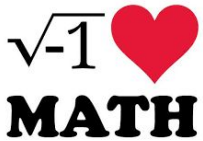
An Embedding is Worth 1000 Words

Jaya Zenchenko
March 29, 2018
Austin Big Data AI Meetup

About Me

- BA Math
- MS Math Education
- MS Applied Math
- Research Scientist
  - DoD: Images, Video, Graphs
- Data Scientist
  - NLP, Ranking, Unsupervised Learning
- Data Science Manager
  - Images, Text
- Founder of “Women in Data Science - Austin” Meetup



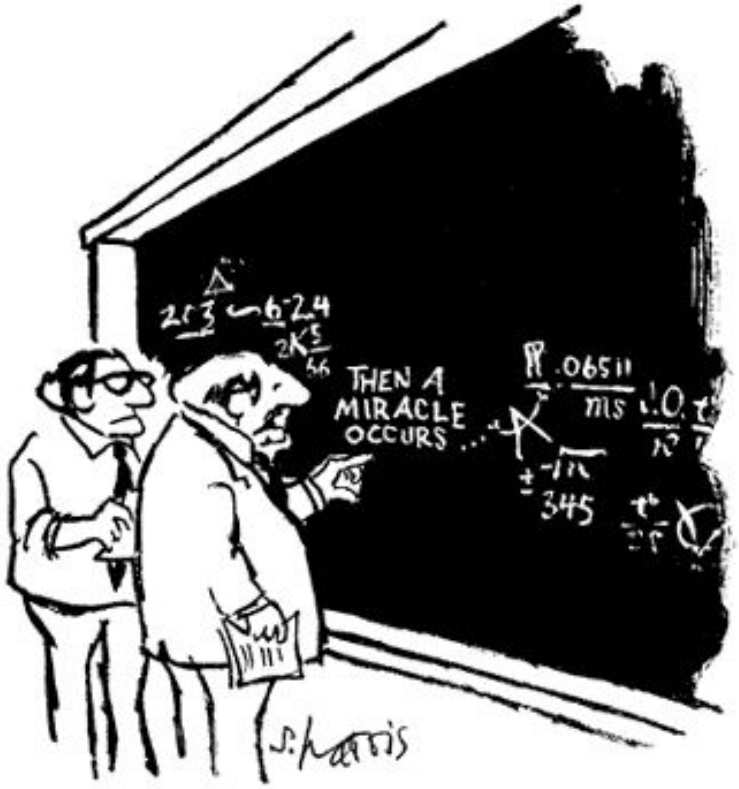
@datanerd_jaya

bijaya.zenchenko@gmail.com

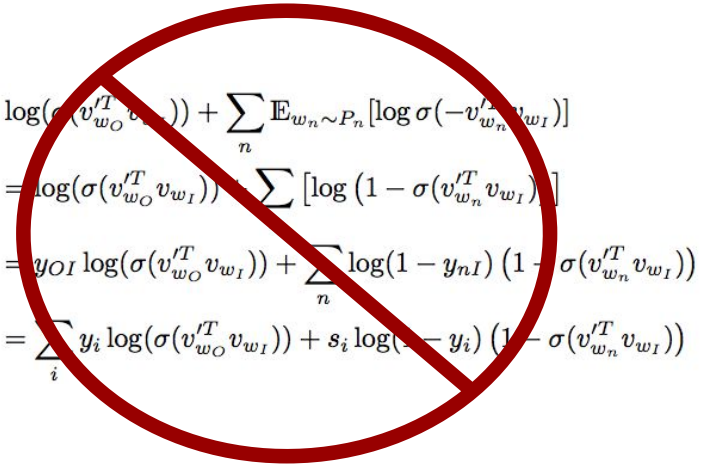
Credit: http://briasimpson.com/wp-content/uploads/2013/04/Swamp-Overwhelm.jpg

Pop Quiz!

- What is an embedding and why do I care?
- Is an embedding really worth 1000 words?
- What does it mean for words to be related?
- What are some gotchas in dealing with text data?
- What approaches can I use to get insights from my text data?



"I think you should be more explicit here in step two."


$\log(\sigma(v'^T_{w_O} v_{w_I})) + \sum_n \mathbb{E}_{w_n \sim P_n}[\log \sigma(-v'^T_{w_n} v_{w_I})]$
$= \log(\sigma(v'^T_{w_O} v_{w_I})) + \sum_n \left[\log\left(1 - \sigma(v'^T_{w_n} v_{w_I})\right)\right]$
$= y_{OI} \log(\sigma(v'^T_{w_O} v_{w_I})) + \sum_n \log(1 - y_{nI}) \left(1 - \sigma(v'^T_{w_n} v_{w_I})\right)$
$= \sum_i y_i \log(\sigma(v'^T_{w_O} v_{w_I})) + s_i \log(\ - y_i) \left(1 - \sigma(v'^T_{w_n} v_{w_I})\right)$

Word Embeddings

- Coined in 2003
- Also called “Word Vectors”
- Text being converted into numbers
- Not just for words! Sentences, documents, etc.
- Not magic!

Why am I here?

- Word Embeddings are cool!
- No magic
- Highlight built in functionality of **“gensim”** and **“scikit-learn”** python package to quickly tackle your next text based project

What Can We Do?

- Get insights
- Search/Retrieval
- Clustering (grouping)
- Identify Topics (i.e. themes)

- Apply techniques to other data sets (images, click through, etc)!!

Definitions

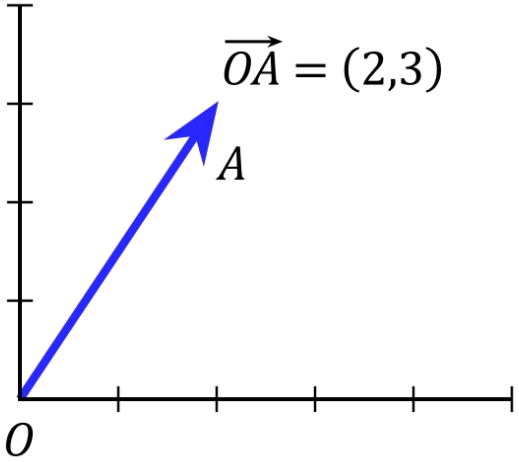
Vector/Embedding

Vector or Embedding

2-dimensional vector = [2, 3]

3-dimensional vector = [2, 3, 1]

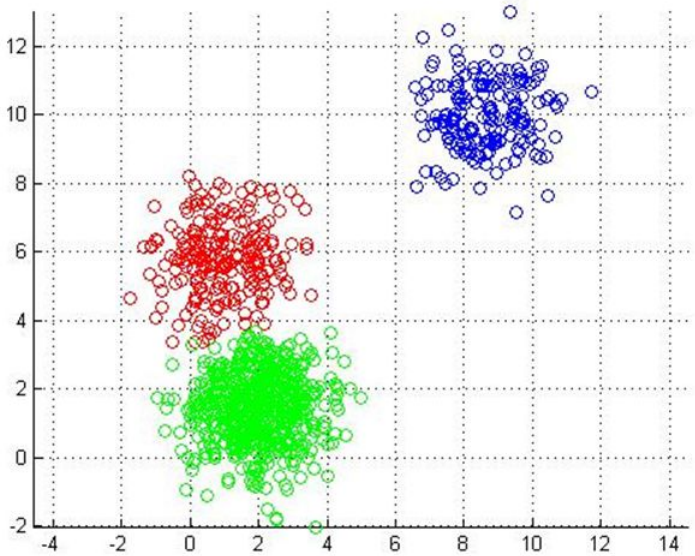
N-dimensional vector = [2, 3, 1, 5, ..., nth value]



Credit: https://maths-wiki.wikispaces.com/Vectors

Clustering

Grouping points that are close - what is “close”?



Credit: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Text Based Similarity

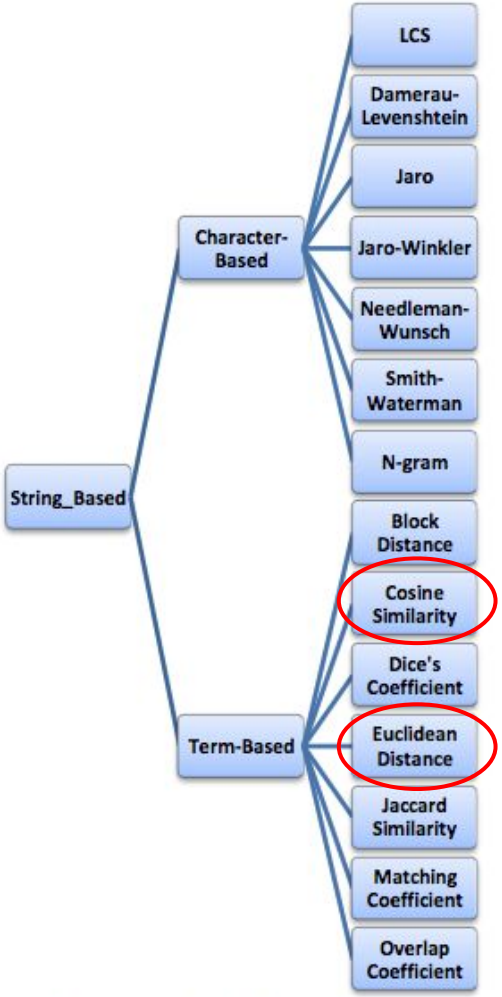
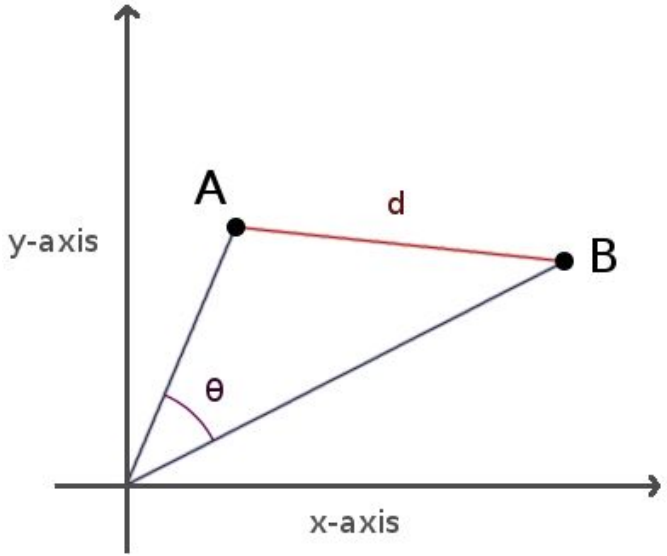


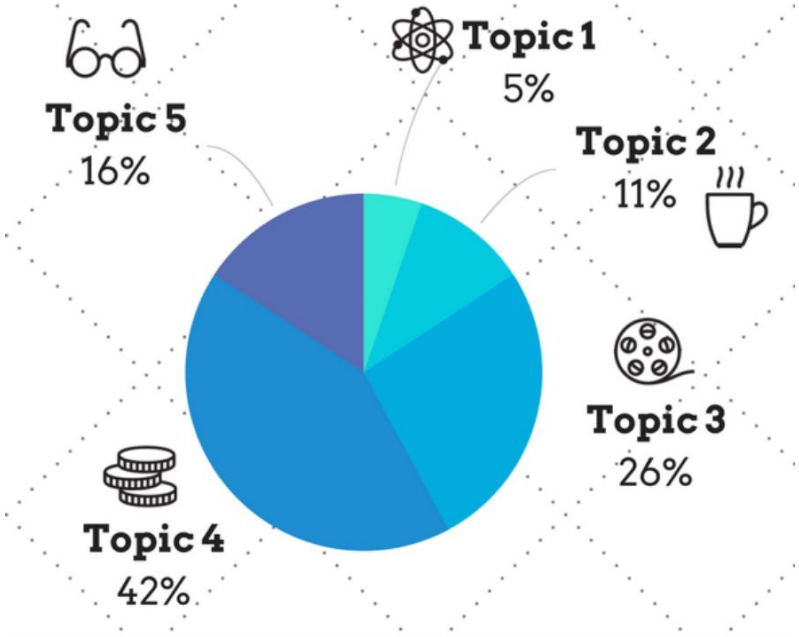
Fig 1: String-Based Similarity Measures



Credit: https://alexn.org/blog/2012/01/16/cosine-similarity-euclidean-distance.html

Topics

Themes found in our data. I.e. “restaurants”, “amenities”, “activities”



Credit: https://nlpforhackers.io/topic-modeling/

Brief History of Word Embeddings

- 17th century - philosophers such as Leibniz and Descartes put forward proposal for codes to relate words in different languages

- 1920s patent filed by Emanuel Goldberg
  - “Statistical Machine that searched for documents stored on film”

Credit: http://museen-dresden.de/index.php?lang=de&node=termine&resartium=events&tempus=week&locus=technischesammlungen&event=2680
Resource: https://en.wikipedia.org/wiki/History_of_natural_language_processing

Brief History of Word Embeddings

- 1945 - Vannevar Bush - Inspired by Goldberg
  - Desire for collective “memory” machine to make knowledge accessible

Credit: https://en.wikipedia.org/wiki/Vannevar_Bush

Themes In History - NLP

- Machine Translation
- Information Retrieval
- Language Understanding

“You shall know a word by the company it keeps—J. R. Firth (1957)”

Distributional hypothesis

"You shall know a word by the company it keeps—J. R. Firth (1957)"

*Words that occur in **similar contexts** tend to have **similar meanings***
(Harris, 1954)

Similar Meaning

- *Words that occur in similar contexts tend to have **similar meanings***

Type of "Similarity"	**Definition**	**Examples**
Semantic Relatedness	Any relation between words	Car, Road Bee, Honey
Semantic Similarity	Words used in the same way	Car, Auto Doctor, Nurse

Many more in Computational Linguistics!

Resource: https://arxiv.org/pdf/1003.1141.pdf

Context

- *Words that occur in similar **contexts** tend to have similar meanings*

- What is context?
  - A word?
  - A sentence?
  - A whole document?
  - A group (or window) or words?
  - ...

Similar

- *Words that occur in **similar** contexts tend to have similar meanings*

- What does ‘similar context’ mean mathematically?
  - Count words that appear together in context?
  - Should we count words within the context]?
  - Count how far apart they are?
  - Weight them?
  - …
- Thinking about “context”, “similar”, “meaning” in so many ways leads to evolution of different word embeddings

- ***Words*** *that occur in similar contexts tend to have similar meanings*

- What is a ***word??***
  - Words when printed are letters surrounded by white space and/or end of sentence punctuations (, . ? !)
  - Words are combined to form **sentences** that follow language rules

- What is a **sentence??**

EASY!

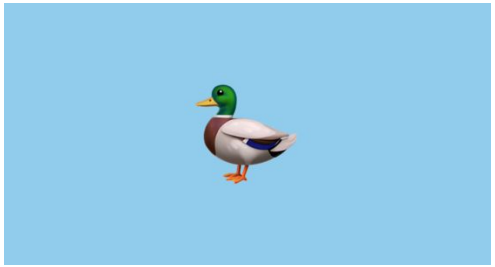
Separate the text by ‘.’ , ‘!’, ‘?’

Dr. Ford did not ask Col. Mustard the name of Mr. Smith’s dog.

Data Preprocessing - Clean and Tokenize

- Very important - your results may change drastically
- Tokenize - to split text into “tokens”
  - Required for gensim
- To keep or not to keep:
  - Numbers
  - Punctuation
  - Stop words (Common words - no universal list)
  - Sparse words
  - HTML tags
  - ...
- Other languages may tokenize differently!

- “i made her duck”



Credit: https://emojipedia.org/duck/
https://design.tutsplus.com/tutorials/how-to-animate-a-character-throwing-a-ball--cms-26207

Named Entity Extraction - Annotate Example

- “i love my apple”

Credit: https://www.dabur.com/realfruitpower/fruit-juices/apple-fruit-history
https://www.apple.com/shop/product/MK0C2AM/A/apple-pencil-for-ipad-pro

Data Preprocessing - Annotate

- Annotate
  - Part of speech (Noun, Verb, etc)
  - Named entity recognition (Organization, Money, Time, Locations, etc.)
- Choose to append, keep or ignore

More at https://spacy.io/usage/linguistic-features#section-named-entities

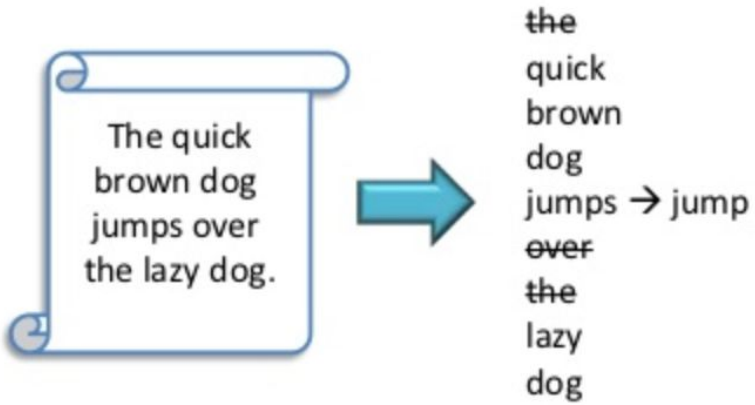
Data Preprocessing - Reduce Words

- Stemming
  - Reduce word to “word stem”
  - Crude heuristics to chop off word endings
  - Many approaches - Porter Stemming in Gensim
  - Fast!
  - Running -> run
- Lemmatize
  - Properly reduce the word based on part of speech annotation
  - Available in gensim
  - Slow
  - Better -> good
- Both differ based on language!

Resource: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Preprocessing Example

- Remove stop words and stem



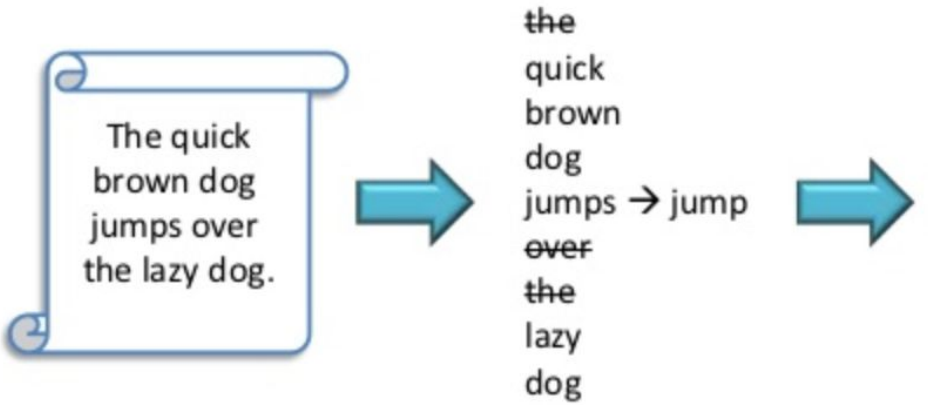
Credit: https://www.slideshare.net/mgrcar/text-and-text-stream-mining-tutorial-15137759

Embeddings

- Word
- Context (sentence, document, etc)

One-Hot-Encoding

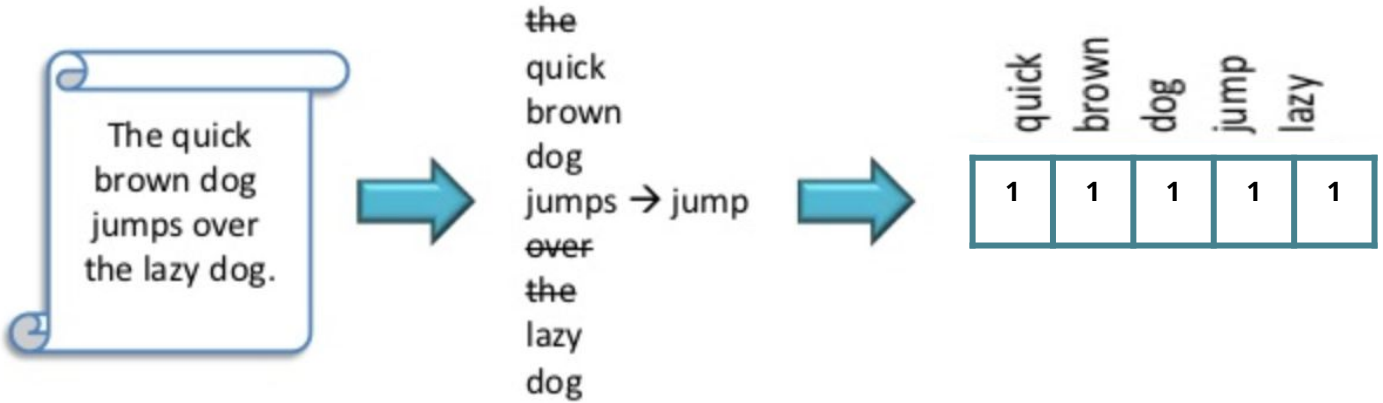
- Convert words to vector - available in scikit-learn



quick	1	0	0	0	0
brown	0	1	0	0	0
dog	0	0	1	0	0
jump	0	0	0	1	0
lazy	0	0	0	0	1

Boolean Embedding

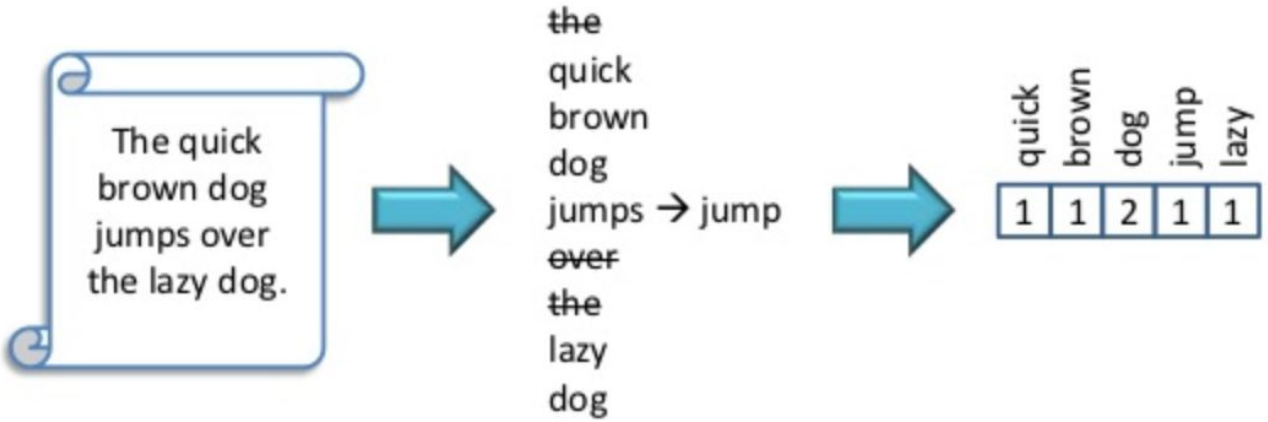
- Convert document to vector - 1 if the word exists, 0 otherwise
- Available in scikit-learn



Credit: https://www.slideshare.net/mgrcar/text-and-text-stream-mining-tutorial-15137759

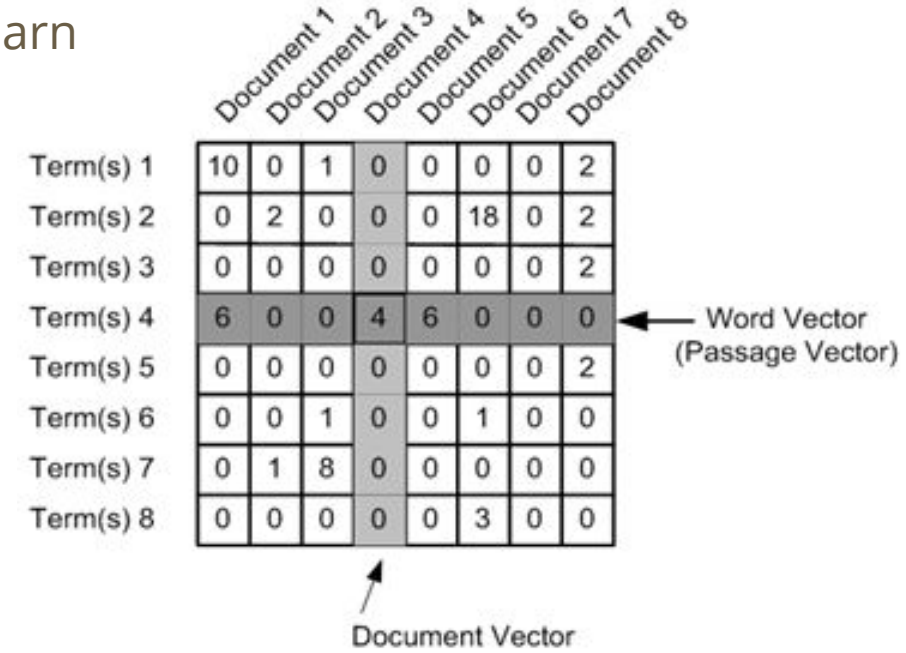
Bag of Words

- “Term Frequency” - count the frequency of the word in a document
- “Count Vectorizer” in scikit-learn



Document Term Matrix

- Term frequency for multiple documents
- “Count Vectorizer” in scikit-learn

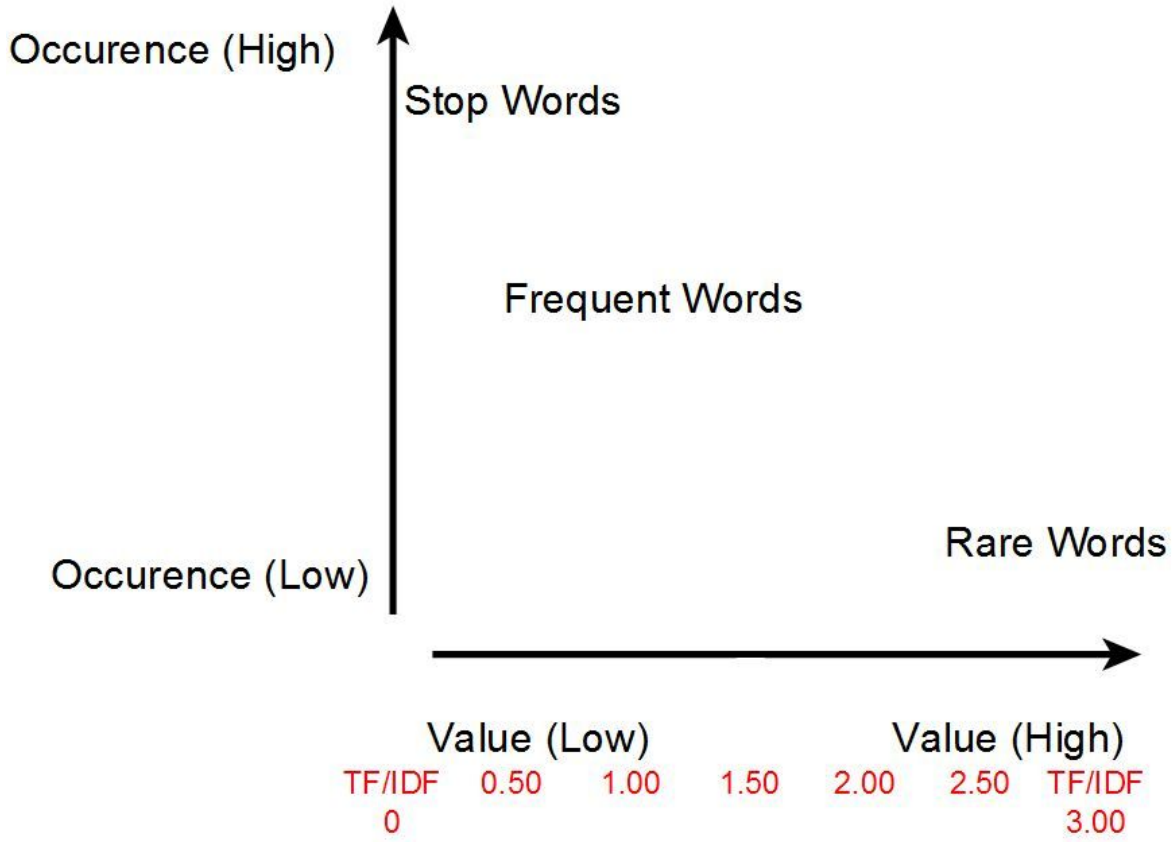


Credit: http://ryanheuser.org/word-vectors-2/

Weighting

- Why do we want to weight words?
  - “The”, “and”, ...

- Term Frequency Inverse Document Frequency (TFIDF)
  - Reduce the weight of very common words that appear in many of the documents
  - Applies to Document-Term Matrix



Credit: http://trimc-nlp.blogspot.com/2013/04/tfidf-with-google-n-grams-and-pos-tags.html

Word Co-Occurrence Matrix

Word-Word Matrix for a given Context Window (# words to consider on each side).

Word Co-Occurrence Matrix

Context Window Size = 1

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

The resulting counts matrix will then be:

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Credit: https://towardsdatascience.com/word-to-vectors-natural-language-processing-b253dd0b0817

Weighting

- Why do we want to weight word pairs?
  - “New” York” vs “in” “the”

- Pointwise Mutual Information (PMI)
  - Higher weight for mutually common words that are infrequent

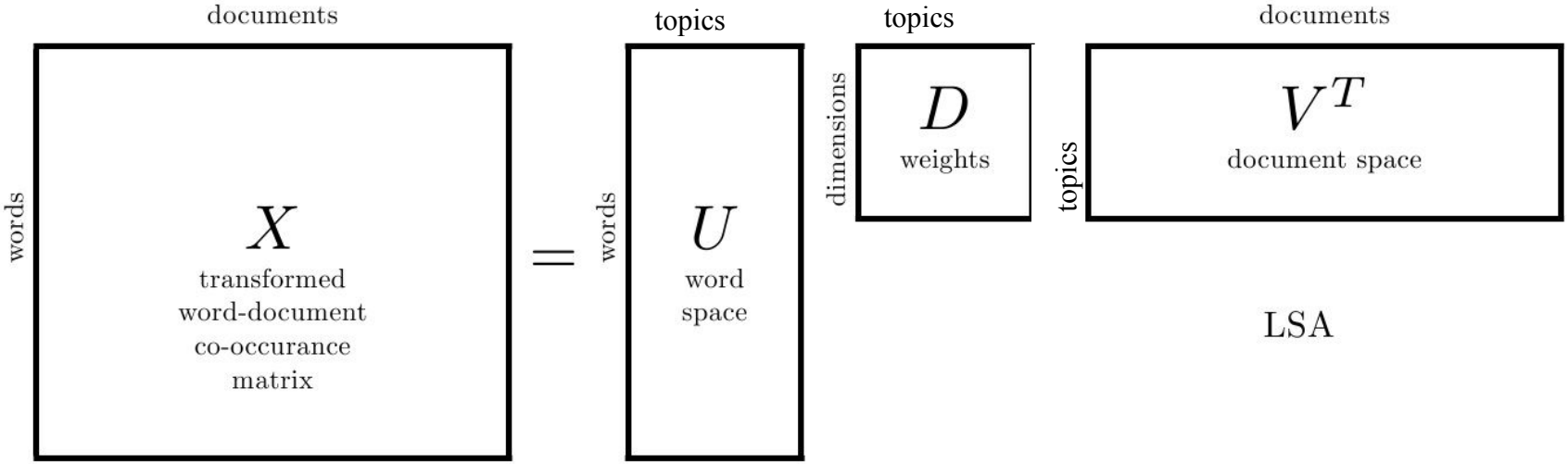
Pointwise Mutual Information (PMI)

word 1	*word 2*	*count word 1*	*count word 2*	count of co-occurrences	PMI
star	trek	8264	1594	1489	8.63974676575
car	driver	5578	2749	1384	8.41470768304
it	the	283891	3293296	3347	-1.72037278119
are	of	234458	1761436	1019	-2.09254205335

Credit: https://en.wikipedia.org/wiki/Pointwise_mutual_information

Topic Embedding - Latent Spaces

One approach: **Latent Semantic Analysis** (LSA) - Singular Value Decomposition (SVD) on the Document-Term Matrix or TFIDF Weighted Matrix



Credit: https://tex.stackexchange.com/questions/258811/diagram-for-svd

Topic Embeddings

- LSA/LSI - Latent Semantic Analysis or Indexing
  - Used for Search and Retrieval
  - Can only capture linear relationships
  - Use Non-Negative Matrix Factorization for “understandable” topics

- LDA (Latent Dirichlet Allocation)
  - Can capture non-linear relationships
- Guided LDA (Semi-Supervised LDA)
  - Seed the topics with a few words!

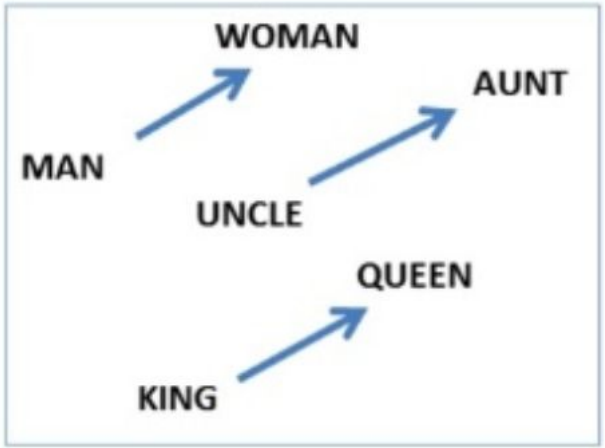
Prediction Based Embeddings

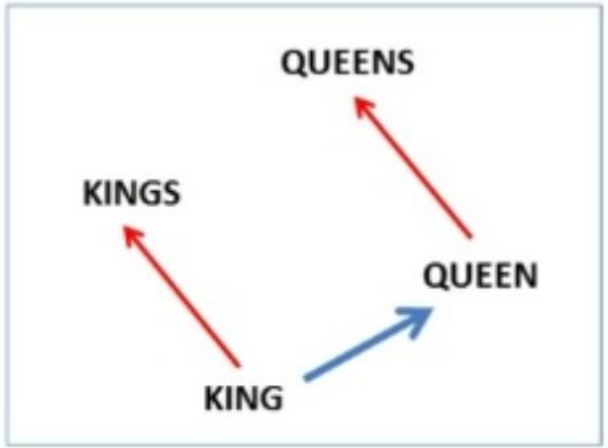
- Frequency Based -> Prediction Based
- Neural architecture to train
- Predict words based on other words (or characters!)

Neural Word Embedding - Word2Vec

vec(“king”) - vec(“man”) + vec(“woman”) = vec(“queen”)

Trained on Google News Data - 3 million words and phrases.





Credit: https://www.slideshare.net/dnldimitri/student-lecture-for-master-course-on-d-dimitrihermans

Popular Neural Word Embeddings by Mikolov

- **Word2Vec** (2013) - Better at **semantic similarity**
  - *brother : sister*

- **fastText** - Better at **syntactic similarity** due to character n-grams
  - *great : greater*

- Similar architecture for both - one trained on words, the other on character n-grams

Resources: https://en.wikipedia.org/wiki/Word2vec

Pretrained Embeddings

- Quickly leverage pretrained embeddings trained on different data sets (google news, wikipedia, etc)
- Many available in baseline gensim package
- To gain deeper domain specific insights - train your own model !

- Additional models available to download - different languages, etc
  https://github.com/Hironsan/awesome-embedding-models

Popular Neural Word Embedding

- GloVe (2014) - by Pennington, Socher, Manning
  - Closest to a variant of word co-occurrence matrix
  - Pre-trained model available in gensim
- Comparison of Word2Vec and GloVe by Radim Řehůřek's (gensim author)

Resource: https://nlp.stanford.edu/projects/glove/

Breaking News!

- Omar Levy proves Word2Vec is basically SVD on Pointwise Mutual Information (PMI) weighted Co-Occurrence word matrix!

- Levy NIPS 2014 - “Neural Word Embeddings as Implicit Matrix Factorization”
- Chris Moody @ StichFix - Oct 2017 - “Stop Using Word2Vec”

Credit: http://www.keepcalmandposters.com/poster/5718140_keep_calm_and_show_me_the_data

Why Word Embeddings?

- Too much text data!

- I want to know

Credit: https://blog.beeminder.com/allthethings/

Data & Packages

HomeAway

gensim
topic modelling for humans

scikit learn

python

jupyter

elastic

Credit: https://disneyworld.disney.go.com/entertainment/magic-kingdom/move-it-shake-it-dance-play-it
https://pragmaticarchitect.files.wordpress.com/2013/06/mabi87.png

Example of Data Preprocessing

```
review_data_list[0]
```

Last executed 2018-03-06 01:56:55 in 3ms

```
"nice experience this was our first experience renting a house, we were skeptical, but very pleasantly surprised.  th
e house was very clean and completely stocked.  it is just about 10 minutes to disney and maybe 15 to international d
rive.  the area was very quiet, but it was off season, i don't know what it'd be like during holiday time.  the pool
was heated to a comfortable temperature, but it's small.  also if you have small kids the pool is too deep for them a
t the shallow end, we would not be able to let our little ones swim there. all in all, it was a nice experience, and
we would do it again.  and, the price was more than fair."
```

```
print(preprocess_string(review_data_list[0], CUSTOM_FILTERS))
```

Last executed 2018-03-06 01:57:52 in 4ms

```
['nice', 'experi', 'experi', 'rent', 'hous', 'skeptic', 'pleasantli', 'surpris', 'hous', 'clean', 'complet', 'stock',
'10', 'minut', 'disnei', 'mayb', '15', 'intern', 'drive', 'area', 'quiet', 'season', 't', 'know', 'd', 'like', 'holid
ai', 'time', 'pool', 'heat', 'comfort', 'temperatur', 's', 'small', 'small', 'kid', 'pool', 'deep', 'shallow', 'end',
'abl', 'let', 'littl', 'on', 'swim', 'nice', 'experi', 'price', 'fair']
```

Search Example

```
new_doc = "I want to be close to disneyworld and stay in a clean house"
vec_bow = dictionary.doc2bow(preprocess_string(new_doc, CUSTOM_FILTERS))
vec_lsi = lsi[vec_bow] # convert the query to LSI space
```

Last executed 2018-03-08 18:28:27 in 4ms

```
sims = index[vec_lsi]
sims = sorted(enumerate(sims), key=lambda item: -item[1])
```

Last executed 2018-03-08 18:28:28 in 77ms

```
print('\n\n'.join([str(each[1])+ ' : ' + str(review_data_list[each[0]]) for each in sims[0:5]]))
```

Last executed 2018-03-08 18:36:31 in 123ms

```
0.8907449 : very clean house was very clean would recommend this house to anyone.had everything that you would need.
close to everything you would like to do

0.8863686 : the house was very clean and very close to disney the house had all the emenities we needed for 2 familie
s. pool was great.

0.8365079 : close to everything the house is comfy. we was really at the parks for our time but the house is really c
lose to everything.

0.8257486 : great house.   great house.  very clean & close to everything.

0.8239716 : great house we stayed 9 beatiful days in the house, it has all we need.it's close to the attractions and
all the comoditties.we will return.
```

Code to Create Embeddings

```
w2v_3_model = gensim.models.Word2Vec(tokentext, size=100, seed=42, min_count=5, window=3, workers=8)
```

Last executed 2018-03-08 21:05:19 in 8.46s

```
from gensim.models import fasttext
gensim_fasttext_3_model = fasttext.FastText(sentences=tokentext, sg=1, hs=0, size=100,
                                            alpha=0.05, window=3, min_count=5,
                                            max_vocab_size=None, word_ngrams=1,
                                            sample=0.001, seed=42, workers=8,
                                            min_alpha=0.0001, negative=5,
                                            cbow_mean=1,  iter=5, null_word=0, min_n=3,
                                            max_n=5, sorted_vocab=1, bucket=2000000,
                                            trim_rule=None, batch_words=10000)
```

Last executed 2018-0

> *The size of the sliding window has a strong effect on the resulting vector similarities. Large windows tend to produce more topical similarities […], while smaller windows tend to produce more functional and syntactic similarities.*

— Page 128, Neural Network Methods in Natural Language Processing, 2017.

Models Created

- Word2Vec - window size = 3, window size = 10
- fastText - window size = 3, window size = 10

fastText FTW

```
gensim_fasttext_3_model.wv.most_similar("disny")
```

Last executed 2018-03-08 21:08:51 in 5ms

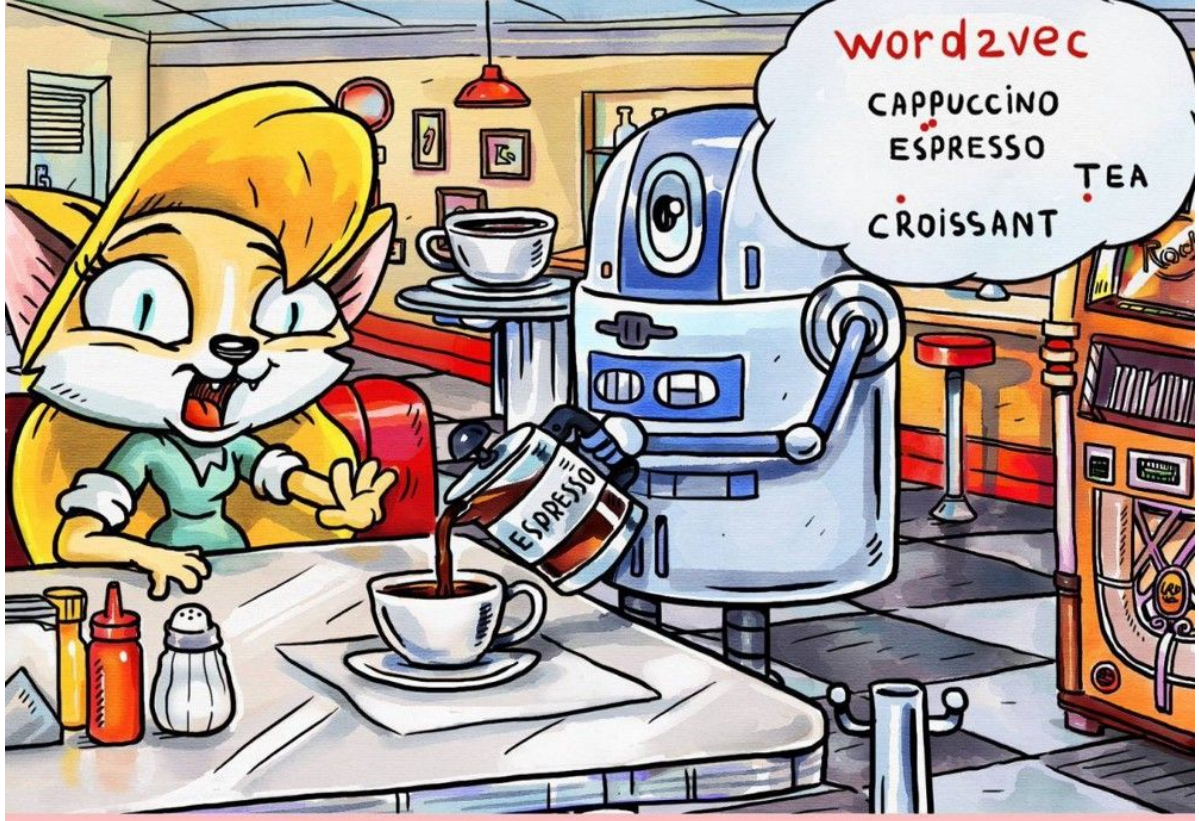
```
[('disneyworld', 0.8671283721923828),
 ('disnei', 0.8629084229469299),
 ('disneypark', 0.8584001064300537),
 ('disneyland', 0.7954359054565430),
 ('wdw', 0.6515198945999146),
 ('godisneyvac', 0.6241835355758667),
 ('walt', 0.6233668327331543),
 ('univer', 0.6085501313209534),
 ('dinsei', 0.5904804468154907),
 ('amus', 0.5838533639907837)]
```

* “most_similar” presents ordered words, similarity score

```
w2v_3_model.wv.most_similar("disny")
```

Last executed 2018-03-08 21:08:45 in 239ms

```
/usr/local/anaconda/lib/python3.5/site-packages/
_similar` (Method will be removed in 4.0.0, use
  if __name__ == '__main__':

---------------------------------------------
KeyError                                  Traceb
<ipython-input-72-aa99fe277874> in <module>()
----> 1 w2v_3_model.most_similar("disny")

/usr/local/anaconda/lib/python3.5/site-packages/
   1300                    stacklevel=2
   1301                )
-> 1302                return func(*args, **kwa
   1303
   1304            return new_func1

/usr/local/anaconda/lib/python3.5/site-packages/
opn, restrict_vocab, indexer)
   1373        Refer to the documentation for `
   1374        """
-> 1375        return self.wv.most_similar(posi
   1376
   1377    @deprecated("Method will be removed

/usr/local/anaconda/lib/python3.5/site-packages/
e, topn, restrict_vocab, indexer)
    503                mean.append(weight * wor
    504            else:
--> 505                mean.append(weight * sel
    506                if word in self.vocab:
    507                    all_words.add(self.v

/usr/local/anaconda/lib/python3.5/site-packages/
    451            return result
    452        else:
--> 453            raise KeyError("word '%s' no
    454
    455    def most_similar(self, positive=None

KeyError: "word 'disny' not in vocabulary"
```



- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

Credit: https://medium.com/data-science-group-iitr/word-embedding-2d05d270b285

Where Can I Go?

```
w2v_3_model.wv.most_similar(
    preprocess_string("epcot", CUSTOM_FILTERS),
```

Last executed 2018-03-08 21:05:29 in 103ms

```
[('gatorland', 0.8041095733642578),
 ('legoland', 0.7663977742195129),
 ('aquatica', 0.7443041801452637),
 ('seaworld', 0.7292442321777344),
 ('hollywood', 0.7221823930740356),
 ('mk', 0.7191978693008423),
 ('univer', 0.7019450664520264),
 ('cape', 0.6963222026824951),
 ('airboat', 0.6873927116394043),
 ('discoveri', 0.6730793714523315),
 ('typhoon', 0.6678956151008606),
 ('parad', 0.6677874326705933),
```

```
gensim_fasttext_3_model.wv.most_similar(
    preprocess_string("epcot", CUSTOM_FILTERS),
```

Last executed 2018-03-08 21:08:44 in 109ms

```
[('kingdom', 0.7928781509399414),
 ('hollywood', 0.7928774356842041),
 ('mk', 0.7485172748565674),
 ('mgm', 0.7070378661155701),
 ('blizzard', 0.6959882974624634),
 ('studio', 0.6865368485450745),
 ('typhoon', 0.6746012568473816),
 ('anim', 0.6607571840286255),
 ('magic', 0.6596393585205078),
 ('kingdon', 0.6476495265960693),
 ('arabian', 0.6296646595001221),
 ('aquatica', 0.6243083477020264),
```

Find me some grub!

```
w2v_3_model.wv.most_similar(
    positive=['mcdonald']
             , topn=20)
```

Last executed 2018-03-08 22:54:11 in 6ms

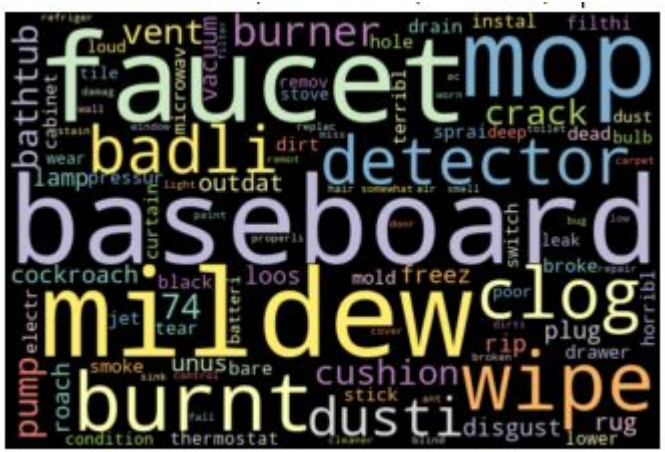
```
[('dunkin', 0.9355274438858032),
 ('dixi', 0.9258970022201538),
 ('supertarget', 0.9235088229179382),
 ('donut', 0.9197370409965515),
 ('walgreen', 0.9135407209396362),
 ('winn', 0.9109386205673218),
 ('chick', 0.9049642086029053),
 ('perkin', 0.9014164209365845),
 ('subwai', 0.8994849920272827),
 ('starbuck', 0.8964569568634033),
 ('applebe', 0.8952544927597046),
 ('chili', 0.8934187889099121),
 ('wawa', 0.8908361196517944),
 ('panera', 0.8903055787086487),
```

Clustering

- Clustered the word embeddings using KMeans clustering
- Showing the word size in wordcloud based on word weight from TFIDF
- Results from Word2Vec - Window Size = 3

What do people love?

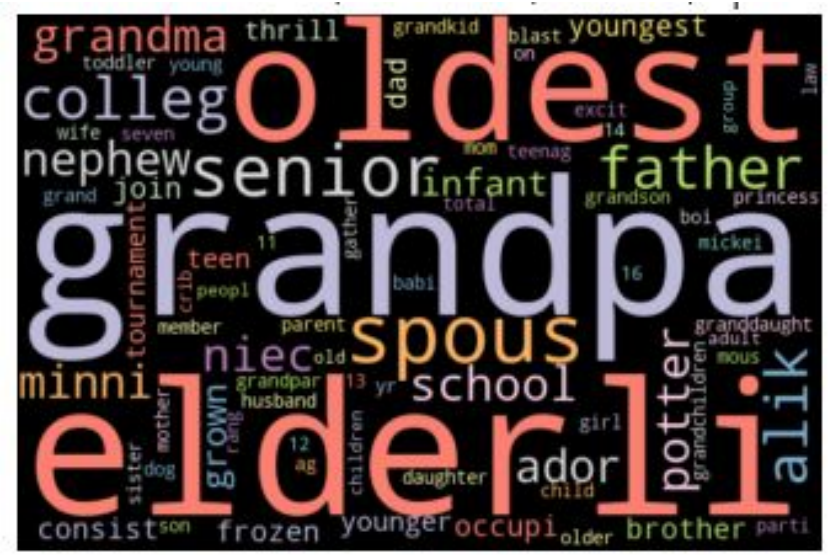
What do people hate?



How to stock the house?

What activities?

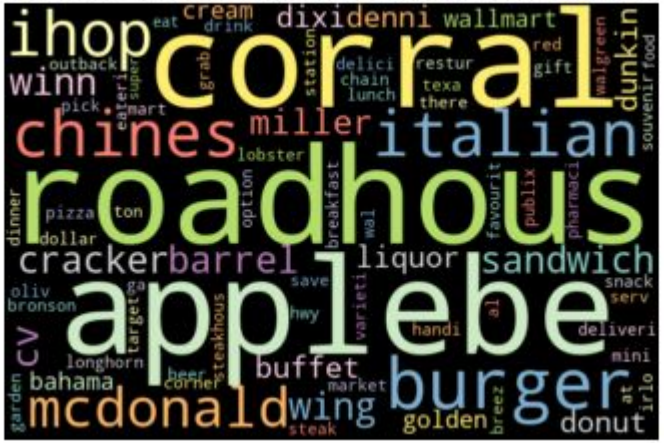
Who do people travel with?



When do they travel?

Window Size Matters!

Cluster 11 : WORD CLUSTER - Word2Vec(vocab=9273, size=100, alpha=0.025), Window = 10



> *The size of the sliding window has a strong effect on the resulting vector similarities. Large windows tend to produce more topical similarities […], while smaller windows tend to produce more functional and syntactic similarities.*

— Page 128, Neural Network Methods in Natural Language Processing, 2017.

Credit: https://machinelearningmastery.com/what-are-word-embeddings/

Surprise me!

Dirty Data!

Pop Quiz!

- What does it mean for words to be related?
  - Many things - semantic, syntactic, similar, related, etc
- What is an embedding and why do I care?
  - Way to convert text data into numbers so we can use computers to do the work
- Is an embedding really worth 1000 words?
  - It can be worth 10000 words! Based on how big the entire data set is
- What are some gotchas in dealing with text data?
  - Preprocessing, window size, and other hyperparameters in using Word2Vec or FastText
- What approaches can I use to find insights in my data?
  - Semantic Indexing, similar words, clustering, word2vec, fasttext

Thank you!
Questions?

Appendix

Embedding	Pros	Cons
Bag-of-Words	**Simple, fast**	**Sparse, high dimension** Does not capture position in text Does not capture semantics
TFIDF	**Easy to compute** Easily compare similarity between 2 documents	**Dense, high dimension** Does not capture position in text Does not capture semantics
Topic Space	Lower dimension Captures semantics Handles synonyms in documents **Used for search/retrieval**	**Number of topics needs to be defined** Could be slow in high dimension Topics need to be “hand labeled”
Word2Vec	Can leverage pretrained models **Understand relationships between words** Better for analogies	**Inability to handle unseen words** Active research - To go from word vectors to sentence vectors
fastText	Character based **Can deal with unseen words** Can leverage pretrained models	**Longer to train than Word2Vec** Active research - To go from word vectors to sentence vectors

Resources..

- https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html
- https://web.stanford.edu/class/cs124/
- https://datascience.stackexchange.com/questions/11402/preprocessing-text-before-use-rnn/11421
- https://nlp.stanford.edu/IR-book/pdf/12lmodel.pdf
- http://ruder.io/word-embeddings-1/index.html
- https://www.gavagai.se/blog/2015/09/30/a-brief-history-of-word-embeddings/
- https://www.ischool.utexas.edu/~ssoy/organizing/l391d2c.htm
- https://en.wikipedia.org/wiki/Information_retrieval
- https://en.wikipedia.org/wiki/As_We_May_Think
- https://en.wikipedia.org/wiki/Latent_semantic_analysis
- https://pdfs.semanticscholar.org/5b5c/a878c534aee3882a038ef9e82f46e102131b.pdf - “A survey of text similarity”
- http://www.jair.org/media/2934/live-2934-4846-jair.pdf

More Resources...

- https://cs224d.stanford.edu/lecture_notes/notes1.pdf
- https://www.quora.com/What-are-the-advantages-and-disadvantages-of-TF-IDF
- https://simonpaarlberg.com/post/latent-semantic-analyses/
- https://www.quora.com/What-are-the-advantages-and-disadvantages-of-Latent-Semantic-Analysis
- http://elliottash.com/wp-content/uploads/2017/07/Text-class-05-word-embeddings-1.pdf
- http://ruder.io/secret-word2vec/index.html#addingcontextvectors
- https://www.linkedin.com/pulse/what-main-difference-between-word2vec-fasttext-federico-cesconi/
- https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Highlight of Gensim Functions

- Gensim:
  - parsing.preprocessing
  - models.tfidfmodel
  - models.lsimodel
  - models.word2vec
  - models.fastText
  - models.keyedvectors
  - Descriptions at: https://radimrehurek.com/gensim/apiref.html
  - Tutorials at: https://github.com/RaRe-Technologies/gensim/tree/develop/docs/notebooks

Highlight of Other Python Functions

- Sklearn:
  - TfidfVectorizer
  - KMeans Clustering
  - Incredible documentation overall : http://scikit-learn.org/stable/index.html
- Wordcloud:
  - https://github.com/amueller/word_cloud/tree/master/wordcloud

Open Source Tool for Text Search

- Really Fast!!
- Built on Lucene - Apache Project - almost 20 years old and still evolving
- Lucene - set the standard for search and indexing

Example Code - Data Preprocessing

```
from gensim.parsing.preprocessing import strip_punctuation, strip_tags, strip_non_alphanum, remove_stopwords, STOPWORDS
from gensim.parsing.preprocessing import strip_multiple_whitespaces, preprocess_string, stem_text, split_alphanum
```

```
CUSTOM_FILTERS = [lambda x: x.lower(), strip_punctuation, strip_non_alphanum, split_alphanum, remove_stopwords,
                  strip_multiple_whitespaces, stem_text]
```

Last executed 2018-03-06 01:56:18 in 3ms

Example Code - TFIDF and Topic Embedding (LSA)

```
tfidf = TfidfVectorizer(analyzer=lambda x: x)
tfidf_data = tfidf.fit_transform(tokentext)
```

Last executed 2018-03-08 18:27:01 in 2.22s

```
from gensim import corpora, models, similarities
tfidf = models.TfidfModel(corpus, id2word=dictionary)
```

Last executed 2018-03-08 18:27:03 in 638ms

```
corpus_tfidf = tfidf[corpus]
```

Last executed 2018-03-08 18:27:03 in 102ms

```
lsi = models.LsiModel(corpus_tfidf, id2word=dictionary, num_topics=50)
```

Last executed 2018-03-08 18:27:12 in 8.51s

```
index = similarities.MatrixSimilarity(lsi[corpus_tfidf])
```

Last executed 2018-03-08 18:27:21 in 16.4s

Semantic vs Syntactic

Semantic

Syntactic

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Credit: https://arxiv.org/pdf/1301.3781v3.pdf

Fun Reviews - Amazon

Hutzler 571 Banana Slicer by Hutzler Manufacturing Co.

"What can I say about the 571B Banana Slicer that hasn't already been said about the wheel, penicillin, or the iPhone?"
Mrs Toledo

"Gone are the days of biting off slice-sized chunks of banana and spitting them onto a serving tray…. Next on my wish list: a kitchen tool for dividing frozen water into cube-sized chunks."
N. Krumpe

"As shown in the picture, the slices is curved from left to right. All of my bananas are bent the other way."
J. Anderson

Read more customer reviews