Daniela Jiménez Téllez - A01654798

Word Embeddings Pre-entrenados

Función most_similar_words

```python
import numpy as np

def most_similar_words(word, vectors, index_to_key, key_to_index, topn=10):

    # retrieve word_id corresponding to given word
    word_id = key_to_index.get(word)

    # retrieve embedding for given word
    word_embedding = vectors[word_id]

    # calculate similarities to all words in our vocabulary
    similarities = np.dot(vectors, word_embedding)

    # get word_ids in ascending order with respect to similarity score
    word_ids = np.argsort(similarities)

    # reverse word_ids to have most similar words at the top
    word_ids = word_ids[::-1]

    # get boolean array with element corresponding to word_id set to false
    mask = np.ones(len(word_ids), dtype=bool)
    mask[word_id] = False

    # obtain new array of indices that doesn't contain word_id
    word_ids = word_ids[mask]

    # get topn word_ids
    top_ids = word_ids[:topn]

    # retrieve topn words with their corresponding similarity score
    top_words = [(index_to_key[idx], similarities[idx]) for idx in top_ids]

    # return results
    return top_words
```

Resultados

```
('cacti', 0.66345644),
('saguaro', 0.6195855),
('pear', 0.5233487),
('cactuses', 0.5178282),
('prickly', 0.51563185),
('mesquite', 0.4844855),
('opuntia', 0.45400846),
('shrubs', 0.4536207),
('peyote', 0.4534496)]
```

Función analogy

```python
from numpy.linalg import norm

def analogy(positive, negative, vectors, index_to_key, key_to_index, topn=10):

    # find ids for positive and negative words
    pos_ids = [key_to_index[word] for word in positive]
    neg_ids = [key_to_index[word] for word in negative]
    given_word_ids = pos_ids + neg_ids

    # get embeddings for positive and negative words
    pos_emb = np.sum(vectors[pos_ids], axis=0)
    neg_emb = np.sum(vectors[neg_ids], axis=0)

    # get embedding for analogy
    emb = pos_emb - neg_emb

    # normalize embedding
    emb = emb / norm(emb)

    # calculate similarities to all words in out vocabulary
    similarities = np.dot(vectors, emb)

    # get word_ids in ascending order with respect to similarity score
    ids_ascending = np.argsort(similarities)

    # reverse word_ids
    ids_descending = ids_ascending[::-1]

    # get boolean array with element corresponding to any of given_word_ids set to false
    given_words_mask = np.ones(len(ids_descending), dtype=bool)
    given_words_mask[given_word_ids] = False

    # obtain new array of indices that doesn't contain any of the given_word_ids
    ids_descending = ids_descending[given_words_mask]

    # get topn word_ids
    top_ids = ids_descending[:topn]

    # retrieve topn words with their corresponding similarity score
    top_words = [(index_to_key[idx], similarities[idx]) for idx in top_ids]

    # return results
    return top_words
```

Resultados

```
('queen', 0.67132765),
('princess', 0.5432625),
('throne', 0.53861046),
('monarch', 0.53475744),
('daughter', 0.49802512),
('mother', 0.49564427),
('elizabeth', 0.48326525),
('kingdom', 0.47747087),
('prince', 0.46682402)]
```