cases} PMI & if\ PMI > 0 \\ 0 & else \end{cases}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

PPMI	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

(Jurasky & Martin 2021)

Documents as vectors

Document representation as a *bag-of-words* (bow)

Approach: **a document in a collection is represented by the words it contains from the collection vocabulary**

The **term-document matrix** was proposed by (Salton et al., 1975) to model documents in vectorial space in Information Retrieval

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.3 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.

Note: a word can also be represented by the documents in which it occurs

From binary word presence to weighted variants, tf

Raw count as the **term frequency*** (Luhn, 1957):

$$\text{tf}_{t,d} = \text{count}(t,d)$$

*In statistics, frequency means the number of times an event occurs and not the quotient of the number of times a periodic event occurs over the time t

To mitigate the raw frequency, we use:

$$\text{tf}_{t,d} = \begin{cases} 1 + \log_{10} \text{count}(t,d) & \text{if } \text{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

(a word appearing 100 times in a document doesn't make that word 100 times more likely to be relevant to the meaning of the document)

From binary word presence to weighted variants, tf x idf

> "***Term specificity*** *in a text can be calculated by taking into consideration the* ***« physical» presence of the word in a text*** *with the* ***weight of its « general » importance in a corpus***" (Karen Spärck Jones, 1972)

The **tf-idf** weighted value $w_{t,d}$ for word t in document d combines $\mathbf{tf_{t,d}}$ with **idf**

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

The **inverse document frequency** emphasizes discriminative words

$$\mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right)$$

- **N total number of documents in collection**
- The **document frequency** $\mathbf{df_t}$** of term t, number of documents in which t occurs
- Because of the large number of documents, we also use a log function

** **collection frequency** of a term is the total number of times the word appears in the whole collection in any document

From words to … sentences, paragraphs, documents

- Words represented by vectors with document dimensions (term-document matrix) or word dim. (term-term)
- Values in each dimension can be counts, weighted by tf-idf or PPMI

A document can be represented by taking the vectors of all the words in the document and computing the **centroid** (multidimensional version of the mean)

Given k word vectors w_1, w_2, ..., w_k, the centroid document vector d is

$$d = \frac{w_1 + w_2 + \ldots + w_k}{k}$$

The centroid of a set of vectors is a single vector that has the minimum sum of squared distances to each of the vectors in the set

Issues

- high number of dimensions (vocabulary size or collection size)
- matrix is sparse (many zero inside)
- word order is not captured

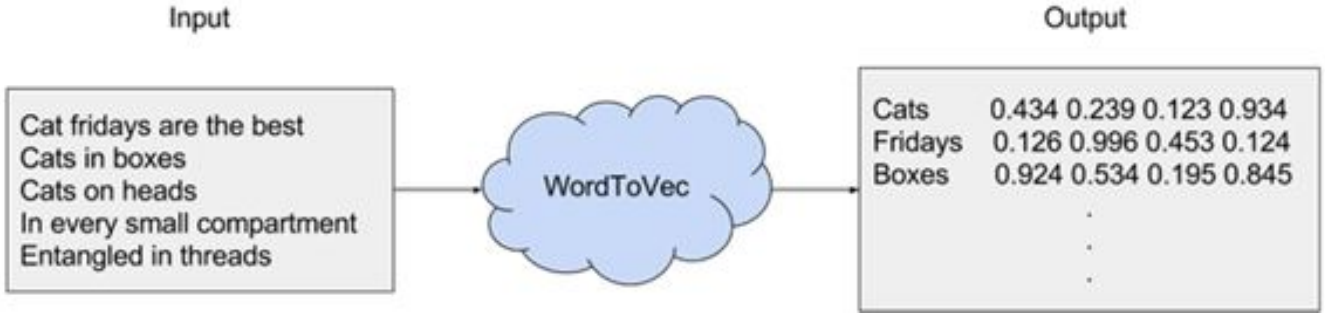
Need for methods to

- **reduce the number of dimensions and make the matrices more dense** in order to optimize the memory use and the computation
- **capture more linguistic features** (word order, meaning in context…)

Short dense words representation (Non-Contextual Embeddings) : word2vec, GloVe, fastText

Word embedding algorithms (Mikolov et al.@Google, 2013ab)

- Map words in short dense vectors of real values (low dimensions, no zero)



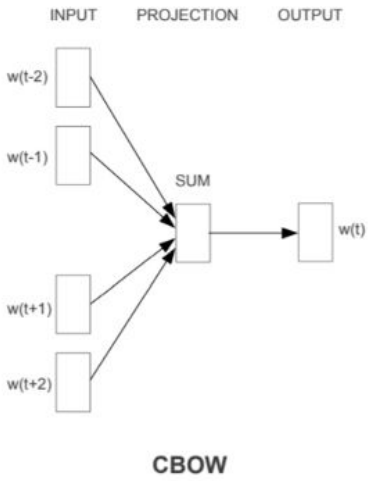
- Fast, efficient to train, code referred as « word2vec » (w2v) and pretrained embeddings available online
- (2013a) presents **skip-gram** and **continuous bag of words (CBOW)** ;
  - CBOW architecture similar to the probabilistic feedforward neural network language model (NNLM) proposed by (Bengio et al., 2003)
  - (2013b) give more details on skip-gram

Word embedding algorithms (Mikolov et al.@Google, 2013ab)

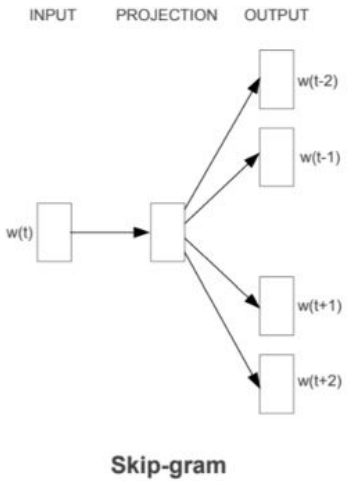
CBOW architecture similar to the probabilistic feedforward neural network language model (NNLM) proposed by (Bengio et al., 2003)

(2013b) give more details on skip-gram; w2v may mean this algorithm

predicts the current word based on the context



CBOW



Skip-gram

predicts surrounding words given the current word

Intuition of skip-gram

- A **context word c** is likely to occur near the **target word w**
  if its **embedding** vector is **similar** to the target **embedding**”

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
              c1          c2    w     c3       c4
```

- Train a **classifier to predict this probability** on embedding similarity
- Use the **learned weights as the word embeddings**
- **Self-supervision** (Bengio et al., 2003): treat the target word and a neighboring context word as **positive examples**
- Randomly sample **other words in the lexicon to get negative** samples

skip-gram detail 1

Use the **dot product** to compute the similarity between context and word vectors

$$Similarity(w,c) \approx \mathbf{c} \cdot \mathbf{w}$$

And use the **logistic/sigmoid function,** phi, to turn it into a **probability**

$$P(+|w,c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c} \cdot \mathbf{w})}$$

$$\begin{aligned} P(-|w,c) &= 1 - P(+|w,c) \\ &= \sigma(-\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1+\exp(\mathbf{c} \cdot \mathbf{w})} \end{aligned}$$

Much simpler than counting co-occurrence or predicting words

Training a binary logistic regression demand less sophisticated training algorithms than a multi-layer neural network with hidden layers

skip-gram detail 2

Considering a context of L words c in a window 1 to L and assuming their respective independency, simply multiply their probabilities:

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(\mathbf{c_i} \cdot \mathbf{w})$$

$$\log P(+|w, c_{1:L}) = \sum_{i=1}^{L} \log \sigma(\mathbf{c_i} \cdot \mathbf{w})$$

For each word, stores two embeddings, one for the word as a target, and one for the word considered as context

W and C matrices, each containing an embedding for every one of the |V| words in the vocabulary V, are the parameters to learn

Learning skip-gram embeddings

- select c, context window size
- select k, ratio of number of negative examples wrt to positives ones (choose negative words thanks to a weighted unigram frequency to give rare noise words slightly higher probabilities)

1. Assign a random embedding vector for each of the N vocabulary words
2. Iteratively shift the embedding of each word w to minimize the loss function L

$$
\begin{aligned}
L &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos})+\sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos})+\sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log\sigma(c_{pos}\cdot w)+\sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

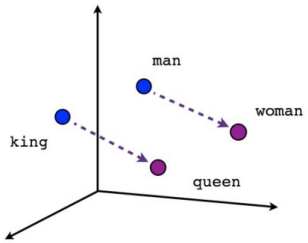
First term, assign the real context word c_{pos} a high probability of being a neighbor,
and second term, assign each of the noise words c_{neg_i} a high probability of being a non-neighbor

(all multiplied because we assume independence).

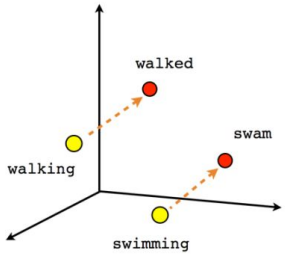
I.e. maximize the dot product of the word with the actual context words, and minimize the dot products of the word with the k negative sampled non-neighbor words

Word embedding algorithms (Mikolov et al.@Google, 2013ab)

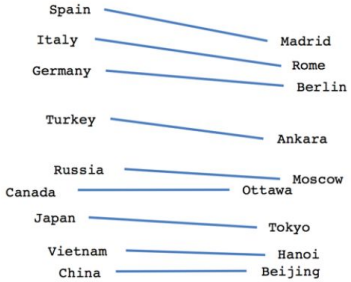
- CBOW < Skip-gram
  - CBOW faster to train, slightly better accuracy on frequent words and capture of syntactic relationships
  - Skip-gram good performance with little training data, especially rare words, better at capturing semantic
- « Related » words (e.g. synonyms) are effectively closed in the vector space



Male-Female



Verb tense



Country-Capital

- Non-contextual word representation: always the same meaning whatever the context
- Embeddings don’t just reflect the statistics of their input, but also amplify bias

GloVe, Word Embedding with Global Vectors (Pennington et al., 2014)

- Problem : word2vec observes only the local context of words
- Approach : GloVe **pre-calculates global statistics on words relations at the corpus level** then injects this knowledge when building the embeddings
- It considers the probability of words co-occurring by calculating the probability of occurrence ratio

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

‘solid’ is more likely to co-occur with “ice” than with “steam”, but “gas” is more likely to co-occur with “steam” than with “ice”

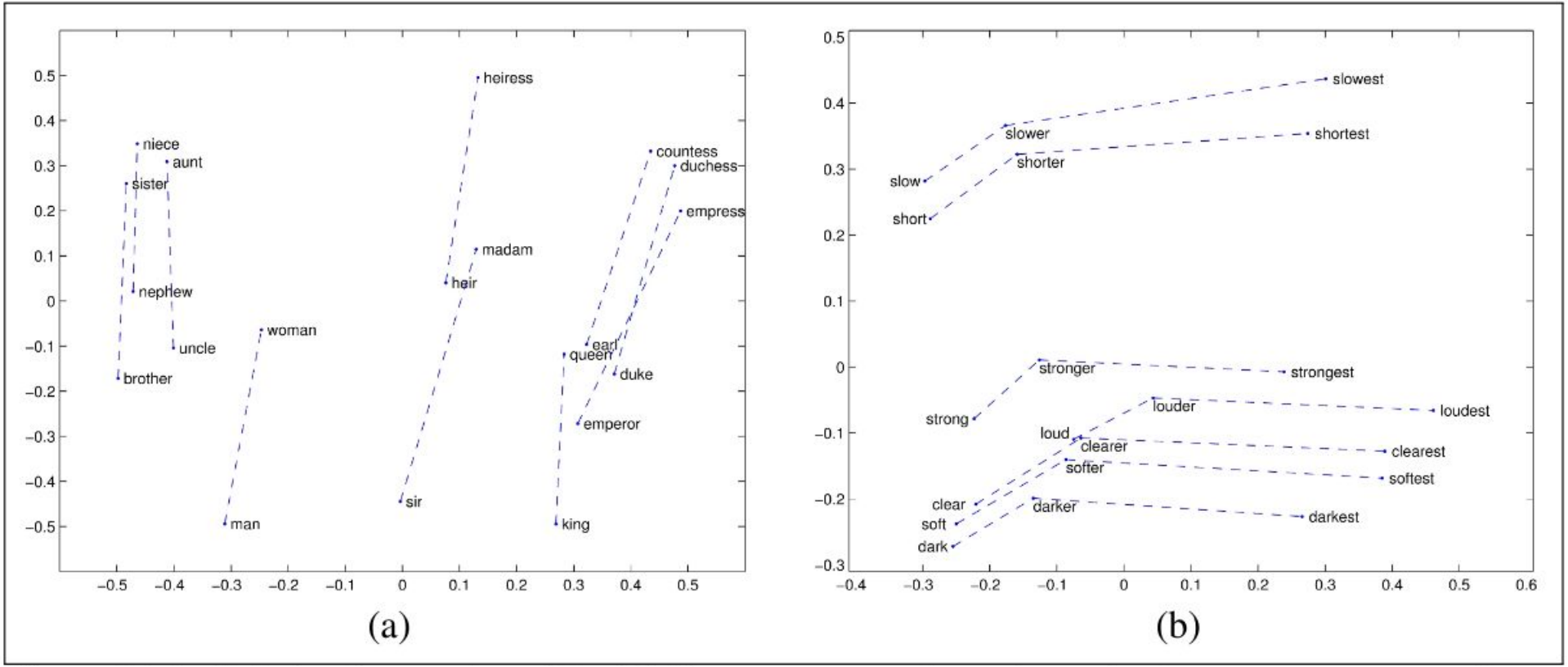
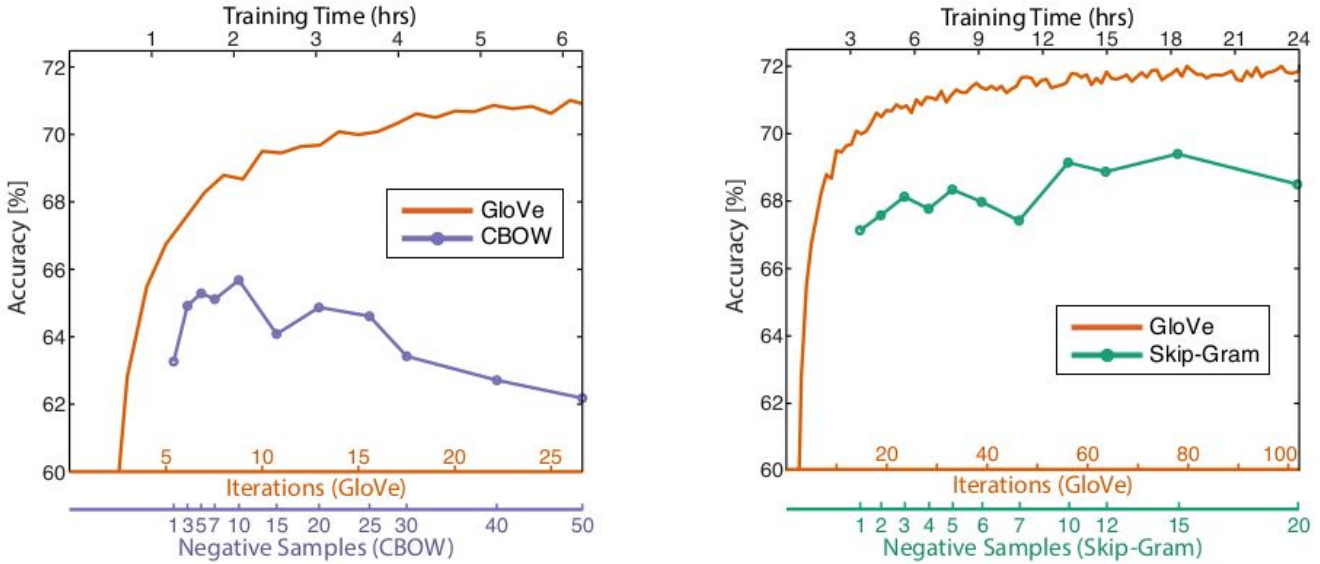


Figure 6.16 Relational properties of the GloVe vector space, shown by projecting vectors onto two dimensions. (a) $(\overrightarrow{\text{king}}) - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}$ is close to $\overrightarrow{\text{queen}}$. (b) offsets seem to capture comparative and superlative morphology (Pennington et al., 2014).

GloVe outperforms Word2Vec on word similarity tasks and Named Entity Recognition (NER)



(a) GloVe vs CBOW

(b) GloVe vs Skip-Gram

Figure 4: Overall accuracy on the word analogy task as a function of training time, which is governed by the number of iterations for GloVe and by the number of negative samples for CBOW (a) and skip-gram (b). In all cases, we train 300-dimensional vectors on the same 6B token corpus (Wikipedia 2014 + Gigaword 5) with the same 400,000 word vocabulary, and use a symmetric context window of size 10.

Subword Embedding avec *fastText* (Bojanowski et al., 2017)

- Problem : word2vec and GloVe does not use morphological information
  - Inflexion forms of a given word are represented by distinct vectors without sharing any parameters (e.g. "help", "helps", "helped")
  - Words with the same derivation forms does not share any parameter neither (e.g. the relation between "efficient" and "efficiently" is the same than "slow" and "slowly")
- Approach : fastText, word2vec extension, based on subwords **embeddings (i.e. character n-grams)** (Bojanowski et al., 2017)
  - 'helping' with n=3 gives '<he', 'hel', elp', 'lpi', 'ing', ng>
  - a central word is then represented by summing the subwords vectors
- ⇨**improves representation of rare words and out-of-vocabulary words**
- Open source library, including **pre-trained word vectors for 157 languages** https://fasttext.cc

Doc2vec

TODO, read (Le and Mikolov, 2014) + search for variants

Word embeddings vs Language Model

Problem :

« A word can have different meaning depending on its occurring context » …

Summary

Semantics representations of words, sentences, documents as vectors, a point in high-dimensional space, also called **embeddings**

Focus here on **static** representations (e.g. each word is mapped to a a fixed embeddings) vs… contextual

Sparse models i.e. mostly zeros (since most words simply never occur in the context of others)

- whose dimensionality can be high (e.g. corpus, vocabulary size)
- Various weighting functions one **hot vector, tf-idf, PMI**

vs. **dense** models

- short with a low dimensionality (50-1000)
- **Word2vec'** skip-gram algorithm, **FastText**, **Glove**

Similarities computed by some function of the **dot product** between vectors with the **cosine**, a normalised dot product, the most popular

References

NLP Compendium

- Jurafsky, Dan and Martin, James H. 2024. Speech and Language Processing, 3rd ed. draft, Aug. 20.

References: Sparse high-dimensional vectors

- Firth, J.R. 1957. A synopsis of linguistic theory 1930-1955. In Studies in Linguistic Analysis, pp. 1-32. Oxford: Philological Society. Reprinted in F.R. Palmer (ed.), Selected Papers of J.R. Firth 1952-1959, London: Longman (1968).
- Harris, Z. 1954. Distributional structure. Word, 10(23): 146-162.
- Weaver, W. 1955. Translation. In W.N. Locke and D.A. Booth (eds.), Machine Translation of Languages, Cambridge, MA: MIT Press.
- Salton, Gérard. 1971. The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice Hall
- Salton Gérard, Wong A., Yang. C. S. 1975. A vector space model for automatic indexing, Communications of the ACM, v.18 n.11, p. 613-620, novembre
- Sparck Jones, K. 1972. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28(1):11–21.
- Luhn, H. P. 1957. A statistical approach to the mechanized encoding and searching of literary information. IBM Journal of Research and Development, 1(4):309–317.
- Kenneth Ward Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. Computational Linguistics, 16(1):22–29.

Short-dense vectors

- Tomas Mikolov Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S. et Dean, Jeff, « Efficient Estimation of Word Representations in Vector Space », arXiv 1301.3781, 2013a
- Tomas Mikolov Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S. et Dean, Jeff, « Distributed representations of words and phrases and their compositionality », Advances in Neural Information Processing Systems, arXiv 1310.4546, 2013b
- Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155, 2003
- Bengio, Y., A. Courville, and P. Vincent. 2013. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798–1828Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov. 2017. Enriching word vectors with subword information. TACL, 5:135–146. https://fasttext.cc
- Pennington, J., R. Socher, and C. D. Manning. 2014. GloVe: Global vectors for word representation. EMNLP
- Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14). JMLR.org, II–1188–II–1196.