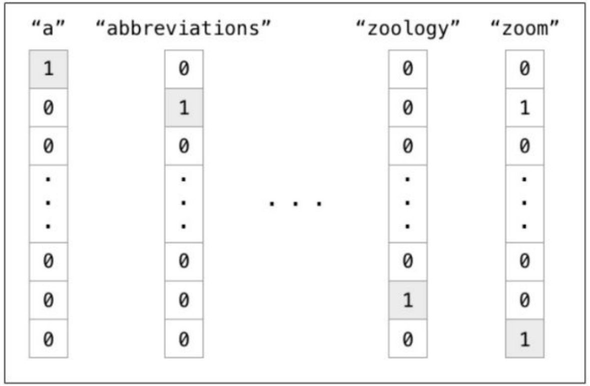
CAS ML
Natural Language Processing

Dr. Mark Rowan – Rowan Cognitive Data Science Solutions

Revisiting Word Embeddings

- Representations of words as numerical vectors in a high-dimensional space.
- Capture semantic and syntactic relationships: Word embeddings encode meaning and context, allowing for similarities and differences between words to be captured.
- Wide range of applications:
  - text classification, sentiment analysis, machine translation, information retrieval…
- Pre-trained word embedding models enable easy integration into various NLP tasks:
  - Word2Vec, GloVe, FastText

Classical representation: one-hot encoding



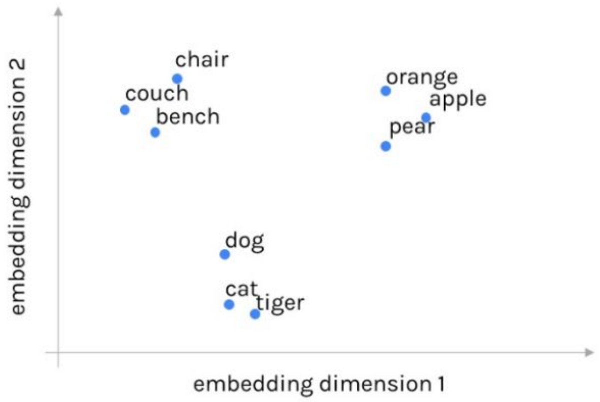
- Assign every word to a position identifier (ID)
- Perform one-hot encoding of these words based on IDs
- Typically used for categorical data
- Poor representation of words - no context or semantics
- Unable to scale even with medium sized vocabulary

Why word embeddings?

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

- Use unsupervised or semi-supervised learning to automatically learn numeric representations
- Represent each word with a fixed sized vector of numbers - this is known as a word embedding
- A word is known by the company it keeps!
- Various concepts or dimensions of a word are represented by this fixed-width dense vector

Word embedding representations



- Map linguistic units into a continuous vector space with a lower dimension (dense representation)

  Linguistic unit can be - word, characters, n-grams, sentences, paragraphs, documents

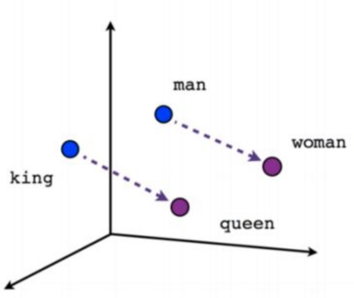
- The meaning of the linguistic unit is represented by the multi-dimensional numeric vector (embedding)
- Based on the distributional hypothesis - linguistic units with similar distributions have similar meanings
- The distributional property is usually induced from document or context or textual vicinity (like a sliding window)

Word embedding properties 1

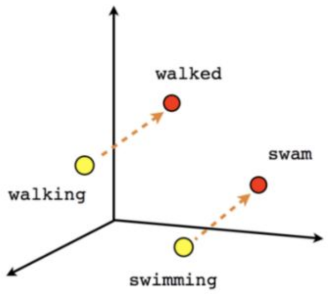
$$\text{good} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

- Embeddings are compact, dense and low dimensional representations
- Build efficient dense representations using co-occurences and context
- Each single component of the embedding vector representation does not have any meaning of its own
- Meaning of a word is smeared across all dimensions of its embedding
- The interpretable features (word contexts) are hidden and distributed among uninterpretable embedding components

Word embedding properties 2



Male-Female



Verb tense

Read more:
https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c

Gensim for word embeddings

- Word2Vec - https://radimrehurek.com/gensim/models/word2vec.html
- Doc2Vec - https://radimrehurek.com/gensim/models/doc2vec.html
- fastText - https://fasttext.cc/, gensim: models.fasttext – FastText model
- GloVe (Global Vectors for word representations) - https://nlp.stanford.edu/projects/glove/
- Load pre-trained models - http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/

```
>>> from gensim.test.utils import common_texts, get_tmpfile
>>> from gensim.models import Word2Vec
>>>
>>> path = get_tmpfile("word2vec.model")
>>>
>>> model = Word2Vec(common_texts, size=100, window=5, min_count=1, workers=4)
>>> model.save("word2vec.model")
```

Hands-on

Hands-on: Text Mining 101

Objective: Explore a large text dataset using basic NLP techniques, and perform sentiment analysis

Data: https://ai.stanford.edu/~amaas/data/sentiment/
IMDB Movie Reviews dataset containing 25k movie reviews with sentiment labels

Libraries:

- TextBlob: for text preprocessing and sentiment analysis
- NLTK: for stopwords removal
- Gensim: for word vectorization

Objectives

1. Text Preprocessing:
    a. Tokenization: Split the text into individual tokens.
    b. Lowercasing: Convert all text to lowercase for consistency.
    c. Stopwords Removal: Remove common words (e.g., "the," "is") that do not carry significant meaning.

2. Sentiment Analysis:
    a. Utilize TextBlob to analyze the sentiment polarity (positive or negative) of each movie review.
    b. Calculate the overall sentiment distribution of the dataset.

3. Analysis and Insights:
    a. Examine statistical features such as word count, sentence count, and average word length.
    b. Perform exploratory analysis: what are the most common words in positive and negative reviews? What patterns or trends can you find?

3. Word Vectorization with Word2Vec:
    a. Train a Word2Vec model using Gensim to create word embeddings.
    b. Can you find the most similar words to "movie", "good", or "action"?
    c. Can you find word analogies using vector arithmetic? For example, what is to "hero" as "actor" is to "actress"?
    d. Can you identify clusters of related words in the vector space? Are there groups of words that have similar vector representations?

4. Optional: Text Classification (Bonus Challenge):
    a. Build a basic text classification model using word vectors as input, and sentiment labels as output.
    b. Build your own model to predict sentiment labels given text.
    c. Split the dataset into training and testing sets and evaluate the model's performance.

→ Findings to be presented tomorrow morning!
Each group gets 5-10 minutes.

Useful code snippets

- Tokenization using TextBlob

```
from textblob import TextBlob

text = "This is a sample sentence for
tokenization."
blob = TextBlob(text)
tokens = blob.words
print(tokens)
```

- Stemming using NLTK

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
word = "running"
stemmed_word = stemmer.stem(word)
print(stemmed_word)
```

- Lemmatization using TextBlob

```
from textblob import Word

word = "running"
lemmatized_word = Word(word).lemmatize()
print(lemmatized_word)
```

- Stopword removal using NLTK

```
from nltk.corpus import stopwords

stopwords = set(stopwords.words('english'))
text = "This is a sample sentence for stopwords removal."
tokens = text.split()
filtered_tokens = [token for token in tokens if token.lower() not in stopwords]
filtered_text = ' '.join(filtered_tokens)
print(filtered_text)
```

- Word vectorization using Gensim

```
from gensim.models import Word2Vec

sentences = [["I", "like", "apples"], ["I", "love", "bananas"], ["Fruits", "are",
"delicious"]]
model = Word2Vec(sentences, min_count=1)
word_vector = model.wv['apples']
print(word_vector)
print(model.wv.most_similar('apples'))
```

Errors? Try `pip install numpy==1.19 gensim==4.0`

Reading the data

Get the data into colab:
!wget <URL>
!tar xf <file>

```
import os
import pandas as pd

texts = []
labels = []

data_path = '/content/acmlImdb/train/'
neg_directory = os.path.join(data_path,'neg')
for filename in os.listdir(neg_directory):
    file_path = os.path.join(neg_directory, filename)
    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read()
        texts.append(text)
        labels.append('neg')

df = pd.DataFrame({'text': texts, 'label': labels})
```