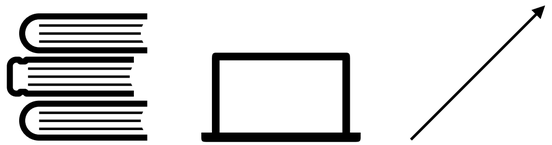
Word2Vec



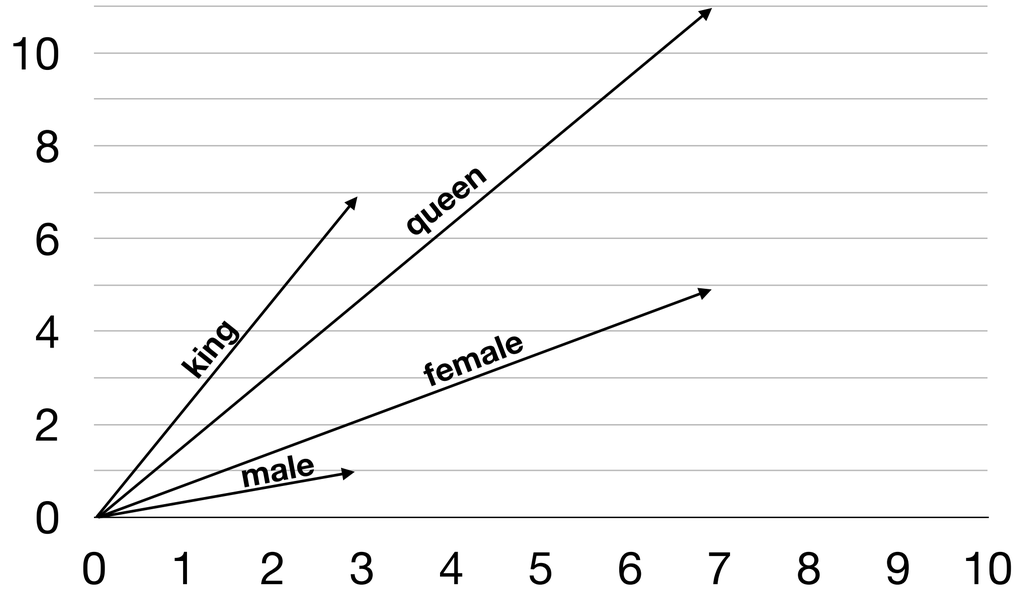
Yet Another Natural Language Processing Technique

Ian Delabie

A Basic Introduction to Word2Vec

- algorithm (neural network model) that transforms a given word into a numerical vector using its context within a training corpus or a training corpora

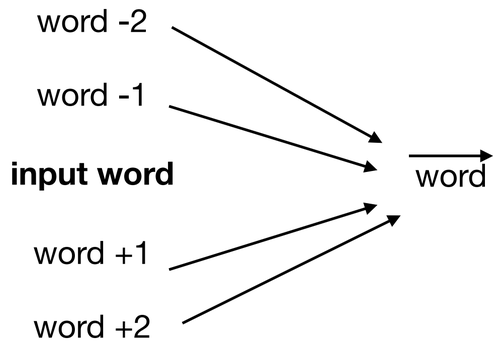
King is the title given to a **male** monarch in a variety of contexts. The **female** equivalent is **queen**.



Two Ways to Vectorize Words

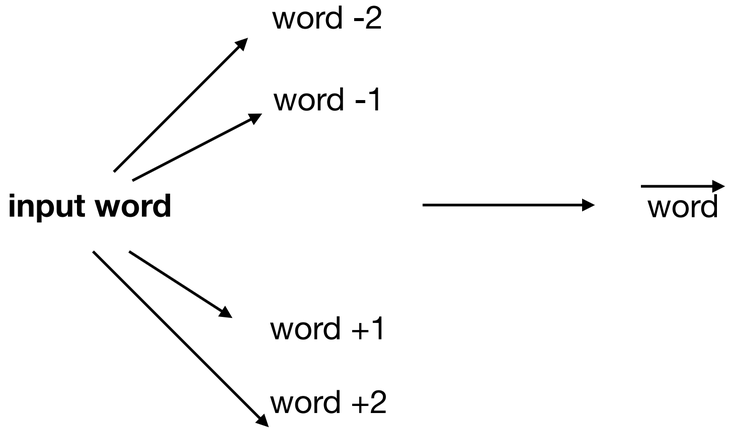
Continuous Bag of Words (CBOW)

context of input word generates vector



Skip-gram

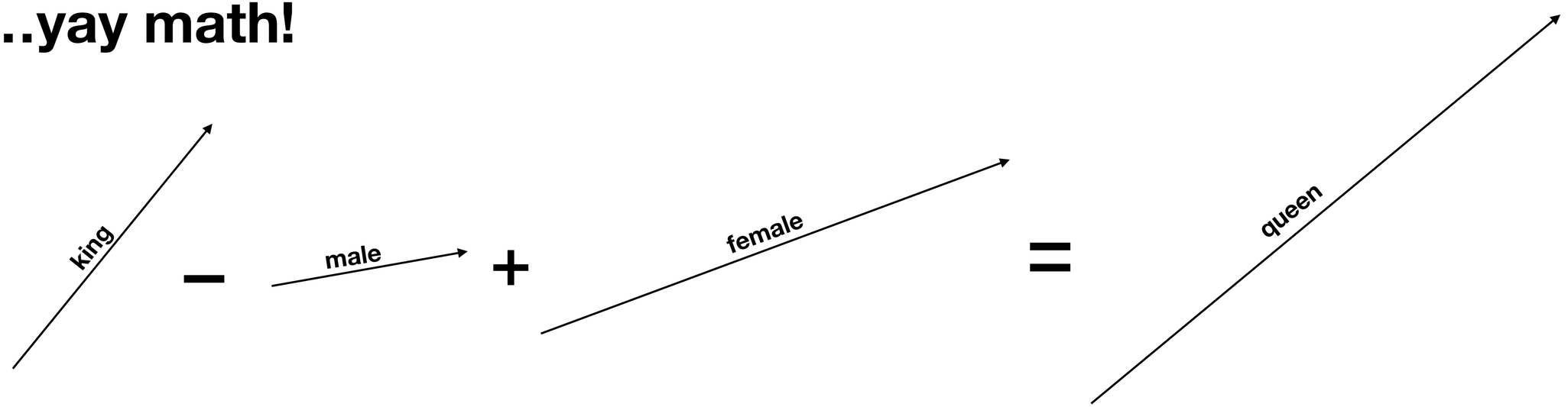
input word generates “vectorized context”



But, why?

Because…yay math!

Vector Algebra



Cosine Similarity

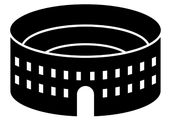
$$\frac{\overrightarrow{king} \cdot \overrightarrow{queen}}{||\overrightarrow{king}|| \times ||\overrightarrow{queen}||} = 0.986933$$

$$\frac{\overrightarrow{king} \cdot \overrightarrow{female}}{||\overrightarrow{king}|| \times ||\overrightarrow{female}||} = 0.854788$$

Ok, but so what for the social sciences?

Manipulation and comparison of word embeddings can be useful…

- the study of **language** and **semantics** provide an essential insight into **human behavior** and **societies**



  - words can have a **multitude of meanings** depending on the cultural, political and societal context

- and these meanings can **shift** over time reflecting **cultural/political/societal transformations**

  - Wouldn't it be nice to able to **use computational methods to study semantics** across and within cultural/political/societal contexts?

Word2Vec

The Beauty of Word2Vec

- encoding **semantic information** into a vector format has attractive applications across the social sciences
    - cosine similarity between vectors equates to semantic similarity

        - **tracking semantic evolutions**

            Evolution of the meaning of “equality” in the U.S. using historical newspaper articles (Rodman 2020)

            Evolution of social class identifiers across economic transformation of the 20th century (Kozlowski, Teddy and Evans 2019)

        - **semantic classification**

            Predicting political ideology of legislators from parliamentary debates (Rheault and Cochrane 2020)

            Predicting student learning from post-class student comments (Luo, Sorer, Good and Mine 2015)

Word2Vec in the Computational Ecosystem

- available across multiple open-source natural language processing and unsupervised topic modeling libraries (most have pre-processed corpora to train model)

spaCy AllenNLP GENSIM topic modelling for humans

- GenSim offers the lowest (arguably) entry barrier to Word2Vec and seems to have the most flexibility
    - Many option to customize to Word2Vec models (vector dimension, computational efficiency, choice of CBOW or Skip-gram)
    - Easier to train your own model + access to a variety of pre-trained corpora
- Many Word2Vec extensions
    - Doc2Vec- vector representation of documents
    - BioVectors- vector representation of biological sequences
    - Top2Vec- vector representation of topics

Basic Worked Example (Part 1)

Introducing the royal family of vectors

```
#Importing Word2Vec from GenSim
from gensim.models.word2vec import Word2Vec
import gensim.downloader as api
```

➜ Word2Vec algorithm using GenSim (open source topic modeling library)

```
#Downloading existing corpora from the Gensim database - Text from Wikipedia
dataset = api.load("text8")
```

➜ Downloading pre-processed training corpus- Wikipedia articles

```
#Tokenizing
data = [d for d in dataset]
data_part1 = data[:1000]
```

➜ Subsetting corpus to speed up computation

```
#Training the Word2Vec model with text from Wikipedia (in the gensim database)
word2vec = Word2Vec(data_part1, min_count=2)
```

➜ Vectorizing words using CBOW model (default) in Word2Vec

```
#Displaying the vector for the word "King"
word2vec.wv['king']
```

```
array([-0.6247059 ,  2.734628  ,  0.07881403,  0.8415265 ,  1.0919182 ,
       -1.1927114 , -0.15348631,  0.31121325,  1.2566534 , -0.67269063,
        0.5604339 ,  2.6822858 ,  2.854661  ,  1.7861245 ,  3.8821514 ,
       -1.1519864 ,  1.8963438 , -0.42539087,  2.2171638 , -0.6451796 ,
        1.3156921 , -0.77337134,  2.1716301 ,  3.0620973 ,  2.2385871 ,
        1.9413279 ,  0.76725775, -1.1715403 , -1.7636768 ,  0.4159524 ,
       -0.37398598, -2.7588546 ,  0.58317375, -1.5069119 ,  1.7781116 ,
        1.7118299 , -0.42223325, -1.4407419 ,  1.7727789 ,  1.5959058 ,
       -0.94584805, -0.01704919, -0.263002  ,  2.6214325 ,  0.9325347 ,
       -1.944263  , -3.248921  , -1.9091796 ,  1.0752041 ,  0.41822544,
        3.2948973 ,  0.09252111, -3.8474638 , -2.503489  , -0.3280734 ,
        0.8311117 ,  3.3224537 ,  0.52778506,  0.30486453,  1.3455604 ,
        0.24585572,  0.9263525 ,  1.793085  , -1.0099862 ,  0.3908879 ,
       -1.2896887 , -1.3128164 , -0.11316916, -0.41064888, -1.254687  ,
        2.5540981 , -2.1212764 , -1.9593897 ,  1.3195864 ,  0.33235133,
        1.4594887 ,  2.838726  , -0.72939336, -0.65724856,  0.7372532 ,
        0.6839381 ,  1.0291715 ,  0.5574616 ,  1.6526141 ,  0.13806847,
       -0.33238834, -0.01475193,  1.5813116 , -0.07260821,  1.6930803 ,
        0.7732846 ,  0.8207004 , -2.3087122 , -1.3797399 , -1.9110872 ,
        0.5951446 ,  1.8832892 ,  0.76675755, -2.1122017 , -2.0272765 ],
      dtype=float32)
```

➜ The word “king” as a 100-dimensional vector

Basic Worked Example (Part 2)

```
#Cosine Similarity between King and Queen
word2vec.wv.similarity('king', 'queen')
```

```
0.71888465
```

```
#Cosine Similarity between King and Female
word2vec.wv.similarity('king', 'female')
```

```
0.04135818
```

```
#Cosine Similarlity between King and Male
word2vec.wv.similarity('king', 'male')
```

```
0.070132636
```

```
#Cosine Similarity between Male and Female
word2vec.wv.similarity('male', 'female')
```

```
0.92445123
```

➜ Semantic similarity -“king”, “queen”, “male”, “female”

Slightly Less Basic Worked Example (Part 1)

Trump January 6th Speech

```
#Importing necessary modules and tokenizing original text
from gensim.models.word2vec import Word2Vec
from smart_open import smart_open
from gensim.utils import simple_preprocess
from gensim import corpora
from collections import defaultdict

tokens = [simple_preprocess(sentence, deacc=True) for sentence in open(r'Trump_JAN6.txt', encoding='utf-8')]
```

➜ Modules: GenSim (Word2Vec), smart_open, collections

```
#Creating a list of stop words and cleaning corpus of these stop words
stopwords = set('for a of the and to in who it'.split(' '))

cleaned_corpus = [[word.lower().strip() for word in sent if word not in stopwords] for sent in tokens]
```

➜ Create list of stop words to purge from speech (i.e., corpus)

```
# Creating a dictionary to rid the corpus of unique words
frequency = defaultdict(int)
for text in cleaned_corpus:
    for token in text:
        frequency[token] += 1

fully_cleaned_corpus = [[token for token in text if frequency[token] > 1] for text in cleaned_corpus]
```

➜ Use pythonic dictionaries to rid speech of unique words

```
model_jan6=gensim.models.Word2Vec(fully_cleaned_corpus,min_count=2)

len(model_jan6.wv)
```

```
819
```

➜ Train Word2Vec model

Slightly Less Basic Worked Example (Part 2)

```
model_jan6.wv.most_similar(positive="fraud")
```

```
[('that', 0.9955363869667053),
 ('are', 0.9954153895378113),
 ('you', 0.995338499546051),
 ('they', 0.9952858686447144),
 ('is', 0.995254397392273),
 ('he', 0.9952491521835327),
 ('people', 0.9951519966125488),
 ('we', 0.9951363205909729),
 ('was', 0.9950646162033081),
 ('election', 0.9950599670410156)]
```

➜ “Fraud” Semantic Neighbors

```
model_jan6.wv.similarity('fraud', 'election')
```

```
0.9950598
```

```
model_jan6.wv.similarity('fair', 'election')
```

```
0.94566756
```

➜ Semantic Similarity- Fraud, Election, Ballots, Fair

```
model_jan6.wv.similarity('fraud', 'ballots')
```

```
0.9948075
```

```
model_jan6.wv.similarity('fair', 'ballots')
```

```
0.9450196
```

Pittfals 

- No vector representation of words outside the vocabulary (i.e. outside the training corpora)
- Semantic similarity at sub-word level undetected (ignores prefix & suffix similarities)
- No distinction of words with multiple meanings (one word, one vector representation)

Alternatives 

- GloVe - counts the frequency of words in context (word to word co-occurrence matrix)

  + captures word embedding in global context

- FastText - breaks words into subwords to construct word embeddings

  + vector representation for words outside vocabulary, prefix & suffix similarities

- BERT - compares any given word to the other words in the sentence (masked language modeling)

  + can account for multiple meanings

Additional Resources

If you really must…

Basic Conceptual Introduction

https://jalammar.github.io/illustrated-word2vec/

https://medium.com/@kashyapkathrani/all-about-embeddings-829c8ff0bf5b

https://code.google.com/archive/p/word2vec/

Presentation and Worked Examples (Jupyter Notebooks)

https://github.com/idelabie/Word2Vec

Advanced Conceptual Overview

https://stackabuse.com/implementing-word2vec-with-gensim-library-in-python/

Text-as-Data (PS2751)

https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html

https://radimrehurek.com/gensim/models/word2vec.html#gensim.models.word2vec.Word2Vec

https://research.google/pubs/pub41224/

https://cs224d.stanford.edu/lecture_notes/notes1.pdf

Applications in the social sciences

https://tinyurl.com/RodriguezSpring

https://tinyurl.com/KozlowskiTaddyEvans

https://tinyurl.com/RheaultCochrane

https://tinyurl.com/EmmaRodman

https://tinyurl.com/LuoShaymaaGodaMine