ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI FORLÌ

91258 / B0385
Natural Language Processing

Lesson 13. Hands on Word Embeddings

Alberto Barrón-Cedeño
a.barron@unibo.it

10/11/2025

Previously

- Skip-gram
- CBOW

Table of Contents

Chapter 6 of Lane et al. (2019)

Pre-Trained Models

Some Pre-Trained Models

Model	Provider	Description
word2vec	Google	300D from English Google News articles[1]
fastText	Facebook	157 languages from Wikipedia and Common Crawl[2]
word2vec/GloVe	CNR	Italian embeddings from the Wikipedia
word2vec	UCampania	Italian embeddings[3]

There are many pre-trained models and diverse libraries to handle them.

Just query your favorite search engine

[1] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM
[2] https://fasttext.cc
[3] https://mlunicampania.gitlab.io/italian-word2vec/

Gensim

Gensim

- Scalable, open source, and efficient Python library
- It includes many resources, including word2vec, doc2vec, FastText, LDA, and more
- All information, including (very nice) documentation at `https://radimrehurek.com/gensim/`

Let us see

Gensim

Most similar items

`word_vectors.most_similar()`

Among the most interesting arguments:

- **positive** list of vectors to be added together before looking for the neighbours
- **negative** subtraction (or exclusion) of the elements
- **topn** number of elements to retrieve

Let us see

Gensim
Least similar items (closed set)

`word_vectors.doesnt_match()`

It returns the element from the input list with the lowest similarity with respect to the rest

Let us see

Gensim
More operations

Adding and Subtracting
We can use `most_similar()` again, this time with the negative parameter

Let us see

Computing similarities
`word_vectors.similarity()`

Let us see

Gensim
Getting the Vectors

Gensim (and other libraries) have implemented these interfaces to perform some *standard* operations

To go beyond, one needs to get access to the actual vectors

`word_vectors[word]`

Let us see

Model Construction

Model Construction

Considerations

- If you are working in other language than English, Google's provided word2vec is not an option (FastText might be)
- Google's word2vec is built on news; fastText has versions built on the Wikipedia and on common crawl...
  analysing scientific papers or literature?
  Probably not
- You want to work on COVID-19 or any other recent topic?
  Many relevant terms wont appear

Alternatives

- Opting for some of the previous representations
- Getting a more up-to-date existing embedding space
- Build your own model

Model Construction

Pre-Processing

Typical pre-processing pipeline

- Tokenisation
- Lowercasing (optional)
- Sentence splitting

Input Embedded list of tokenised sentences

$$[[w_{0,0}\ w_{0,1}\ w_{0,2} \ldots w_{0,k}], [w_{1,0}\ w_{1,1}\ w_{1,2} \ldots w_{1,l}], \ldots [w_{x,0}\ w_{x,1} \ldots w_{x,m}]]$$

Model Construction

Training

Training a word2vec model with gensim

Tutorial: `https://rare-technologies.com/word2vec-tutorial/`

Let us see

Considerations

- Training on relatively large corpora might take some time
  (Brown is small and took me a bit less than 1 minute on a 2.5GHz Quad-Core i7, 16GB RAM)
- Large corpora (e.g., the Wikipedia) can require a significant amount of time/memory

Model Construction

Trimming and Saving

Reminder We do not care about the output

`model.init_sims(replace=True)`

- Freezes the model
- Stores the hidden-layer weights
- Discards the output-layer weights

not necessary since gensim 4.0

Now we simply have to save the model with `model.save()`
Let us see

GloVe

GloVe

Global Vectors (Pennington et al., 2014)[4]

- It uses a global word-word co-occurrence matrix
- Learning objective: word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence

- It produces similar matrices to word2vec
- It converges, even with smaller corpora
- It is more accurate with the same amount of data

[4] https://nlp.stanford.edu/projects/glove/

GloVe

GloVe vs word2vec

RaRe Technologies comparison[5]

Settings: 600 dims • context window of 10 • 1.9B words of *en* Wikipedia.

Algorithm	**acc (word analogy)***	**wallclock time**	**peak RAM (MB)**
I/O only	–	3m	25
GloVe, 10 epochs, lr 0.05	67.1	4h12m	9,414
GloVe, 100 epochs, lr 0.05	67.3	18h39m	9,452
word2vec, hierarchical skip-gram, 1 epoch	57.4	3h10m	266
word2vec, negative sampling (10 samples), 1 epoch	68.3	8h38m	628
word2vec, Google 300d	55.3	–	–

* *a* is to *b* as *c* is to ?

[5] rare-technologies.com/making-sense-of-Word2vec/#glove_vs_word2vec

fastText

fastText

Predicts the surrounding character [2, 3]-grams rather than the surrounding words (Bojanowski et al., 2017)[6]

- Pre-trained models available in 250+ languages
- Built on Wikipedia editions (variable quality)
- Built on common crawl

Models available at `https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md`

Let us see

[6] `https://github.com/facebookresearch/fastText`

Some Remarks

LSA is a better (faster) option for long documents e.g., for clustering

Online learning An existing model can be *adapted* (but new words cannot be added)

doc2vec possible representation based on linear combinations of word2vec

References

Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov
2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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

Pennington, J., R. Socherm, and C. Manning
2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, Pp. 1532–1543.