Text Mining and Search
Lab 1.3

Word Embeddings

Luca Herranz-Celotti

University of Milano-Bicocca, 2024

Instructions for Google Colab

Instructions for Google Colab

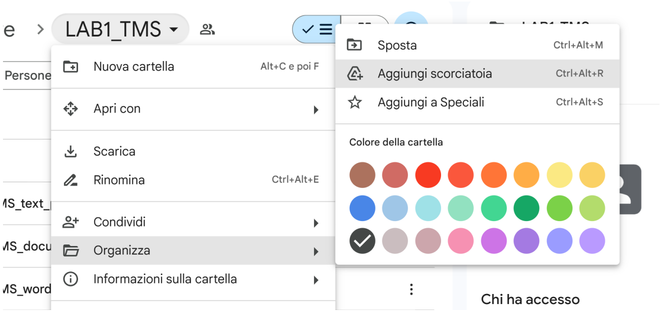
- Go to https://linktr.ee/lucarrtt

Instructions for Google Colab

- Go to https://linktr.ee/lucarrtt
- Select the ‘Tutorial 1 - Text Mining and Search 2024’ button

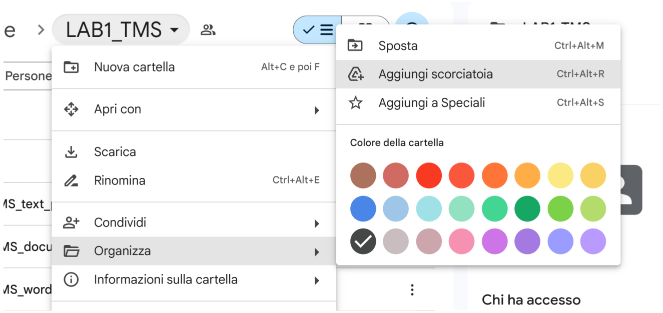
Instructions for Google Colab

- Go to https://linktr.ee/lucarrtt
- Select the 'Tutorial 1 - Text Mining and Search 2024' button
- Make a shortcut of the folder to your drive



Instructions for Google Colab

- Go to https://linktr.ee/lucarrtt
- Select the ‘Tutorial 1 - Text Mining and Search 2024’ button
- Make a shortcut of the folder to your drive



- Open the Notebook and save it on drive.

Introduction

Introduction and Purpose

Create a vector representation of a text.
But Why!?

Introduction and Purpose

Create a vector representation of a text.
But Why!?
Numbers are easier to work with! Moreover, you can use a variety of ML models created on vectors.

Introduction and Purpose

We have different ways of representing the knowledge:

Introduction and Purpose

We have different ways of representing the knowledge:

- Binary Vector Representation

Introduction and Purpose

We have different ways of representing the knowledge:

- Binary Vector Representation
- BOW (Bag-of-Words) Representation

Introduction and Purpose

We have different ways of representing the knowledge:

- Binary Vector Representation
- BOW (Bag-of-Words) Representation
- TF-IDF

Introduction and Purpose

It would be cool to capture the semantic information as well!

$$\text{Pen} \leftrightarrow \text{Writer} \sim \text{Brush} \leftrightarrow \text{Painter}$$

The basic representations fail to capture these relationships between words.

Word Embeddings

Embeddings

One hot encoding (and similar) creates a vector representation that is too sparse and it does not capture the relations between words.

Embeddings

One hot encoding (and similar) creates a vector representation that is too sparse and it does not capture the relations between words.

We want to «embed» the words in a lower dimension space while conserving its properties (like relationships).

Embeddings

So something like

$$[\text{Pen}] - [\text{Writer}] + [\text{Painer}] \sim [\text{Brush}]$$

should be true!

Word2Vec

Word2Vec

- Based on «Shallow» Neural Networks

Word2Vec

- Based on «Shallow» Neural Networks
- Groups similar words together based on co-occurrences

Word2Vec

- Based on «Shallow» Neural Networks
- Groups similar words together based on co-occurrences
- Possible Architectures:
  - Continuous Bag-of-Words (CBoW)
    Predict a word given the context (surrounding words)
  - continuous Skip-Gram
    Predict the context given a word

Word2Vec

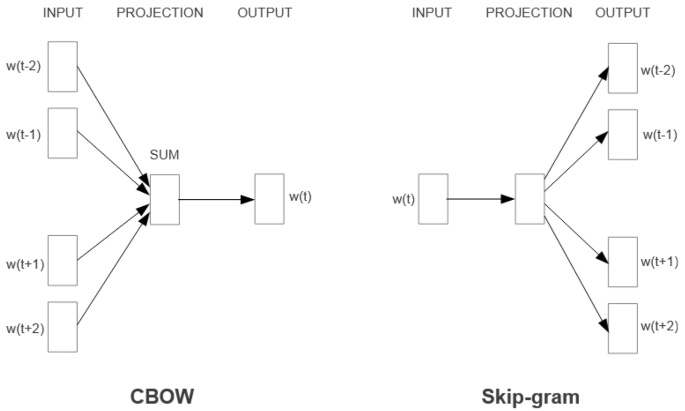


Figure: CBoW vs Skip-Gram.[1]

[1] Mikolov, Le, and Sutskever, *Exploiting Similarities among Languages for Machine Translation*.

Word2Vec

- In python:

```
from gensim.models import Word2Vec

# Train the model
model = Word2Vec(sentences=corpus, vector_size=100, window=5, min_count=1, workers=4)

# Save the model
model.save(path)
```

- sentences must be a re-startable iterator!
- vector_size is the dimension of the word space
- window is the context length

Word2Vec

To use a pre-trained model:

```
from gensim import downloader

# See all the pretrained models available:
print(list(gensim.downloader.info()['models'].keys()))

# load a pre-trained model
model = downloader.load('word2vec-google-news-300')

# train a model even further for 5 epochs
model.train(new_corpus, total_examples=50, epochs=5)
# get a word vector (embedding) in numpy
w_emb = model.wv['word']

# get top 10 similar words
sim_words = model.wv.most_similar('word2', topn=10)
```

GloVe

GloVe

- Brings back a more global approach to word embeddings.

GloVe

- Brings back a more global approach to word embeddings.
- Word2Vec sees only local context and forgets about global properties.

GloVe

- Brings back a more global approach to word embeddings.
- Word2Vec sees only local context and forgets about global properties.
- GloVe using a co-occurrence matrix to generate the word embeddings.

GloVe

- Brings back a more global approach to word embeddings.
- Word2Vec sees only local context and forgets about global properties.
- GloVe using a co-occurrence matrix to generate the word embeddings.
- Pretrained Glove models are available in gensim (downloader module).

Exercises

Exercise 1 - Single Words

1. Train a Word2Vec model on the pre-processed presidents' speeches.
   Use both CBoW and Skip-Gram!
2. Find the words most similar to, e.g. "state", "states", "president", or others.
3. Take a pre-trained gensim model and redo step 2.
4. Use the pre-trained model to check relationships:
   - King + Woman - Man = ?
   - France + Paris - London = ?

Phrases

How to create bigram in gensim? Use Phrases

```
from gensim.models import Phrases

bigram_generator = Phrases(tokenized_data, ....)
bigram_tokens = bigram_generator[tokenized_data]
```

What about trigrams? Well repeat the process but this time with bigram tokens.

```
trigram_generator = Phrases(bigram_tokens, ...)
trigram_tokens = trigram_generator(bigram_tokens)
```

Exercise 2 - BiGrams

1. Train a Bigram detector on the presidents' speeches. Hint: Use the Phrases class of gensim.
2. Train a Word2Vec model (both CBoW and Skip-Gram) on the bigram detector.
3. Find the words most similar to, e.g. "united_states", "american_people", or others.

Document Embeddings

Document Embedding

Document Embedding

- Word embeddings are ok, but what about documents.

Document Embedding

- Word embeddings are ok, but what about documents.
- A lot of tasks focus on documents and not only on words!

Document Embedding

- Word embeddings are ok, but what about documents.
- A lot of tasks focus on documents and not only on words!
- Examples: Sentiment Analysis, Sarcasm detection, Passage Retrieval, ad-hoc retrieval

Document Embedding

- Word embeddings are ok, but what about documents.
- A lot of tasks focus on documents and not only on words!
- Examples: Sentiment Analysis, Sarcasm detection, Passage Retrieval, ad-hoc retrieval
- How to get a document embedding?

Average Embedding

Average Embedding

- First idea: Take the average or sum of all words present in the document.

Average Embedding

- First idea: Take the average or sum of all words present in the document.
- Actually it is not a very bad idea, in practise it works!

Average Embedding

- First idea: Take the average or sum of all words present in the document.
- Actually it is not a very bad idea, in practise it works!
- Better idea: aggregate word embeddings with some weights (maybe tf-idf)

Doc2Vec

Doc2Vec

- Doc2Vec is similar to Word2Vec, but instead of Words it used the document (hence the name)!

Doc2Vec

- Doc2Vec is similar to Word2Vec, but instead of Words it used the document (hence the name)!
- It introduces paragraph id as a «token» in Word2Vec and tries to get an embedding of the document.

Doc2Vec

- Doc2Vec is similar to Word2Vec, but instead of Words it used the document (hence the name)!
- It introduces paragraph id as a «token» in Word2Vec and tries to get an embedding of the document.
- In general, it outperform the simple averaging or sum of Word2Vec Embeddings.

Exercises

Exercise 3

1. Compute document embeddings for the presidential speeches.
2. The document embedding must we the tf-idf weighed sum of the document terms.
3. Create a function that given the speech id computes the top-n (parameter) most similar documents.

Exercise 4

1. Create document embeddings for presidential speeches, this time using Doc2Vec
2. Create a function that given the speech id computes the top-n (parameter) most similar documents.
3. (Additional point) Create a function that computes the cosine similarity of a query (any query) and the document embeddings and return the top n.