92586 Computational Linguistics

Lesson 5. From Word Counts to *Meaning*

Alberto Barrón-Cedeño

Alma Mater Studiorum-Università di Bologna
a.barron@unibo.it @_albarron_

19/03/2020

Previously

- Pre-processing
- BoW representation
- One rule-based sentiment model
- One statistical model (Naïve Bayes)

Previously

- Pre-processing
- BoW representation
- One rule-based sentiment model
- One statistical model (Naïve Bayes)
- *tf-idf* (+ Zipf's law)

Table of Contents

Jumping from Chapter 3 to Chapter 4 of Lane et al. (2019)

tf-idf Tools

tf-idf Implementation

- We "hand-coded" the *tf-idf* implementation

[1] `http://scikit-learn.org/`. As usual, install it the first time; e.g., `pip install scipy; pip install sklearn`

tf-idf Implementation

- We "hand-coded" the *tf-idf* implementation
- Optimised and easy-to-use libraries exist

[1] `http://scikit-learn.org/`. As usual, install it the first time; e.g., `pip install scipy; pip install sklearn`

tf-idf Implementation

- We "hand-coded" the *tf-idf* implementation
- Optimised and easy-to-use libraries exist
- `scikit-learn` is a good alternative[1]

Let us see

[1] `http://scikit-learn.org/`. As usual, install it the first time; e.g., `pip install scipy; pip install sklearn`

tf-idf

Final Remarks

- *tf-idf*-like weighting is in the core of search engines and related technology

- *tf-idf*-like weighting is in the core of search engines and related technology
- Okapi BM25 has been one of the most successful ones (Robertson and Zaragoza, 2009)

Okapi	First system using BM25 (U. of London)
BM	best matching
25	Combination of BM11 and BM15

- *tf-idf*-like weighting is in the core of search engines and related technology
- Okapi BM25 has been one of the most successful ones (Robertson and Zaragoza, 2009)

Okapi	First system using BM25 (U. of London)
BM	best matching
25	Combination of BM11 and BM15

- Cosine similarity is a top choice distance for most text vector representations.

- *tf-idf*-like weighting is in the core of search engines and related technology
- Okapi BM25 has been one of the most successful ones (Robertson and Zaragoza, 2009)

Okapi	First system using BM25 (U. of London)
BM	best matching
25	Combination of BM11 and BM15

- Cosine similarity is a top choice distance for most text vector representations.
- Nothing prevents you from weighting n-grams, for $n = [1, 2, \ldots]$

Topic Vectors

Topic Vectors

What for?

"[. . .] using the correlation of normalized frequencies with each other to group words together in topics to define the dimensions of new topic vectors." (Lane et al., 2019, p. 98)

Topic Vectors

What for?

"[. . .] using the correlation of normalized frequencies with each other to group words together in topics to define the dimensions of new topic vectors." (Lane et al., 2019, p. 98)

What can we achieve with this?

- Compare texts on the basis of meaning (not keywords)
- Search based on meaning

Topic Vectors

What for?

“[. . .] using the correlation of normalized frequencies with each other to group words together in topics to define the dimensions of new topic vectors.” (Lane et al., 2019, p. 98)

What can we achieve with this?

- Compare texts on the basis of meaning (not keywords)
- Search based on meaning
- Represent the subject of a statement/document or corpus
- Extract keywords

Topic Vectors

Limitation of word vectors

d_1	Una macchinna rossa
d_2	Le macchinne blu

Topic Vectors

Limitation of word vectors

d_1	Una macchinna rossa
d_2	Le macchinne blu
	$\downarrow$
d'_1	macchinn ross
d'_2	macchinn blu

Topic Vectors

Limitation of word vectors

$$
\begin{array}{ll}
d_1 & \text{Una macchinna rossa} \\
d_2 & \text{Le macchinne blu} \\
 & \downarrow \\
d_1' & \text{macchinn ross} \\
d_2' & \text{macchinn blu} \\
 & \downarrow \\
\vec{d_1} & [\mathbf{1}, 1, 0] \\
\vec{d_2} & [\mathbf{1}, 0, 1]
\end{array}
$$

Topic Vectors

Limitation of word vectors

d_1	Una macchinna rossa
d_2	Le macchinne blu
	$\downarrow$
d'_1	macchinn ross
d'_2	macchinn blu
	$\downarrow$
$\vec{d_1}$	$[\mathbf{1}, 1, 0]$
$\vec{d_2}$	$[\mathbf{1}, 0, 1]$

$$cos(d_1, d_2) > 0$$

Topic Vectors

Limitation of word vectors

d_1	Un'automobile rosso
d_2	Le macchinne blu

Topic Vectors

Limitation of word vectors

d_1	Un'automobile rosso
d_2	Le macchinne blu
	$\downarrow$
d'_1	automob ross
d'_2	macchinn blu

Topic Vectors

Limitation of word vectors

d_1	Un'automobile rosso
d_2	Le macchinne blu
	$\downarrow$
d_1'	automob ross
d_2'	macchinn blu
	$\downarrow$
$\vec{d_1}$	$[1, 1, 0, 0]$
$\vec{d_2}$	$[0, 0, 1, 1]$

Topic Vectors

Limitation of word vectors

d_1 Un'automobile rosso
d_2 Le macchinne blu

$\downarrow$

d'_1 automob ross
d'_2 macchinn blu

$\downarrow$

$\vec{d_1}$ $[1, 1, 0, 0]$
$\vec{d_2}$ $[0, 0, 1, 1]$

$$cos(d_1, d_2) = 0$$

Topic Vectors

- We need to infer what $w \in d$ "signifies"
- Indeed, we need to infer what $\{w_k, w_{k+1}, \ldots\} \in d$ "signify"

Topic Vectors

- We need to infer what $w \in d$ "signifies"
- Indeed, we need to infer what $\{w_k, w_{k+1}, \ldots\} \in d$ "signify"
- We can achieve it with a different kind of vector

Topic Vectors

- We need to infer what $w \in d$ "signifies"
- Indeed, we need to infer what $\{w_k, w_{k+1}, \ldots\} \in d$ "signify"
- We can achieve it with a different kind of vector

Word-topic vector One vector represents one word

Document-topic vector One vector represents one document (by combining its word-topic vectors)

Topic Vectors

- We need to infer what $w \in d$ "signifies"
- Indeed, we need to infer what $\{w_k, w_{k+1}, \ldots\} \in d$ "signify"
- We can achieve it with a different kind of vector

Word-topic vector: One vector represents one word

Document-topic vector: One vector represents one document (by combining its word-topic vectors)

These models can deal with polysemy (e.g., homonyms) at some extent

Common-Sense Topic Modeling

Scenario

- We are processing sentences about pets, Central Park, and New York
- Three topics: `petness`, `animalness`, `cityness`

Example from (Lane et al., 2019, p. 101–102)

Common-Sense Topic Modeling

Scenario

- We are processing sentences about pets, Central Park, and New York
- Three topics: `petness`, `animalness`, `cityness`

- `cat` and `dog` should contribute similarly to `petness`
- `NYC` should contribute negatively to `animalness`
- `apple` should contribute mildly to `cityness`

Example from (Lane et al., 2019, p. 101–102)

Common-Sense Topic Modeling

Scenario

- We are processing sentences about pets, Central Park, and New York
- Three topics: `petness`, `animalness`, `cityness`

- `cat` and `dog` should contribute similarly to `petness`
- `NYC` should contribute negatively to `animalness`
- `apple` should contribute mildly to `cityness`

	score		
topic	**high**	**medium**	**low**
Petness	cat, dog		NYC, apple
Cityness	NYC	apple	cat, dog

Example from (Lane et al., 2019, p. 101–102)

Common-Sense Topic Modeling

Scenario

- We are processing sentences about pets, Central Park, and New York
- Three topics: `petness`, `animalness`, `cityness`

- `cat` and `dog` should contribute similarly to `petness`
- `NYC` should contribute negatively to `animalness`
- `apple` should contribute mildly to `cityness`

	score		
topic	**high**	**medium**	**low**
Petness	cat, dog		NYC, apple
Cityness	NYC	apple	cat, dog

Let us see

Example from (Lane et al., 2019, p. 101–102)

Common-Sense Topic Modeling

- The relationships between words and topics can be "flipped"
- We transpose the 3×6 matrix (3 topic vectors) to produce topic weights for each word

Common-Sense Topic Modeling

- The relationships between words and topics can be "flipped"
- We transpose the 3×6 matrix (3 topic vectors) to produce topic weights for each word

- We have a 6-dimensional *tf-idf* vector
- We multiply it by a 3×6 weights matrix

Common-Sense Topic Modeling

- The relationships between words and topics can be “flipped”
- We transpose the 3×6 matrix (3 topic vectors) to produce topic weights for each word

- We have a 6-dimensional *tf-idf* vector
- We multiply it by a 3×6 weights matrix
- We end up with a 3D document vector

Common-Sense Topic Modeling

- The relationships between words and topics can be "flipped"
- We transpose the 3×6 matrix (3 topic vectors) to produce topic weights for each word

- We have a 6-dimensional *tf-idf* vector
- We multiply it by a 3×6 weights matrix
- We end up with a 3D document vector

Let us see

Common-Sense Topic Modeling

- The relationships between words and topics can be "flipped"
- We transpose the 3×6 matrix (3 topic vectors) to produce topic weights for each word

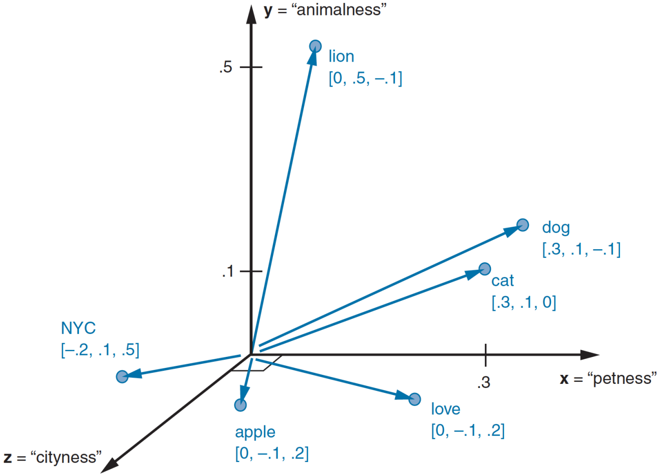
- We have a 6-dimensional *tf-idf* vector
- We multiply it by a 3×6 weights matrix
- We end up with a 3D document vector

Let us see

Advantages

- We can visualise 3D vectors
- A 3D vector space is convenient for classification: it can be sliced with a hyperplane to divide it into classes

Common-Sense Topic Modeling



Borrowed from (Lane et al., 2019, p. 104)

Common-Sense Topic Modeling

In summary. . .

$\vec{d}$ a *tf-idf* vector of size $|V|$

Common-Sense Topic Modeling

In summary...

$\vec{d}$ a *tf-idf* vector of size $|V|$

M a $3 \times V$ weight matrix

Common-Sense Topic Modeling

In summary. . .

$\vec{d}$ a *tf-idf* vector of size $|V|$

M a $3 \times V$ weight matrix

$\downarrow$

$\vec{d_t}$ a topic vector of size 3

From one vector space to another

high-dimensional *tf-idf* space $\rightarrow$ low-dimensional **topic** vector space

Common-Sense Topic Modeling

In summary. . .

$$\begin{array}{rl} \vec{d} & \text{a } \textit{tf-idf} \text{ vector of size } |V| \\ M & \text{a } 3 \times V \text{ weight matrix} \\ & \downarrow \\ \vec{d_t} & \text{a topic vector of size 3} \end{array}$$

From one vector space to another

high-dimensional *tf-idf* space $\rightarrow$ low-dimensional **topic** vector space

How can we **learn** the "transformation" matrix?

Towards a Topic Space

You shall know a word by the company it keeps

J. R. Firth (1957)

Towards a Topic Space

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

- We have corpora
- We have pre-processors
- We can produce *tf-idf* matrices

Towards a Topic Space

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

- We have corpora
- We have pre-processors
- We can produce *tf-idf* matrices

We can count co-occurrences → the company of a word

Latent Semantic Analysis

Latent Semantic Analysis

- An algorithm to gather words (*tf-idf* matrix) into topics
- It captures meaning of words
- It is a dimension reduction technique

Latent Semantic Analysis

- An algorithm to gather words (*tf-idf* matrix) into topics
- It captures meaning of words
- It is a dimension reduction technique

AKA

- Principal Component Analysis (PCA)
- Latent Semantic Indexing (LSI, in IR)

Latent Semantic Analysis

Linear discriminant analysis (LDA)

A supervised algorithm (it needs labeled data!)

Latent Semantic Analysis

Linear discriminant analysis (LDA)

A supervised algorithm (it needs labeled data!)

Algorithm

```
1. Compute the centroid of the vectors in the class
2. Compute the centroid of the vectors not in the
class
3. Compute the vector difference between the centroids
```

Latent Semantic Analysis

Linear discriminant analysis (LDA)

A supervised algorithm (it needs labeled data!)

Algorithm

```
1. Compute the centroid of the vectors in the class
2. Compute the centroid of the vectors not in the
class
3. Compute the vector difference between the centroids
```

Centroid: average

Basic algebra!

Let us see

Latent Semantic Analysis

Linear discriminant analysis (LDA)

- We are not relying on individual words
- We are gathering up words with similar “semantics”

Latent Semantic Analysis

Linear discriminant analysis (LDA)

- We are not relying on individual words
- We are gathering up words with similar "semantics"

LDA has learned the `spaminess` of words and documents

Coming Next

- Training and Evaluation in Machine Learning
- More LSA (from 4.2, p 111)

References

Lane, H., C. Howard, and H. Hapkem
2019. *Natural Language Processing in Action*. Shelter Island, NY: Manning Publication Co.

Robertson, S. and H. Zaragoza
2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333—-389.