Text Sequence

DL-U08

Outline

- Recap: N-Grams and Bag of words
- One-hot encoding of words and characters
- Word Embedding

Word n-grams

- Word n-grams are groups of N (or fewer) consecutive words that you can extract from a sentence.
- The sentence “The cat sat on the mat.”
- It may be decomposed into set of
  - bag-of-2-grams: {"The", "The cat", "cat", "cat sat", "sat", "sat on", "on", "on the", "the", "the mat", "mat"}
  - bag-of-3-grams: {"The", "The cat", "cat", "cat sat", "The cat sat", "sat", "sat on", "on", "cat sat on", "on the", "the", "sat on the", "the mat", "mat", "on the mat “ }

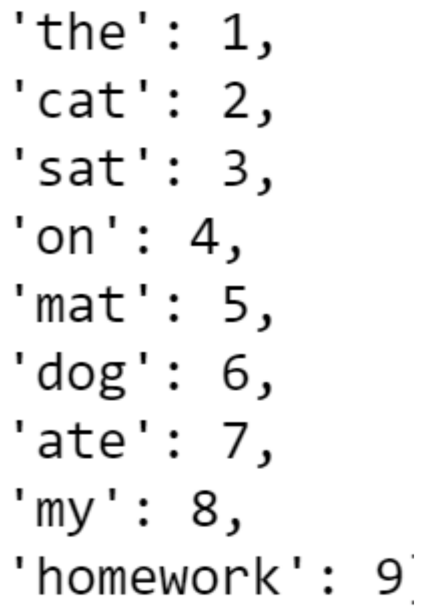
Lost order

- Bag of words or Bag of grams
  - Set → lost order → lost general structure of the sentences
  - To be used in **shallow language-processing models** rather than in deep-learning models

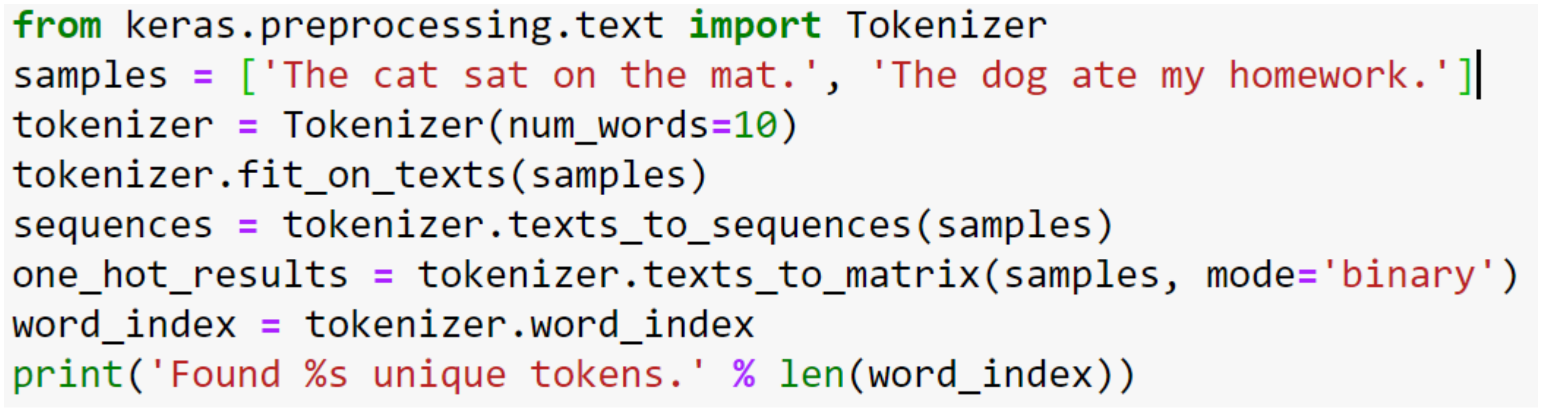
One hot encoding

Note that index 0 is reserved

- 'The cat sat on the mat.' → [0, 1, 1, 1, 1, 1, 0, 0, 0, 0]
- 'The dog ate my homework.' → [0, 1, 1, 1, 1, 1, 0, 0, 0, 0]



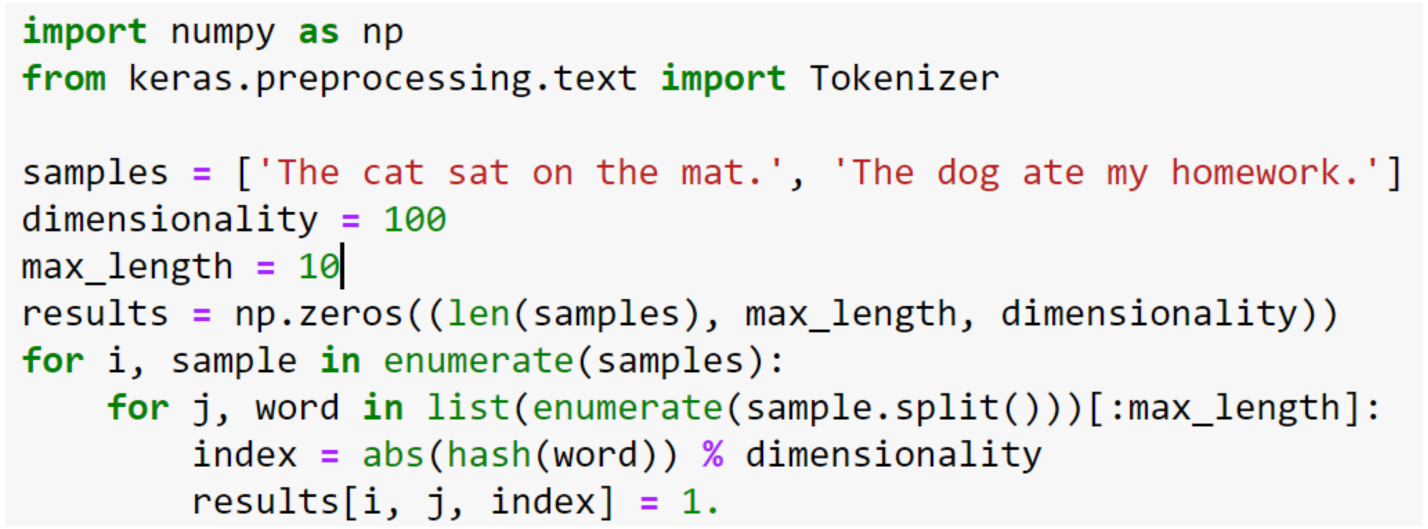
```
'the': 1,
'cat': 2,
'sat': 3,
'on': 4,
'mat': 5,
'dog': 6,
'ate': 7,
'my': 8,
'homework': 9
```



```python
from keras.preprocessing.text import Tokenizer
samples = ['The cat sat on the mat.', 'The dog ate my homework.']
tokenizer = Tokenizer(num_words=10)
tokenizer.fit_on_texts(samples)
sequences = tokenizer.texts_to_sequences(samples)
one_hot_results = tokenizer.texts_to_matrix(samples, mode='binary')
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

One-hot hashing

- When the number of unique tokens in vocabulary is too large to handle explicitly
  - Pros: save memory
  - Cons: hash collisions



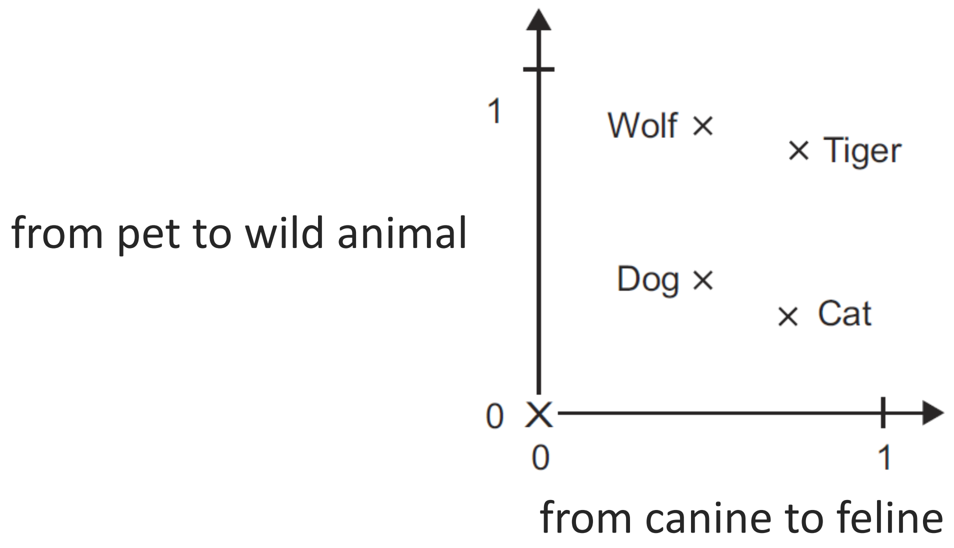
```
import numpy as np
from keras.preprocessing.text import Tokenizer

samples = ['The cat sat on the mat.', 'The dog ate my homework.']
dimensionality = 100
max_length = 10
results = np.zeros((len(samples), max_length, dimensionality))
for i, sample in enumerate(samples):
    for j, word in list(enumerate(sample.split()))[:max_length]:
        index = abs(hash(word)) % dimensionality
        results[i, j, index] = 1.
```

Word embedding

- One-hot vector is sparse - 20,000 tokens, hard-coded
- Word embedding is dense
  - Low dimensional floating-point vectors – 256, 512, 1,024-dimensional
  - Learn from data
- Two ways to obtain word embeddings
  - Learn from scratch, jointly with your own task
  - Use pretrained word embeddings

Geometric of word embedding

- Word embeddings are meant to map human language into a geometric space.
  - geometric distance between any two word vectors reflects the semantic distance between the two words
  - words meaning different things are embedded at points far away from each other, whereas related words are closer



Word-embedding space

- It’ s hard to find a word-embedding space used for any natural-language-processing task
- What makes a good word-embedding space depends heavily on your task
  - the perfect word-embedding space for an English-language movie-review sentiment analysis model may look different from the perfect embedding space for an English language legal-document-classification model, because the importance of certain semantic relationships varies from task to task.

Keras – embedding layer

```
from keras.layers import Embedding

embedding_layer = Embedding(1000, 64)
```

1000 – the number of tokens
64 - dimensional of the embedding

Each word is represented by a 64-d real-valued vector

Train embedding layer

- Input: each word is one-hot encoded.
- The target vectors are initialized with small random numbers.
- The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Back-propagation algorithm.

Pre-trained Embedding

- Instead of learning word embeddings jointly with the problem you want to solve,
- you can load embedding vectors from a precomputed embedding space.
  - Word2vec (https://code.google.com/archive/p/word2vec)
  - Global Vectors for Word Representation (GloVe, https://nlp.stanford.edu/projects/glove)
  - using word-occurrence statistics + unsupervised learning