Machine Learning for Computational Linguistics

Distributed representations

Çağrı Çöltekin

University of Tübingen
Seminar für Sprachwissenschaft

June 14, 2016

Representations of linguistic units

- Most ML methods we use depend on how we represent the objects of interest, such as
  - words, morphemes
  - sentences, phrases
  - letters, phonemes
  - documents
  - speakers, authors
  - …
- The way we represent these objects interacts with the ML methods used
- They also affect what can be learned

Symbolic representations

- A common way to represent words (and other units) is to treat them as individual symbols
  w_1 = 'cat', w_2 = 'dog', w_3 = 'book'
- The symbols do not include any information about the use or meaning of the words or their relation to each other
- They are useful in many NLP tasks, but distinctions between units and their relationships are categorical
  - 'cat' as different from 'dog' as it is from 'book'
  - The relationship between 'cat' and 'dog' is not different from 'story' and 'tale'
- Some of these can be extracted from conventional lexicons or WordNets, but they will still be categorical/hard distinctions
- The similarity/difference decisions are typically made based on hand-annotated data

Vector representations

- The idea is to represent the linguistic objects as vectors

$$\text{cat} = (0.1, 0.3, 0.5, \ldots, 0.4)$$
$$\text{dog} = (0.2, 0.3, 0.4, \ldots, 0.3)$$
$$\text{book} = (0.9, 0.1, 0.8, \ldots, 0.3)$$

- The (syntactic/semantic) differences between the words correspond to distances in the high-dimensional vector space the word vectors live
- Symbolic representations are equivalent to *1-of-K* or *one-hot* vectors

$$\text{cat} = (0, \ldots, 1, 0, 0, \ldots, 0)$$
$$\text{dog} = (0, \ldots, 0, 1, 0, \ldots, 0)$$
$$\text{book} = (0, \ldots, 0, 0, 1, \ldots, 0)$$

  The distances in symbolic/one-hot representation are not useful.

Where does the vector representations come from?

- The vectors are (almost certainly) learned from the data
- The idea goes back to,

  You shall know a word by the company it keeps.
  —Firth (1957)

- In practice, we make use of the contexts where the words appear to determine their representations
- The words that appear in similar contexts are mapped to similar representations
- Context varies from a small window of words around the target word to a complete document

How to calculate word vectors

- Typically we use unsupervised (or self-supervised) methods
- Common approaches:
  - Obtain global counts of words in each context, and use techniques like SVD to assign vectors: the words with high covariances are assigned to similar vectors (LSA/LSI)
  - Predict the words from their context (or the context from the target words), and update the vectors to minimize the prediction error (word2vec, GloVe, …)
  - Model each word as a mixture of latent variables (LDA)

A toy example

A four-sentence corpus with *bag of words* (BOW) model.

The corpus:

```
S1: She likes cats and dogs
S2: He likes dogs and cats
S3: She likes books
S4: He reads books
```

Term-document (sentence) matrix

	S1	S2	S3	S4
she	1	0	1	0
he	0	1	0	1
likes	1	1	1	0
reads	0	0	0	1
cats	1	1	0	0
dogs	1	1	0	0
books	0	0	1	1
and	1	1	0	0

A toy example

A four-sentence corpus with *bag of words* (BOW) model.

The corpus:

```
S1: She likes cats and dogs
S2: He likes dogs and cats
S3: She likes books
S4: He reads books
```

Term-term (left-context) matrix

	#	*she*	*he*	*likes*	*reads*	*cats*	*dogs*	*books*	*and*
she	2	0	0	0	0	0	0	0	0
he	2	0	0	0	0	0	0	0	0
likes	0	2	1	0	0	0	0	0	0
reads	0	0	1	0	0	0	0	0	0
cats	0	0	0	1	0	0	0	0	1
dogs	0	0	0	1	0	0	0	0	1
books	0	0	0	1	1	0	0	0	0
and	0	0	0	0	0	1	1	0	0

Term-document matrices

- The rows are about the terms: similar terms appear in similar contexts
- The columns are about the context: similar contexts contain similar words
- The term-context matrices are typically sparse and large

Term-document (sentence) matrix

	S1	S2	S3	S4
she	1	0	1	0
he	0	1	0	1
likes	1	1	1	0
reads	0	0	0	1
cats	1	1	0	0
dogs	1	1	0	0
books	0	0	1	1
and	1	1	0	0

SVD (again)

- Singular value decomposition is a well-known method in linear algebra
- An $n \times m$ (n terms m documents) term-document matrix X can be decomposed as

$$X = U \Sigma V^T$$

U is a $n \times r$ unitary matrix, where r is the rank of X ($r \leqslant \min(n, m)$). Columns of U are the eigenvectors of XX^T

Σ is a $r \times r$ diagonal matrix of singular values (square root of eigenvalues of XX^T and X^TX)

V^T is a $r \times m$ unitary matrix. Columns of V are the eigenvectors of X^TX

- One can consider U and V as PCA performed for reducing dimensionality of rows (terms) and columns (documents)

Truncated SVD

$$X = U \Sigma V^T$$

- Using eigenvectors (from U and V) that correspond to k largest singular values ($k < r$), allows reducing dimensionality of the data with minimum loss
- The approximation,

$$\hat{X} = U_k \Sigma_k V_k$$

results in the best approximation of X, such that $\|\hat{X} - X\|_F$ is minimum

Truncated SVD

$$X = U \Sigma V^T$$

- Using eigenvectors (from U and V) that correspond to k largest singular values ($k < r$), allows reducing dimensionality of the data with minimum loss
- The approximation,

$$\hat{X} = U_k \Sigma_k V_k$$

results in the best approximation of X, such that $\|\hat{X} - X\|_F$ is minimum

- Note that r may easily be millions (of words or contexts), while we choose k much smaller (at most a few hundreds)

Truncated SVD (2)

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,m} \\ x_{3,1} & x_{3,2} & x_{3,3} & \dots & x_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,m} \end{bmatrix} =$$

$$\begin{bmatrix} u_{1,1} & \dots & u_{1,k} \\ u_{2,1} & \dots & u_{2,k} \\ u_{3,1} & \dots & u_{3,k} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,k} \end{bmatrix} \times \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_k \end{bmatrix} \times \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k,1} & u_{k,2} & \dots & u_{n,m} \end{bmatrix}$$

Truncated SVD (2)

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,m} \\ x_{3,1} & x_{3,2} & x_{3,3} & \dots & x_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,m} \end{bmatrix} =$$

$$\begin{bmatrix} u_{1,1} & \dots & u_{1,k} \\ u_{2,1} & \dots & u_{2,k} \\ u_{3,1} & \dots & u_{3,k} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,k} \end{bmatrix} \times \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_k \end{bmatrix} \times \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k,1} & u_{k,2} & \dots & u_{n,m} \end{bmatrix}$$

The term$_1$ can be represented using the first row of U_k

Truncated SVD (2)

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,m} \\ x_{3,1} & x_{3,2} & x_{3,3} & \dots & x_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,m} \end{bmatrix} =$$

$$\begin{bmatrix} u_{1,1} & \dots & u_{1,k} \\ u_{2,1} & \dots & u_{2,k} \\ u_{3,1} & \dots & u_{3,k} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \dots & u_{n,k} \end{bmatrix} \times \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_k \end{bmatrix} \times \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k,1} & u_{k,2} & \dots & u_{n,m} \end{bmatrix}$$

The document$_1$ can be represented using the first column of V_k^T

Truncated SVD example

The corpus:

```
(S1) She likes cats and dogs
(S2) He likes dogs and cats
(S3) She likes books
(S4) He reads books
```

	S1	S2	S3	S4
she	1	0	1	0
he	0	1	0	1
likes	1	1	1	0
reads	0	0	0	1
cats	1	1	0	0
dogs	1	1	0	0
books	0	0	1	1
and	1	1	0	0

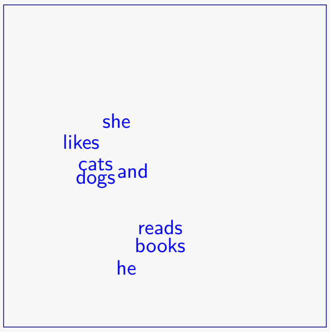
Truncated SVD ($k = 2$)

$$U = \begin{bmatrix} -0.30 & 0.28 \\ -0.24 & -0.63 \\ -0.52 & 0.15 \\ -0.03 & -0.49 \\ -0.43 & 0.01 \\ -0.43 & 0.01 \\ -0.03 & -0.49 \\ -0.43 & 0.01 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 3.11 & 0 \\ 0 & 1.81 \end{bmatrix}$$

$$V^T = \begin{array}{c} \begin{array}{cccc} S1 & S2 & S3 & S4 \end{array} \\ \begin{bmatrix} -0.68 & 0.26 & -0.11 & -0.66 \\ -0.66 & -0.23 & 0.48 & 0.50 \end{bmatrix} \end{array}$$

Truncated SVD (with BOW sentence context)



The corpus:
```
(S1) She likes cats and dogs
(S2) He likes dogs and cats
(S3) She likes books
(S4) He reads books
```


Truncated SVD (with single word context)

The corpus:
```
(S1) She likes cats and dogs
(S2) He likes dogs and cats
(S3) She likes books
(S4) He reads books
```


SVD: LSI/LSA

- SVD applied to term-document matrices are called
  - *Latent semantic analysis* (LSA) if the aim is constructing *term* vectors
  - *Latent semantic indexing* (LSI) if the aim is constructing *document* vectors
- The well known Google *PageRank* algorithm is a variation of the SVD

SVD based vectors: practical concerns

- In practice, instead of raw counts of terms within contexts, the term-document matrices typically contain
  - pointwise mutual information
  - tf-idf

  values.
- If the aim is finding latent (semantic) topics, frequent/syntactic words (*stopwords*) are often removed
- Depending on the measure used, it may also be important to normalize for the document length

SVD-based vectors: applications

- The SVD-based methods is commonly used in information retrieval
  - The system builds document vectors using SVD
  - The search terms are also considered as a 'document'
  - System retrieves the documents whose vectors are similar to the search term
- The SVD-based methods for semantic similarity is also common
- It was shown that the vector space models outperform humans in TOEFL synonym questions and SAT analogy questions

Predictive models

- Instead of dimensionality reduction through SVD, we try to predict
  - either the target word from the context
  - or the context given the target word
- We assign each word to a fixed-size random vector
- We use a standard ML model and try to reduce the prediction error with a method like gradient descent
- During learning, the algorithm optimizes the vectors as well as the model paramters
- In this context, the word-vectors are called embeddings
- This types of models has been very popular during the last few years

word2vec

- word2vec is a popular algorithm and open source application for training word vectors (Mikolov et al. 2013)
- It has two modes of operation

  CBOW or continuous bag of words predict the word using a window around the word

  Skip-gram does the reverse, it predicts the words in the context of the target word using the target word as the predictor
- The algorithm learns two sets of embeddings (one for context, one for target)
- The learning method is simply logistic regression, where word vectors are also updated (besides model parameters)
- Negative examples are sampled from the larger corpus
- It preforms well, and it is much faster than earlier (more complex) ANN architectures developed for this task

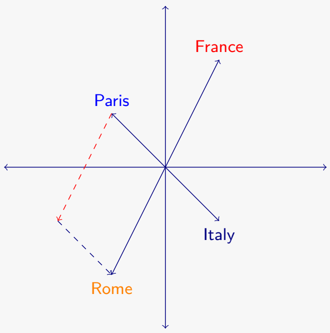
GloVe

- GloVe is another popular method for obtaining word vectors (Pennington, Socher, and Manning 2014)
- It tries to combine intuitions from both SVD-like 'counting' methods, and prediction-based methods
- It typically performs better on smaller data sets as well

Word vectors and syntactic/semantic relations

Word vectors map some syntactic/semantic relations to vector operations

- Paris - France + Italy = Rome
- king - man + woman = queen
- duck - ducks + mouse = mice



Using vector representations

- Dense vector representations are useful for many ML methods
- They are particularly suitable for neural network models
- 'General purpose' vectors can be trained on unlabeled data
- They can also be trained for a particular purpose, resulting in 'task specific' vectors
- Dense vector representations are not specific to words, they can be obtained and used for any (linguistic) object of interest

Evaluating vector representations

- Like other unsupervised methods, there are no 'correct' labels
- Evaluation can be based on
  - Intrinsic evaluation based on success on finding analogy/synonymy
  - Extrinsic evaluation, based on whether they improve a particular task (e.g., parsing, sentiment analysis) or not
  - Correlation with human judgments

Summary

- Dense vector representations of linguistic units (as opposed to symbolic representations) allow calculating similarity/difference between the units
- They can be either based on counting (SVD), or predicting (word2vec, GloVe)
- They are particularly suitable for ANNs, deep learning architectures

Next: practical exercises with word vectors. Make sure you have `word2vec` and/or `GloVe` installed by Thursday.

References

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781. URL: http://arxiv.org/abs/1301.3781.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global Vectors for Word Representation". In: *EMNLP*. Vol. 14, pp. 1532–1543.