Processament del Llenguatge Humà

Lab. 10: Word Embeddings

Gerard Escudero, Salvador Medina i Jordi Turmo

Grau en Intel·ligència Artificial

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Sumari

- Word Embeddings
  - Spacy
  - Word2Vec
  - Seq2Vec
  - FastText
- Exercici

Word Embeddings amb Gensim

Descarregar model pre-entrenats

Podeu descarregar models pre-entrenats de diferents llocs web

word2vec, fastText, ELMo, ...

http://vectors.nlpl.eu/repository/

Carregar un model amb Gensim

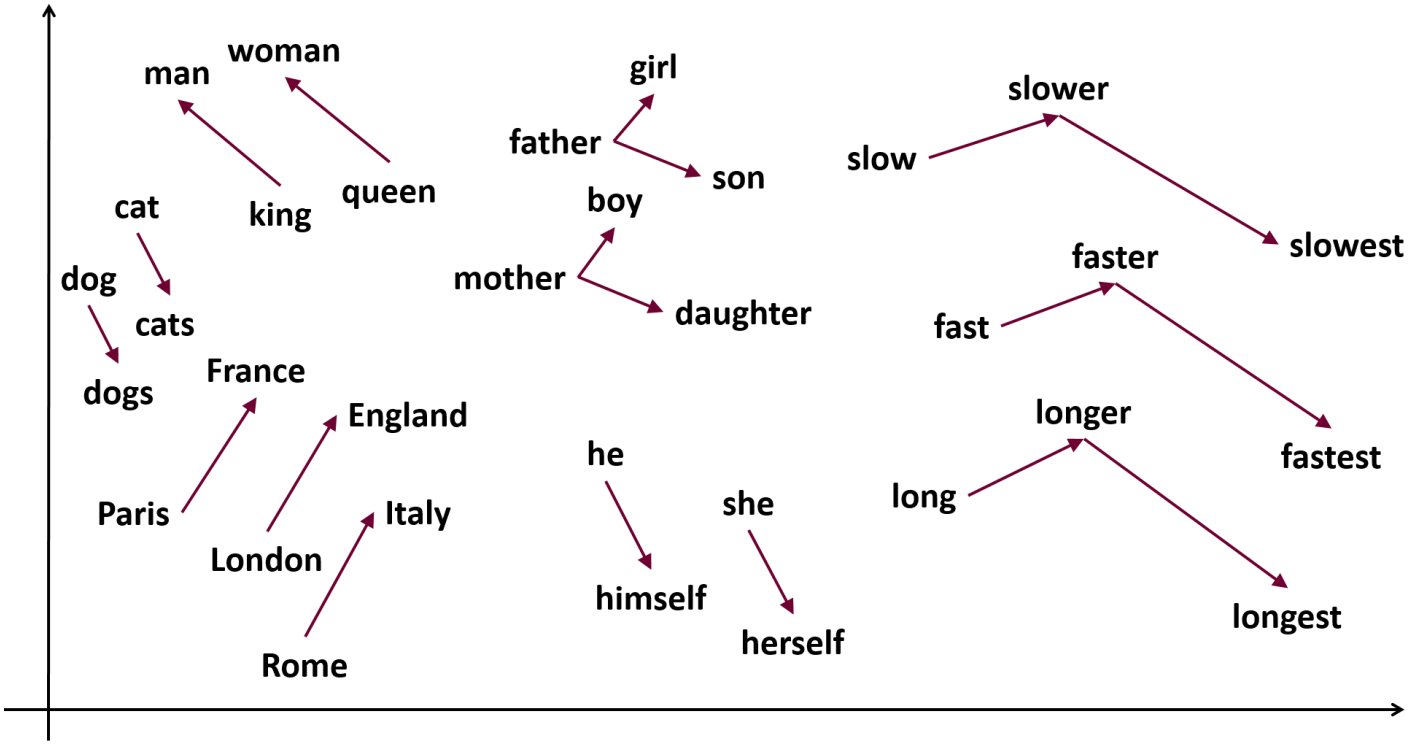
```
from gensim.models import KeyedVectors
# Word2vec permet dos formats: text i binari
kv = KeyedVectors.load_word2vec_format('model.bin', binary=True)
# Obtenir un word-vector
print(kv["paraula"]) # -> NDArray
```

Word Embeddings amb Gensim

Entrenar un model amb Gensim

```
from nltk.corpus import europarl
corpus = europarl_raw.spanish.words()
# Entrenar el model
from gensim.models import word2vec
model = word2vec.Word2Vec(corpus, vector_size=100, window=5, min_count=10, workers
# Obtenir un word-vector
print(model.wv["parlamento"]) # -> NDArray
# <!> Aquest dataset és massa petit, els embeddings generats no són de bona qualit
```

Analogies amb Gensim



Analogies amb Gensim (II)

Calcular paraules més similars

```
kv.most_similar("vector", topn=5)
# -> [('vectors', 0.8542011380195618), ('runge-lenz', 0.8305273652076721), ('pseud
```

Analogies

```
kv.most_similar(positive=["banc", "cadira"], negative=["diners"], topn=5)
# -> [('respatller', 0.6335902810096741), ('tamboret', 0.6063637137413025), ('bkf'
```

Altres

```
kv.doesnt_match(["cadira", "sofa", "gat", "butaca"])
# -> 'gat'
```

Avaluació amb Gensim

Descarregar datasets d'avaluació

https://github.com/vecto-ai/word-benchmarks https://github.com/RaRe-Technologies/gensim/tree/develop/gensim/test/test_data

Avaluar Analogies

```
from gensim.test.utils import datapath
analogies_result = kv.evaluate_word_analogies(datapath('questions-words.txt'))
print(analogies_result[0])
```

Avaluar Similitud

```
analogies_result = kv.evaluate_word_pairs(datapath('wordsim353.tsv'))
print(analogies_result) # -> (pearson, spearman, oov_ratio, )
```

Visualitzar Word Embeddings amb t-SNE

```
# "vocab" és una llista de paraules
X = model.wv[vocab]
# Entrenar el model de t-SNE
tsne = TSNE(n_components=2)
X_tsne = tsne.fit_transform(X)
# Crea un Dataframe
df = pd.DataFrame(X_tsne, index=vocab, columns=['x', 'y'])
# Imprimeix
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.scatter(df['x'], df['y'])
# Afegeix les etiquetes
for word, pos in df.iterrows():
    ax.annotate(word, pos)
```

FastText amb Gensim

També podeu carregar models de FastText amb Gensim

```
import gensim
# Soporta .bin (binari) i .vec (text pla)
model = gensim.models.fasttext.load_facebook_model('cc.en.300.bin.gz')
```

OOV amb FastText

```
'somethingweird' in kv.key_to_index
# -> False
oov_vector = kv['somethingweird']
# -> NDArray
```

N-gram Hashes

```
buckets = gensim.models.fasttext.ft_ngram_hashes('somethingweird', kv.min_n, kv.ma
# Podem obtenir el vector associat al bucket i=0
bucket_vector = kv.vectors_ngrams[buckets[0]]
# I obtenir la paraula més propera
closest = kv.similar_by_vector(bucket_vector)
# -> [('somel', 0.47820645570755005), ('somely', 0.4769449234008789), ('some.There
```

Gensim soporta Memory Maps (mmaps)

Un problema amb Word Embeddings és haver de carregar tots els arrays a memòria. Podeu utilitzar `mmaps` i fer servir el vostre disc en lloc de la RAM.

```
# Heu de guardar el model en un format compatible
model.save('model.bin')
# Llavors podeu carregar el model com a mmap
from gensim.models import FastText
model = FastText.load('model.bin', mmap='r')
```

Word Embeddings amb Spacy

Obtenir Word-Embeddings amb spaCy

```
import spacy
nlp = spacy.load("en_core_web_sm")
sentence = nlp("I sit on a bank.")
sentence[4].vector
# -> NDArray
```

Els vectors són contextuals

```
sentence2 = nlp("I borrow from a bank.")
sentence[4].vector == sentence2[4].vector
# -> False
```

Exercici:

Experimenta amb els Word Vectors

- Prova diferents models pre-entrenats.
- Defineix analogies i sinònims.
- Visualitza aquestes analogies i sinònims amb t-SNE.

Avaluació dels Word Embeddings alineats

- Utilitza Word Embeddings alineats per traduir una part del conjunt de proves d'analogies.
- Avalua el model de català amb aquest conjunt de proves.

Pràctica 4: Word-Embeddings

Donats els següents Datasets: Catalan General Crawling, Text Similarity i Text Classification

Enunciat:

- Entrenar models de Word2Vec (skip-gram) per a diferents mides de Datasets (e.g. 100MB, 500MB, 1GB, complet). Utilitzeu el corpus Catalan General Crawling (podeu afegir-ne d'altres).
  - Opcionalment, GloVe/FastText/CBOW.
- Entreneu un model de Similitud de Text Semàntic (Text Similarity):
  - Implementeu un model de regressió de similitud:
    - Baseline: 2 vectors concatenats + fully connected + sortida.
    - Podeu definir altres estructures.
  - Compareu els resultats amb diferents models d'incrustació de paraules (Word Embedding):
    1. One-Hot (limitar mida, normalitzar?)
    2. Models de Word2Vec/GloVe pre-entrenats:
       - Word2Vec + Mean
       - Word2Vec + Mean ponderada (e.g. TF-IDF)
    3. spaCy ca_core_news_md
    4. RoBERTa spaCy/ca_core_news_trf / roberta-base-ca-v2
       - CLS (Amb spaCy, `doc._.trf_data.tensors[-1]`)
       - Mean (Amb spaCy, mitjana de `doc._.trf_data.tensors[-1]`)
    5. RoBERTa fine-tuned roberta-base-ca-v2-cased-sts

Pràctica 4: Word-Embeddings

- Entreneu el mateix model amb embeddings entrenables inicialitzats amb:
  - Random Embeddings (uniforme)
  - Word2Vec
- Analitzeu els resultats.

Opcional:

- Entreneu un model de classificació amb el conjunt de dades Text Classification

Pràctica 4: Model Baseline

Exemple del model baseline. Podeu implementar-ho amb PyTorch o qualsevol altre framework.

```
import tensorflow as tf
def build_and_compile_model(hidden_size: int = 64) -> tf.keras.Model:
  model = tf.keras.Sequential([
      tf.keras.layers.Concatenate(axis=-1, ),
      tf.keras.layers.Dense(hidden_size, activation='relu'),
      tf.keras.layers.Dense(1)
  ])

  model.compile(loss='mean_absolute_error',
                optimizer=tf.keras.optimizers.Adam(0.001))
  return model

m = build_and_compile_model()
# E.g.
import numpy as np
y = m((np.ones((1, 100)), np.ones((1,100)), ), )
```

Pràctica 4: Embedding Baseline

Amb Word Embeddings entrenables. Passarem la llista d'índexs.

```
import tensorflow as tf
def build_and_compile_model(
        input_length: int = 10, hidden_size: int = 64, dictionary_size: int = 1000
) -> tf.keras.Model:
    input_1, input_2 = tf.keras.Input((input_length, ), dtype=tf.int32, ), tf.kera
    # Define Layers
    embedding = tf.keras.layers.Embedding(
        dictionary_size, embedding_size, input_length=input_length, mask_zero=True
    pooling = tf.keras.layers.GlobalAveragePooling1D()
    concatenate = tf.keras.layers.Concatenate(axis=-1, )
    hidden = tf.keras.layers.Dense(hidden_size, activation='relu')
    output = tf.keras.layers.Dense(1)
    # Pass through the layers
    _input_mask_1, _input_mask_2 = tf.not_equal(input_1, 0), tf.not_equal(input_2,
    _embedded_1, _embedded_2 = embedding(input_1, ), embedding(input_2, )
    _pooled_1, _pooled_2 = pooling(_embedded_1, mask=_input_mask_1), pooling(_embe
    _concatenated = concatenate((_pooled_1, _pooled_2, ))
    _hidden_output = hidden(_concatenated)
    _output = output(_hidden_output)
    # Define the model
    model = tf.keras.Model(inputs=(input_1, input_2, ), outputs=_output, )
    model.compile(loss='mean_absolute_error',
                  optimizer=tf.keras.optimizers.Adam(0.001))
    return model
```