MCIT AWS Machine Learning Training

Use Cases – Amazon Products Reviews (Kaggle Dataset)

Sentiment Analysis

Amgad Zaki, AWS Senior Solutions Architect

amgadz@amazon.com

Use Case

We need to classify reviews as positive or negative for Kaggle Data Set about Amazon Products Reviews. The following are the assumptions:

- If using Comprehend, provide the review for only 1000 reviews (subset)

- If using a developed model, you are free to choose the size of dataset.

Use Case Data Set
Amazon product reviews dataset

~34,000 reviews for Amazon products like Kindle, Fire Tablet etc.

Sources: Amazon.com, BestBuy, Target, Ebay, Walmart

Review ratings: 1-5

Year range: 2010-2018

https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products

Use Case Data Set
Amazon product reviews dataset

reviews.rating	reviews.sourceURLs	reviews.text	reviews.title
5	http://reviews.bestbuy.c	This product so far has not disappointed. My children love to use it and I lik	Kindle
5	http://reviews.bestbuy.c	great for beginner or experienced person. Bought as a gift and she loves it	very fast
5	http://reviews.bestbuy.c	Inexpensive tablet for him to use and learn on, step up from the NABI. He	Beginner tablet for our 9 year old son.
4	http://reviews.bestbuy.c	I've had my Fire HD 8 two weeks now and I love it. This tablet is a great va	Good!!!
5	http://reviews.bestbuy.c	I bought this for my grand daughter when she comes over to visit. I set it u	Fantastic Tablet for kids
5	http://reviews.bestbuy.c	This amazon fire 8 inch tablet is the perfect size. I purchased it for my hus	Just what we expected
4	http://reviews.bestbuy.c	Great for e-reading on the go, nice and light weight, and for the price poin	great e-reader tablet
5	http://reviews.bestbuy.c	I gave this as a Christmas gift to my inlaws, husband and uncle. They lovec	Great for gifts
5	http://reviews.bestbuy.c	Great as a device to read books. I like that it links with my borrowed librar	Great for reading
5	http://reviews.bestbuy.c	I love ordering books and reading them with the reader.	Great and lightweight reader
4	http://reviews.bestbuy.c	Not easy for elderly users cease of ads that pop up.	nice tablet for the price
5	http://reviews.bestbuy.c	Excellent product. Easy to use, large screen makes watching movies and re	Excellent product
4	http://reviews.bestbuy.c	Wanted my father to have his first tablet and this is a very good value. He	Great Value
5	http://reviews.bestbuy.c	Simply does everything I need. Thank youAnd silk works wonders	Excellect
5	http://reviews.bestbuy.c	Got it as a present and love the size of the screen	Living It
5	http://reviews.bestbuy.c	The kindle is easiest to use, graphics and screen crisp, clear, brilliant color	Favorite of any tablet
4	http://reviews.bestbuy.c	nice reader. almost perfect for what i want/need. good bargain	good
4	http://reviews.bestbuy.c	I really like this tablet. I would have given 5 stars but sometimes you have	Nice Tablet for the Price
5	http://reviews.bestbuy.c	Great video quality lots of fun apps fun for the whole family	Great quality tablet
5	http://reviews.bestbuy.c	Love love love my kindle fire 8.....this is what my 9 yr old granddaughter sa	Kindle fire 8
5	http://reviews.bestbuy.c	Excellent tablet with nice screen. I wish Amazon would pre install the play	Excellent reader
5	http://reviews.bestbuy.c	Preloaded with the reading app from Kindle but expandable with other ap	Best Tablet Choice
5	http://reviews.bestbuy.c	Very happy with this product and easy to use..picture is clear, takes great	Great size..
5	http://reviews.bestbuy.c	My grandchildren are home schooled and utilize the tables for many learni	Great tablet for kids!
5	http://reviews.bestbuy.c	Great size, easy to carry for traveling. Need to spend more time Looking in	First Tablet. Lots of possibilities.

Important NLP terms

Corpus: Large collection of words or phrases. Like vocabulary or dictionary of a language.

- Corpus can come from different sources: Documents, web sources, database

Token: Words or phrases extracted from documents.

Feature vector: A numeric array that ML models use for training and classification/regression tasks.

Text cleaning and special cases

We can encounter different types of texts in our data. We will need to handle them in the pre-processing step. For example:

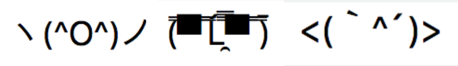
HTML and XML markup: <p> This is a dataset .. </ p>

Twitter mark-up (names, hash tags): #aws #amazon #tesla

Capitalization: Acronyms: NASA, ETA etc.

Phone numbers, dates: +14256565, 12/11/2013

Emojis: 

Emoticons: 

Stop Words

Manually excluded from the text, because they occur too frequently in all documents in the corpus.

There are 179 stop words in NLTK library:

```
{'so', 'yours', 'aren', 'hadn', 'those', "needn't", 'few', 'her', 'then', 'had', 'to', 've', "you'd", 'of', 'him', 'won', 'about', "should've",
'by', 'itself', 'if', 'theirs', 'd', "aren't", 'off', "wasn't", 'do', 'he', 'why', 'ourselves', "mightn't", "shan't", 'there', 'these', 'too',
'needn', 's', 'as', 'been', 'same', 'the', 'can', 'yourselves', "couldn't", 'against', 'now', 'above', 'until', 're', 'shouldn', 'both', 'who',
'most', 'not', 'has', 'once', 'during', "doesn't", 'shan', 'this', 'm', 'such', 'isn', 'we', 'them', 'that', 'each', 'only', 'yourself', 'were',
'at', "wouldn't", 'wasn', 'his', 'himself', 'on', 'again', 'more', "didn't", 'how', 'y', 'our', "it's", 'themselves', "hadn't", 'between', 'after',
'or', 'under', 'be', 'didn', "don't", 'into', 'where', 'ma', 'couldn', "weren't", 'over', 'don', 'an', 'its', 'some', 'doesn', 'my', 'being',
'hasn', "mustn't", 'o', 'own', 'here', 'whom', 'have', 'up', 'ours', 'out', 'hers', "isn't", 'mustn', 'ain', 'll', 'herself', 'should', 'i', 'and',
'from', 'will', 'with', 'myself', 'are', 'other', 'she', 'does', 'was', 'down', "haven't", 'when', 'nor', 'before', 'your', 'which', 'weren', 'no',
'they', 'is', 'mightn', 'further', 'below', 'their', 'but', 'in', 'just', 'a', "hasn't", 'you', "you've", 'haven', 'me', 'all', 'it', 'because',
'for', 'any', 't', 'what', 'am', "that'll", 'very', "you're", "she's", 'while', 'through', "shouldn't", 'wouldn', 'than', "you'll", 'doing', 'did',
'having', "won't"}
```

NLP-Tokenizing

Separating text data into tokens by white space and punctuation as token separators.

Sentence: “I don’t like eggs.”

- **Tokens: “I”, ”don’t”, “like”, “eggs”, “.”**

Tokens: Individual pieces of information from the raw sentence.

Tokens can be further processed depending on their importance.

Stemming or Lemmatization

Stemming: Set of rules to slice a string to a substring. The goal is to remove word affixes (particularly suffixes).

For example:

- Removing "s", "es" which generally indicates plurality.
- Removing past tense suffixes: "ed"

Lemmatization: It looks up words in dictionary and returns the "head" word called a "lemma."

- It is more complex than stemming.
- For best results, word position tags should be provided: Adjective, noun, ..

Stemming and Lemmatization

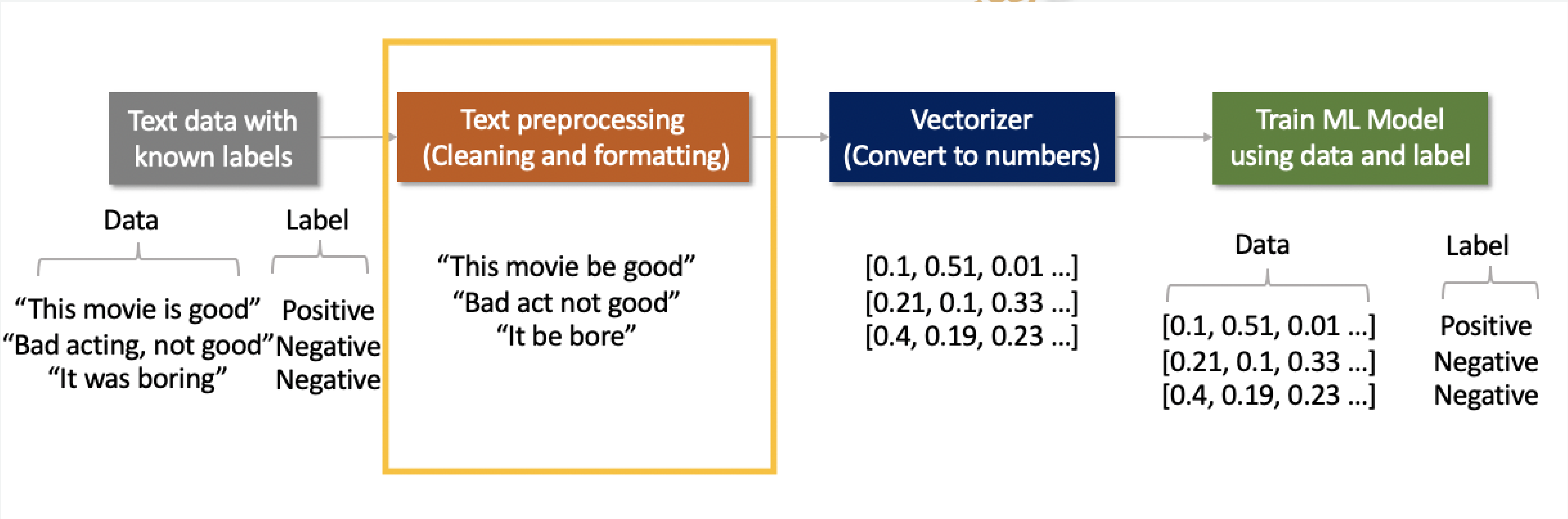
Example: "the children are playing and running. the weather was better yesterday."

Stemming => "the children are **play** and **run**. the weather was better yesterday"

Lemmatizing => "the **child** are playing and running. the weather **wa** better yesterday"

Lemmatizing with pos. tags=> "the **child be play** and **run**. the weather **be good** yesterday"

NLP Pipeline – Training - Movie Reviews



NLP - Vectorization

ML algorithms expect numeric vectors as inputs instead of texts

This transformation is called **vectorization** or **feature extraction**.

Similar meanings will be **closer** to each other in this space.

We will introduce **Bag of Words (BoW)** representation here.

Bag of Words

Each document is represented by a vector with size equal to the size of the corpus (vocabulary).

Each entry is the number of times the corresponding word occurred in the sentence (raw counts method).

Sentence	acting	bad	boring	good	it	movie	was
Good movie	0	0	0	1	0	1	0
Bad acting	1	1	0	0	0	0	0
It was boring movie	0	0	1	0	1	1	1

Bag of Words

Each document is represented by a vector with size equal to the size of the corpus (vocabulary).

Each entry is the number of times the corresponding word occurred in the sentence (raw counts method).

Vocabulary

Sentence	acting	bad	boring	good	it	movie	was
Good movie	0	0	0	1	0	1	0
Bad acting	1	1	0	0	0	0	0
It was boring movie	0	0	1	0	1	1	1

Issues:
1-We lost information inherent in the word order.
2-Large documents can have big word counts compared to small docs.

1-Keeping the order: N-grams

Let's keep some order information with N-grams: "**N**" consecutive words in a text.

N-grams: **"This movie is good"**

- 1-gram: "this", "movie", "is", "good"
- 2-grams: "this movie", "movie is", "is good"
- .....

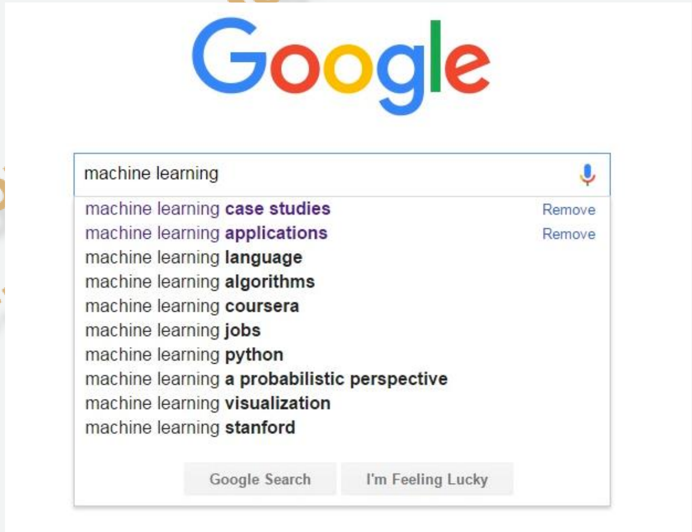
Let's apply to our data (N=2).

Sentence	good movie	bad acting	it was	… …	movie	was
Good movie	1	0	0	… …	1	0
Bad acting	0	1	0	… …	0	0
It was boring movie	0	0	1	… …	1	1

N-Grams

Use case for N-Grams:

- Text sequence prediction
- Phone number completion



Choosing N too large will cause the model to be too complex (higher order).

Different N-gram windows should be used to find the optimum window length when training a model.

2-Term Frequencies

Keeping the counts of occurrence will cause large numbers for certain words/phrases.

Long sentences or documents will have large numbers.

Representation	Formula
raw count	$f_{t,d}$
binary	0, 1
term frequency ($tf_{t,d}$)	$f_{t,d}$ / (# total terms in d)

t: Term,
d: Document

Term Frequencies

Binary: Each token is either one or zero depending on existence in the document.

Sentence	acting	bad	boring	good	it	movie	was
Good movie	0	0	0	1	0	1	0
Bad acting	1	1	0	0	0	0	0
It was boring movie	0	0	1	0	1	1	1

Term Frequency: Token counts divided by total number of tokens in the document.

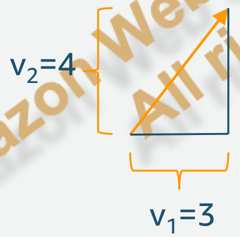
Sentence	acting	bad	boring	good	it	movie	was
Good movie	0	0	0	0.5	0	0.5	0
Bad acting	0.5	0.5	0	0	0	0	0
It was boring movie	0	0	0.25	0	0.25	0.25	0.25

Vector Normalization

Normalized vectors have length (norm) 1.

Divide the vector by its length. After normalization, vector becomes unit vector.

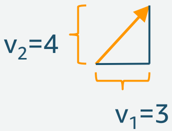
$\vec{v}$ = [3, 4] => v_1=3 and v_2=4



$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2} = \sqrt{3^2 + 4^2} = 5$$

Vector Normalization

Normalized vectors have length (norm) 1.
Divide the vector by its length. After normalization, vector becomes unit vector.
$\vec{v}$ = [3, 4] => v_1=3 and v_2=4

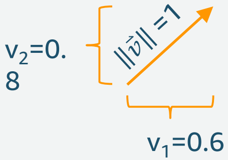


$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2} = \sqrt{3^2 + 4^2} = 5$$

Normalized vector => $\hat{v} = \frac{\vec{v}}{\|\vec{v}\|} = \frac{[3,4]}{5}$ = [3/5, 4/5]=[0.6 0.8]

$(\sqrt{0.6^2 + 0.8^2} = 1)$

This is also called **L2 normalization ($\|\vec{v}\|_2$)**



© 2020 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Normalizing Term Frequency Vector

Term Frequency: Token counts divided by total number of tokens in the document.

Sentence	acting	bad	borin g	good	it	movie	was
It was boring movie	0	0	0.25	0	0.25	0.25	0.25

$\vec{v}$ = [0, 0, 0.25, 0, 0.25, 0.25, 0.25]

$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + v_3^2 + v_4^2 + v_5^2 + v_6^2 + v_7^2} = \sqrt{0^2 + 0^2 + 0.25^2 + 0^2 + 0.25^2 + 0.25^2 + 0.25^2} = 0.5$$

$$\hat{v} = \frac{\vec{v}}{\|\vec{v}\|} = \frac{\vec{v}}{0.5} = \frac{[0, 0, 0.25, 0, 0.25, 0.25, 0.25]}{0.5} = [0, 0, 0.5, 0, 0.5, 0.5, 0.5]$$

Sentence	acting	bad	boring	good	it	movie	was
It was boring movie	0	0	0.5	0	0.5	0.5	0.5

Count Vectorizer

```python
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

texts = ["good movie", "bad acting", "it was boring movie"]

vectorizer = CountVectorizer()
vectorizer.fit(texts)
features = vectorizer.transform(texts)

df = pd.DataFrame(features.toarray(), columns=vectorizer.get_feature_names())

print("Texts:", texts)
print("--------------------------------------------------------")
print(df)
```

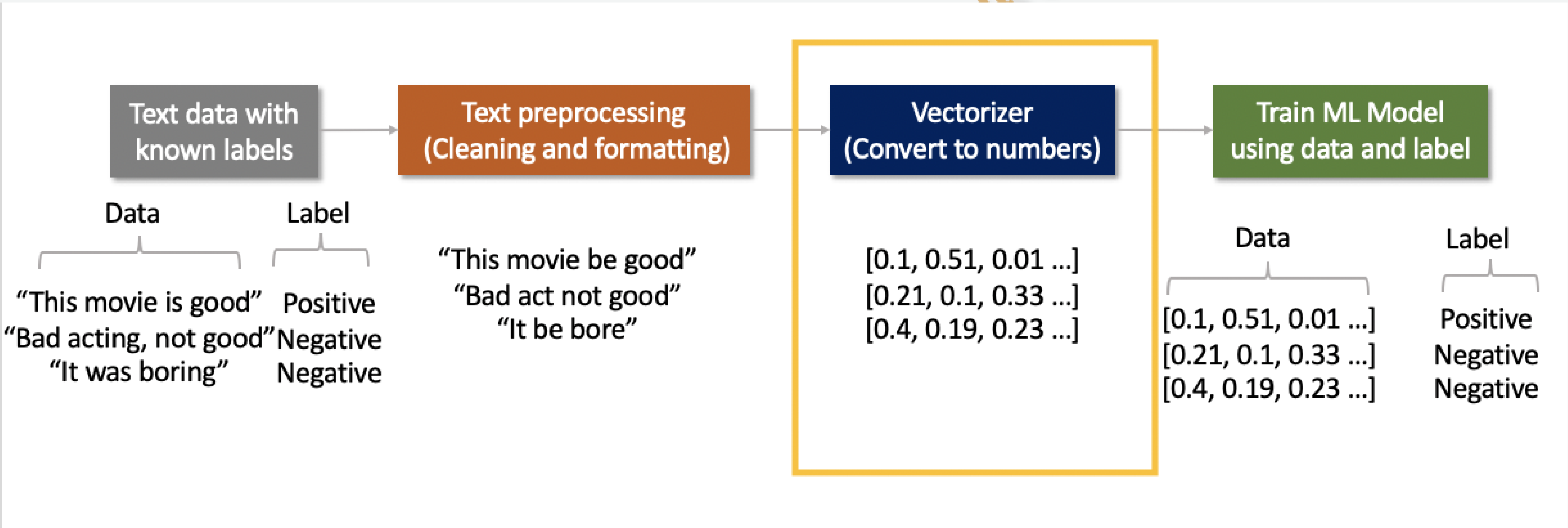
```
Texts: ['good movie', 'bad acting', 'it was boring movie']
--------------------------------------------------------
   acting  bad  boring  good  it  movie  was
0       0    0       0     1   0      1    0
1       1    1       0     0   0      0    0
2       0    0       1     0   1      1    1
```

Term Frequency

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

texts = ["good movie", "bad acting", "it was boring movie"]

vectorizer = TfidfVectorizer(use_idf=False)
vectorizer.fit(texts)
features = vectorizer.transform(texts)

df = pd.DataFrame(features.toarray(), columns=vectorizer.get_feature_names())

print("Texts:", texts)
print("--------------------------------------------------------")
print(df)
```

```
Texts: ['good movie', 'bad acting', 'it was boring movie']
--------------------------------------------------------
     acting       bad  boring      good   it     movie  was
0  0.000000  0.000000     0.0  0.707107  0.0  0.707107  0.0
1  0.707107  0.707107     0.0  0.000000  0.0  0.000000  0.0
2  0.000000  0.000000     0.5  0.000000  0.5  0.500000  0.5
```

NLP Pipeline – Training - Movie Reviews



Term Freq. - Inverse Document Freq.

So far, we only looked at local scale (document level). We also need to consider other documents in the corpus.

The main insight: Meaning is mostly encoded in more rare items in documents.

For example: In sports documents about basketball and soccer,

- We will mostly see words like “play”, “run”, “score.”
- These won’t be useful to distinguish between a basketball and soccer document.

Term Frequency - Inverse Document Frequency

N: total documents

$\mathbf{N_t}$: Number of documents with token/phrase "t" in it.

Document frequency:

$df_t = N_t/N$

Inverse document frequency:

Compute $1/df_t = N/N_t$. Then apply logarithm: $\log(N/N_t)$.

Sklearn also applies smoothing: $idf_t = \log[(N+1)/(N_t+1)] + 1$

Term Frequency Inverse Document Frequency (TFIDF):

$tf\_idf_{t,d} = tf_{t,d} \times idf_t$

A high **TFIDF** is reached by a **high** term frequency and a **high** inverse document frequency (**low** document frequency).

Term Frequency - Inverse Document Frequency

Term Frequency Inverse Document Frequency:

$$tf\_idf_{t,d} = tf_{t,d} \times idf_t$$

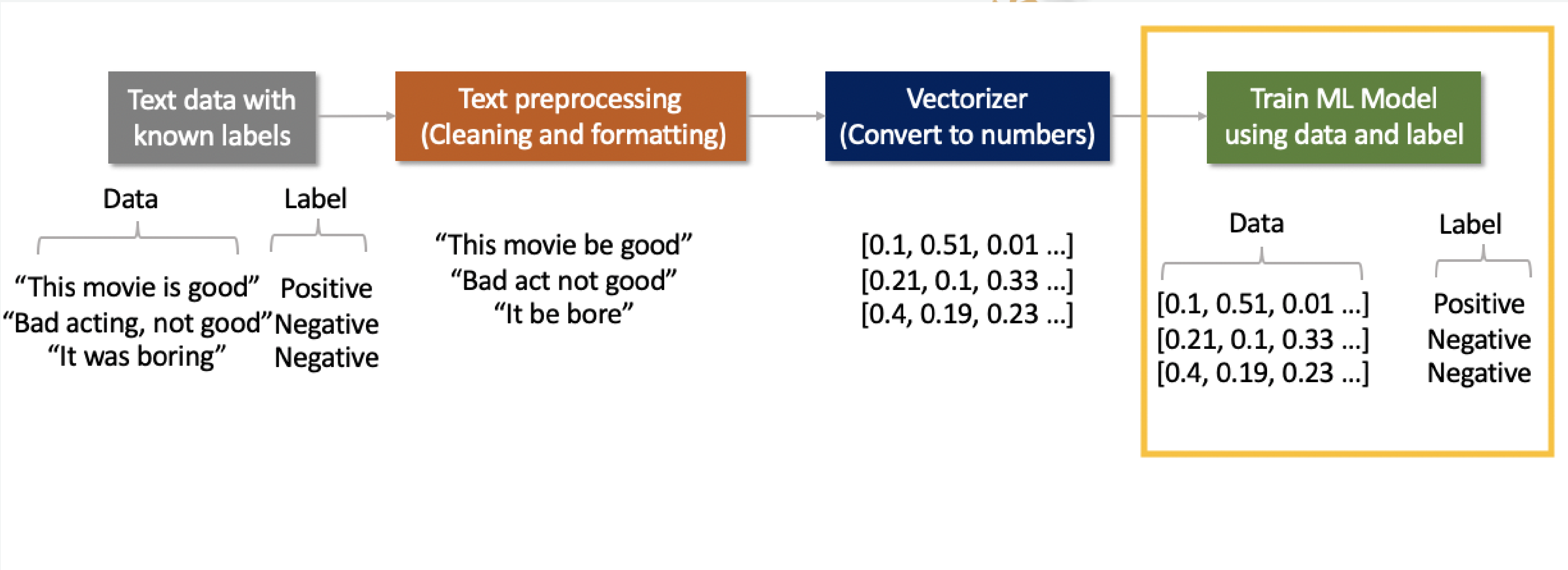
Sentence	acting	bad	boring	good	it	movie	was
Good movie	0	0	0	0.795961	0	0.605349	0
Bad acting	0.707107	0.707107	0	0	0	0	0
It was a boring movie	0	0	0.528635	0	0.528635	0.402040	0.528635

Term Frequency - Inverse Document Frequency

```python
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

texts = ["good movie", "bad acting", "it was boring movie"]

vectorizer = TfidfVectorizer()
vectorizer.fit(texts)
features = vectorizer.transform(texts)

df = pd.DataFrame(features.toarray(), columns=vectorizer.get_feature_names())

print("Texts:", texts)
print("------------------------------------------------------------")
print(df)
```

```
Texts: ['good movie', 'bad acting', 'it was boring movie']
---------------------------------------------------------
     acting       bad    boring      good        it     movie       was
0  0.000000  0.000000  0.000000  0.795961  0.000000  0.605349  0.000000
1  0.707107  0.707107  0.000000  0.000000  0.000000  0.000000  0.000000
2  0.000000  0.000000  0.528635  0.000000  0.528635  0.402040  0.528635
```

NLP Pipeline – Training - Movie Reviews



Naïve Bayes Classifier

Naïve Bayes: A generative model that approximates the data using Bayes' Theorem.

Bayes' Theorem:

Prob. of B given A

Prob. of A

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Prob. of A given B

Prob. of B

Sample Problems:

- Spam / not spam email
- Text topic classification: sports, politics, finance
- Opinion (sentiment): like, neutral, dislike

Bayes' Theorem

$$P(A,B) = P(B)\,P(A \mid B)$$

$$P(A,B) = P(A)\,P(B \mid A)$$

$$P(B)\,P(A \mid B) = P(A)\,P(B \mid A)$$

$$P(A \mid B) = \frac{P(A)\,P(B \mid A)}{P(B)}$$

Naïve Bayes Classifier - Example

Category classification: We have training texts that belong to categories: Finance and Not Finance.

We want to classify new texts using a Naïve Bayes classifier.

Text	Category
It was a clean game.	Not Finance
Oil companies lost over 25 millions yesterday.	Finance
He scored three goals.	Not Finance
Their 3 game winning streak ended yesterday.	Not Finance
The stock market started the day with profits.	Finance

Naïve Bayes Classifier - Example

Which tag does the following sentence belong to?

"Learn stock markets playing this game"

Text	Tag
It was a clean game.	Not Finance
Oil companies lost over 25 millions yesterday.	Finance
He scored three goals.	Not Finance
Their 3 game winning streak ended yesterday.	Not Finance
The stock market started the day with profits.	Finance
Learn stock markets playing this game	?

Naïve Bayes Classifier - Example

Pre-processing:

- Remove stop words and words shorter than 2 characters
- Apply stemming

Text	Processed Text
It was a clean game.	clean game
Oil companies lost over 25 millions yesterday.	oil compani lost million yesterday
He scored three goals.	score three goal
Their 3 game winning streak ended yesterday.	game win streak end yesterday
The stock market started the day with profits.	the stock market start day profit
Learn stock markets playing this game	learn stock market play game

Naïve Bayes Classifier - Example

We will calculate the probabilities:

P(Finance | “learn stock market play game”): Probability of the Finance tag, given the sentence: Learn stock markets playing this game

P(Not Finance | “learn stock market play game”): Probability of the Not Finance tag, given the sentence: Learn stock markets playing this game

We will assign a category to “learn stock market play game” based on whichever probability is larger.

Naïve Bayes Classifier - Example

Recall Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes Classifier - Example

Recall Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(Finance \mid \text{"learn stock market play game"}) = \frac{P(\text{"learn stock market play game"} \mid Finance) \; x \; P(Finance)}{P(\text{"learn stock market play game"})}$$

$$P(Not\ Finance \mid \text{"learn stock market play game"}) = \frac{P(\text{"learn stock market play game"} \mid Not\ Finance) \; x \; P(Not\ Finance)}{P(\text{"learn stock market play game"})}$$

Naïve Bayes Classifier - Example

Recall Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(Finance \mid \text{"learn stock market play game"}) = \frac{P(\text{"learn stock market play game"} \mid Finance) \; x \; P(Finance)}{\boxed{P(\text{"learn stock market play game"})}}$$

$$P(Not\ Finance \mid \text{"learn stock market play game"}) = \frac{P(\text{"learn stock market play game"} \mid Not\ Finance) \; x \; P(Not\ Finance)}{\boxed{P(\text{"learn stock market play game"})}}$$

We wish to know which of these two probabilities is larger.

The denominators are the same! So we don't need to compute the denominators to know which probability is larger.

Naïve Bayes Classifier - Example

$P(Finance \mid \underbrace{\text{"learn stock market play game"}}) \sim P(\text{"}\textbf{learn stock market play game}\text{"} \mid Finance)\ x\ P(Finance) \rightarrow \frac{2}{5}$

- Count how many times “learn stock market play game” is in the finance tag/number of sentences in finance

$P(Non\ Finance \mid \underbrace{\text{"learn stock market play game"}}) \sim P(\text{"}\textbf{learn stock market play game}\text{"} \mid Non\ Finance)\ x\ P(Non\ Finance) \rightarrow \frac{3}{5}$

- Count how many times “learn stock market play game” is in the nonfinance tag/number of sentences in nonfinance

Naïve Bayes Classifier - Example

$$P(Finance \mid \underbrace{\text{"learn stock market play game"}}) \sim P(\text{"}\mathbf{learn\ stock\ market\ play\ game}\text{"} \mid Finance)\ x\ P(Finance) \rightarrow \frac{2}{5}$$

✓ Count how many times “learn stock market play game” is in the finance tag/number of sentences in finance

$$P(Non\ Finance \mid \underbrace{\text{"learn stock market play game"}}) \sim P(\text{"}\mathbf{learn\ stock\ market\ play\ game}\text{"} \mid Non\ Finance)\ x\ P(Non\ Finance) \rightarrow \frac{3}{5}$$

✓ Count how many times “learn stock market play game” is in the nonfinance tag/number of sentences in nonfinance

Problem:

We don’t have any data with the exact sequence: “learn stock market play game.”

Text	Category
clean game	Not Finance
oil compani lost million yesterday	Finance
score three goal	Not Finance
game win streak end yesterday	Not Finance
the stock market start day profit	Finance

Naïve Bayes Classifier - Example

Solution: Be naïve! Assume every word is conditionally independent. ✓

$P(\text{"learn stock market play game"} \mid Finance)$=P(“learn”|Finance) x P(“stock”|Finance) x P(“market”|Finance) x P(“play”| Finance) x P(“game”|Finance)

$P(\text{"learn stock market play game"} \mid Not\ Finance)$=P(“learn”|Not Finance) x P(“stock”|Not Finance) x P(“market”|Not Finance) x P(“play”|Not Finance) x P(“game”|Not Finance)

We can easily calculate these probabilities using our data table!

Naïve Bayes Classifier - Example

$P(\text{"}learn\ stock\ market\ play\ game\text{"}|Finance)$ = P("learn"|Finance) x P("stock"|Finance) x P("market"|Finance) x P("play"| Finance) x P("game"|Finance)

P("learn"|Finance): 0/11 = 0
P("stock"|Finance): 1/11 = 0.091
P("market"|Finance): 1/11 = 0.091
P("play"|Finance): 0/11 = 0
P("game"|Finance): 0/11 = 0

Even one zero probability will make the whole product zero!

Text	Category
clean game	Not Finance
oil compani lost million yesterday	Finance
score three goal	Not Finance
game win streak end yesterday	Not Finance
the stock market start day profit	Finance

Naïve Bayes Classifier - Example

$P("learn\ stock\ market\ play\ game"|Finance)$ = P("learn"|Finance) x P("stock"|Finance) x P("market"|Finance) x P("play"| Finance) x P("game"|Finance)

Apply Laplace Smoothing: Add 1 to numerator and add total number of distinct words to denominator.

P("learn"|Finance): (0+1)/(11+19) = 0.0333
P("stock"|Finance): (1+1)/(11+19) = 0.0666
P("market"|Finance): (1+1)/(11+19) = 0.0666
P("play"|Finance): (0+1)/(11+19) = 0.0333
P("game"|Finance): (0+1)/(11+19) = 0.0333

Text	Category
clean game	Not Finance
oil compani lost million yesterday	Finance
score three goal	Not Finance
game win streak end yesterday	Not Finance
the stock market start day profit	Finance

$P("learn\ stock\ market\ play\ game"|Finance) = 1.638 \times 10^{-7}$

Naïve Bayes Classifier - Example

$P(Finance|"learn\ stock\ market\ play\ game") \approx\ P("learn\ stock\ market\ play\ game"|Finance)$ x P(Finance)

$$= 1.6378 \times 10^{-7} \cdot \frac{2}{5} = \mathbf{0.655 \times 10^{-7}}$$

$P(Not\ Finance|"learn\ stock\ market\ play\ game") \approx P("learn\ stock\ market\ play\ game"|Not\ Finance)$ x P(Not Finance)

$$= 1.4656 \times 10^{-7} \cdot \frac{3}{5} = \mathbf{0.879 \times 10^{-7}}$$

$P(Not\ Finance|"learn\ stock\ market\ play\ game") > P(Finance|"learn\ stock\ market\ play\ game")$

NOT a finance sentence!

Sentiment Analysis

Classifying the polarity of a given text.

Positive or Negative Movie Reviews?:

- Unbelievably disappointing
- Full of funny characters and richly applied satire, and some great plot twists
- This is the greatest comedy ever filmed
- Poor acting. It was pathetic.

Sentiment Analysis

Simplest task:

- Is the attitude of this text positive or negative?

More complex:

- Rank the attitude of this text from 1 to 5

Advanced:

- Detect the target, source, or complex types

Different types (targets)

Emotion:
- Angry, sad, joyful, fearful, ashamed, proud, elated

Mood:
- Cheerful, gloomy, irritable, listless, depressed, buoyant

Interpersonal stances:
- Friendly, flirtatious, distant, cold, warm, supportive, contemptuous

Attitudes:
- Liking, loving, hating, valuing, desiring

Personality traits:
- Nervous, anxious, reckless, morose, hostile, jealous

Why sentiment analysis?

Movie: is this review positive or negative?

Products: what do people think about the new Alexa device?

Public sentiment: how is consumer confidence? Is despair increasing?

Politics: what do people think about this candidate or issue?

Sentiment Lexicons

NLTK has two main sentiment data sources:

- **Bing Liu Opinion Lexicon**

from nltk.corpus import *opinion_lexicon*

- **SentiWordNet**

from nltk.corpus import *sentiwordnet*

Bing Liu Opinion Lexicon

Page:

https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

Data:

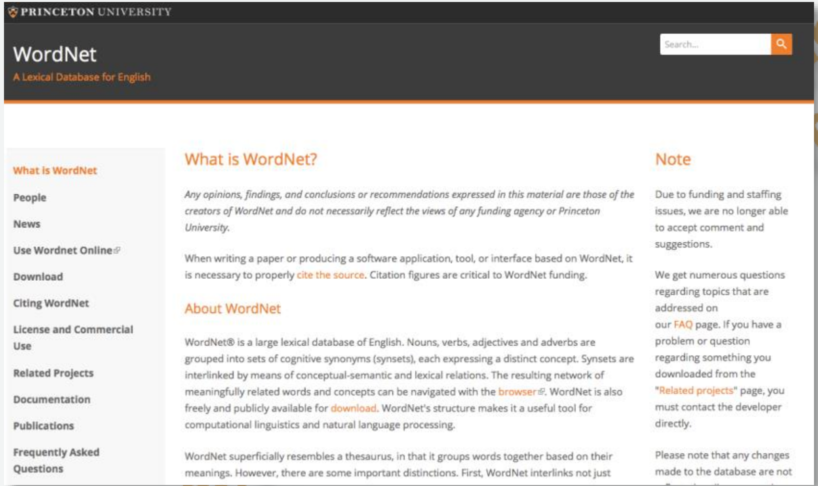
http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar

6786 words

- 2006 positive
- 4783 negative

SentiWordNet

All WordNet synsets automatically annotated for degrees of positivity, negativity, and neutrality/objectiveness



https://github.com/aesuli/sentiwordnet

The current version of SentiWordNet is 3.0, which is based on WordNet 3.0

Example

Let's dive into SentiWordNet dataset and explore sentiments of some words.

Every word has positive and negative meaning scores:

Word	Positive score	Negative score
able	0.125	0
ugly	0	0.625
unbalanced	0.125	0.375

VADER Sentiment Analysis

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis.

Rates sentiments in range [-4, -3, ... ,3, 4], For example -3.7

-4 :Extremely Negative
-3: Very Negative
-2: Moderately Negative
-1: Slightly Negative

0: Neutral

+4 :Extremely Positive
+3: Very Positive
+2: Moderately Positive
+1: Slightly Positive

Sentiment metrics with score [-1, 1]:

-Positive
-Neutral
-Negative
-Compound

VADER Sentiment Analysis

It considers the following cases:

1. Punctuation: Namely the exclamation point **(!)**, increases the magnitude of the intensity

2. Capitalization, specifically using **ALL-CAPS** to emphasize meaning.

3. Degree modifiers (also called intensifiers, booster words, or degree adverbs): “The service is **extremely good**” > “The service is very good” > “The service is marginally good”

4. The contrastive conjunction “but”: “The food here is great, but the service is horrible”

5. Uses tri-gram to extend: “The food here isn’t really all that great”

Word Representations

One-hot-representation:

Assume we have a vocabulary of size 10,000.

V=[a, about, ….., zombie, <UNKN>]

a => [1, 0, 0, …, 0], pos 0

about => [0, 1, ………, 0], pos 1

….. ………

man => [0, 0, ..1,…, 0], pos 5,481

woman => [0, 0, ….1.., 0], pos 8,993

Word Representations

man	woman	king	queen	lemon	apple
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	0	0
..	..	..	..	..	..
..	..	..	..	..	..
..	..	1	0	..	..
1	..	..	1	1	..
..	..	..	..	..	..
..	1	0	..	..	..
..	..	..	..	..	..

One-hot-encoding can't capture semantic relationships.

Word Representations

feature	man	woman	king	queen	lemon	apple
gender	-1	1	-0.93	0.92	0.01	0.03

Word Representations

feature	man	woman	king	queen	lemon	apple
gender	-1	1	-0.93	0.92	0.01	0.03
royal	0.05	0.07	1	1	0.02	0

Word Representations

feature	man	woman	king	queen	lemon	apple
gender	-1	1	-0.93	0.92	0.01	0.03
royal	0.05	0.07	1	1	0.02	0
food	0.01	0.02	0.01	0.01	0.96	0.97

Word Representations

feature	man	woman	king	queen	lemon	apple
gender	-1	1	-0.93	0.92	0.01	0.03
royal	0.05	0.07	1	1	0.02	0
food	0.01	0.02	0.01	0.01	0.96	0.97
fruit	0.01	0.02	0.01	0	0.98	0.99

Word Representations

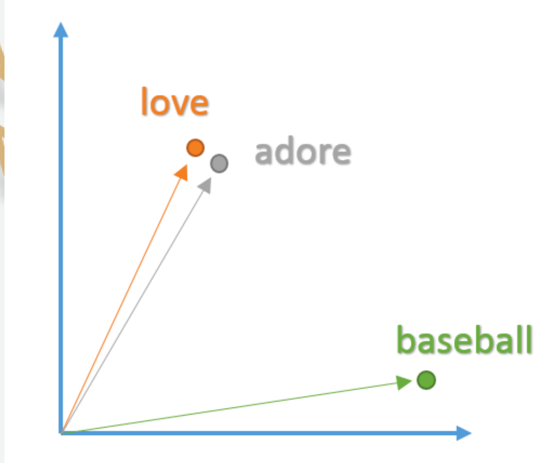
feature	man	woman	king	queen	lemon	apple
gender	-1	1	-0.93	0.92	0.01	0.03
royal	0.05	0.07	1	1	0.02	0
food	0.01	0.02	0.01	0.01	0.96	0.97
fruit	0.01	0.02	0.01	0	0.98	0.99
….						
….						
….						

Word2Vec (2013)

Convert each text or token into a real valued vector.

These vectors carry patterns, information, meaning and semantics.

For example: The two words: "Love" and "Adore" have very similar meaning.

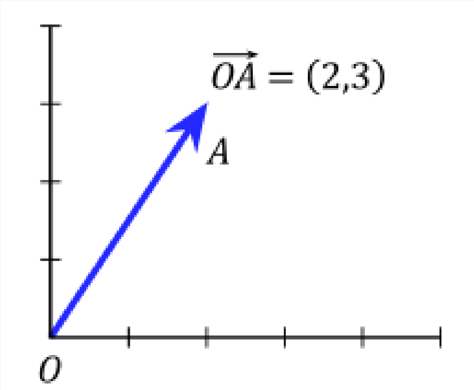


Vectors

- Quantities made of direction and magnitude.
- It can be represented by n dimensions: a = (a_0, a_1, …, a_n)
- Vectors have magnitude and direction.

Magnitude: $\|a\| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$

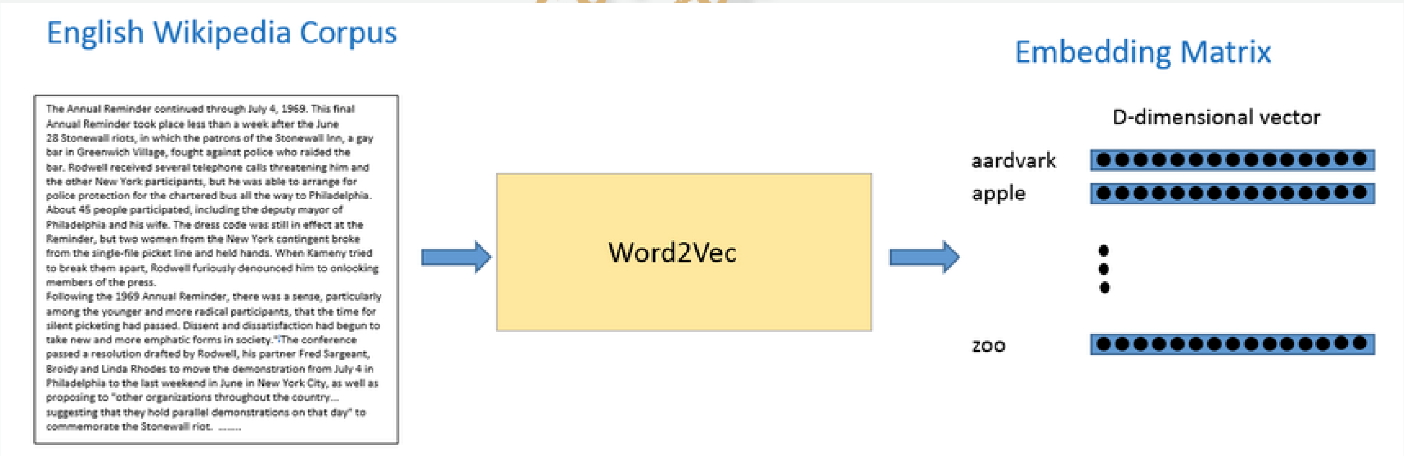
The vector from origin to (2,3)



Word2Vec (2013)

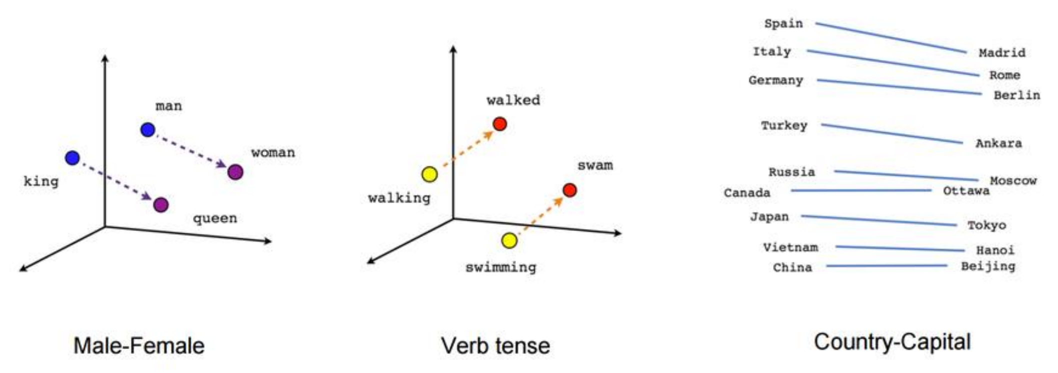
Word vectors are also called “embeddings”.

Word2Vec learns similar words are mentioned in similar contexts and learns to place similar words closer to each other.



Some results from Word2Vec

Locations of words are determined by their meaning.



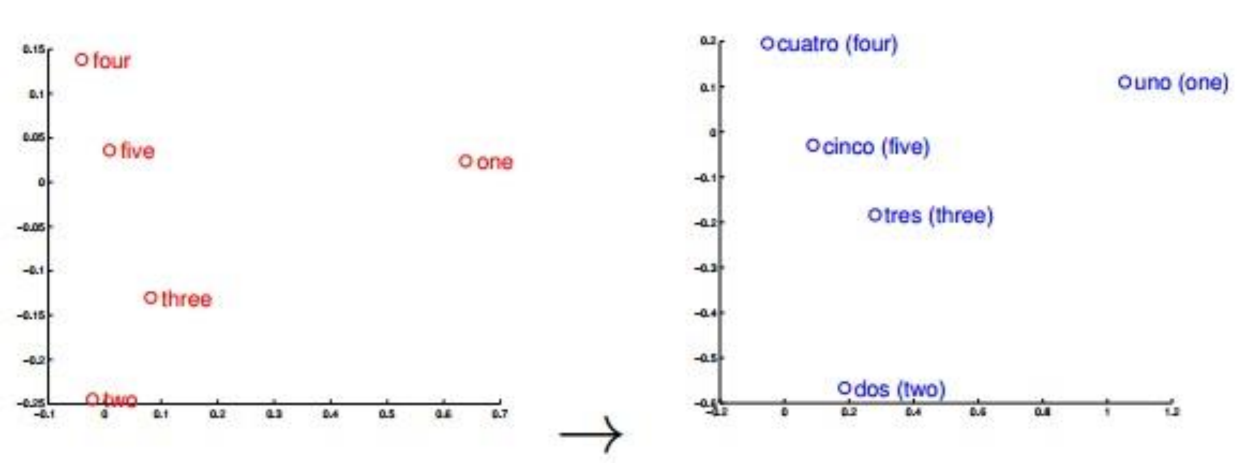
Words with similar meanings will be closer to each other.

- Gaps or distances between word vectors also have meaning.
- If you go to the location of word vector for "king" and move in the same direction and distance between "man" and "woman", you end up near the word "queen". (king – man + woman ≈ queen)

Some results from Word2Vec

Relationships between the words are preserved for different languages.

For example, what is the Spanish word for “one”?



Word2Vec

Paper link here:

https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

Data:

- Trained on Google News dataset (100 billion words).
- It contains 3 million words and phrases. (size: 3 million rows x 300 columns)
- Downloadable here (1.5GB, Official External Google Link)

Sentence Vectors

Main goal: Create a numeric representation for a sentence (document) regardless of its length.

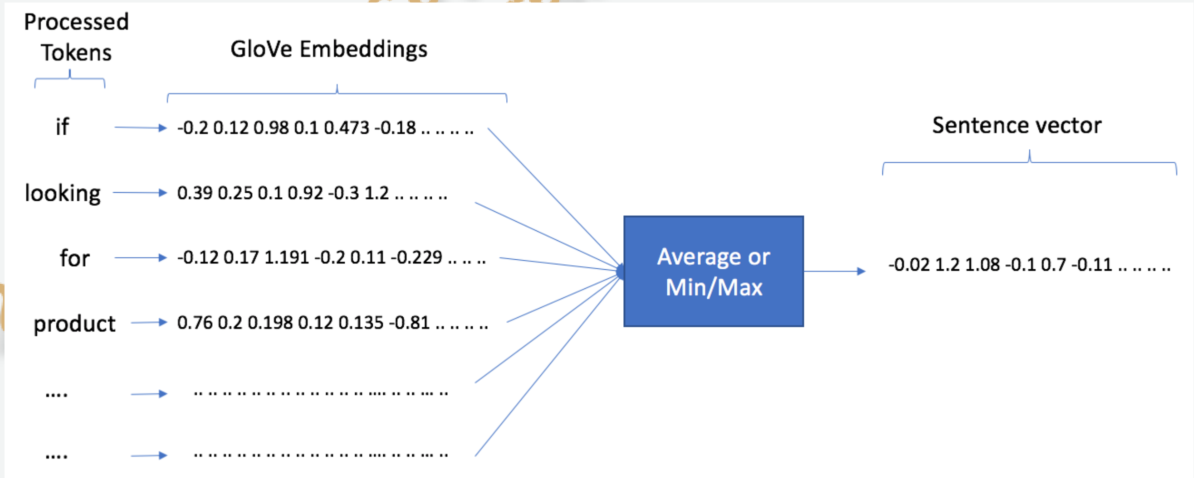
"The school is closed today" => [0.10, 001, -0.1, ……., 0.21] (size n)

"Short sentence" => [0.31, -0.04, 0.29, …….,0.1] (size n)

Method 1 – Average/Sum Word Vectors:

For each sentence:

- Find the corresponding word vector for each word/token.
- Average (or apply min/max) the word vectors for each sentence.



Method 2 – Weighed Sum of Vectors:

The first method assumed that each word/token has the same effect on the sentence vector.

We can apply some weights to each word token. Weights (w1, w2, ..) can be simply Term Frequency, TF-IDFs or document position (title, main text, conclusion).

$$w1 * v1 + w2 * v2 + w3 * v3 + \ldots..$$

Method 3 - Pre-trained System: Universal Sentence Encoder

Google's sentence encoder. Paper is here.

Provides pre-trained models to get fixed size sentence (512) vectors.

Example:

- "I worked today"=> [0.12 0.21 -0.113, 0.15, ….]
- "This is a very long sentence"=> [0.2 -0.19 0.453, 0.2, ….]
- We can use it to encode sentences for text classification problems.

Amazon SageMaker NLP Algorithms

Algorithm	Type	Info	Use Case
BlazingText	Supervised	It is an implementation for Word2vec and text classification algorithms, and it provides the Skip-gram and continuous bag-of-words (CBOW) training architectures	sentiment analysis, named entity recognition, machine translation
Neural Topic Model	Unsupervised	organize a corpus of documents into topics that contain word groupings based on their statistical distribution	classify or summarize documents based on the topics detected or to retrieve information or recommend content based on topic similarities
Latent Dirichlet Allocation	Unsupervised	attempts to describe a set of observations as a mixture of distinct categories (topics are learned as a probability distribution over the words that occur in each document)	Similar to NTM
Sequence-to-Sequence	Supervised	It uses Recurrent Neural Networks (RNNs) and Convolutional Neural Network (CNN) models. Its input is a sequence of tokens (for example, text, audio) and the output generated is another sequence of tokens	machine translation, text summarization, speech-to-text
Object2Vec		It is general-purpose neural embedding algorithm. Object2Vec generalizes the well-known Word2Vec embedding technique for words. It learns embeddings of more general-purpose objects such as **sentences**, customers, and products.	sentiment analysis, document classification, and natural language understanding

Amazon NLP AI Services

Service	Describtion	Features
Comprehend	Amazon Comprehend uses natural language processing (NLP) to extract insights about the content of documents.	Entity recognition, Key phrases extraction, language detection, sentiment analysis, syntax analysis
Translate	Amazon Translate is a text translation service that uses advanced machine learning technologies to provide high-quality translation on demand	translation
Polly	Amazon Polly is a cloud service that converts text into lifelike speech.	generate speech from either plain text or from documents marked up with Speech Synthesis Markup Language (SSML).
Transcribe	Amazon Transcribe uses advanced machine learning technologies to recognize speech in audio files and transcribe them into text.	Speech to text
Lex	Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text.	Chat bots

Thank you!