Natural Language Processing Pt. 1

Terminology: NLU vs. NLP vs. ASR

Module 8: Natural Language Processing

Module Checklist

- ❏ What is Natural Language Processing and its applications?
- ❏ What is text as data?
  - ❏ Bag-of-words, vectors, matrices
- ❏ Supervised learning algorithms
  - ❏ Text classification (Naive Bayes)
- ❏ Next steps for NLP

What is natural language processing?

Natural language vs. artificial language

Natural language is defined as "a language that has developed naturally in use (as contrasted with an artificial language or computer code.)"

The New York Times | https://nyti.ms/2qXzVH2

SCIENCE

The Science Behind the Flamingo's One-Legged Stance

Trilobites

By JOANNA KLEIN   MAY 24, 2017

Squat down as if you're going to sit in a chair. Make sure to keep your back straight, use your hips as a hinge and push your butt backward. Try not to lean forward. Maintain your knees and ankles at 90-degree angles. Now try it on just one leg, and then swap that one with the other. To make it even harder, stand on a foam mat — and close your eyes.

You may feel your body wobbling, or you may fall over. If only you were a flamingo.

Flamingos can stand on one leg for a really long time. They even do it while sleeping. And according to a **study** published Tuesday in Biology Letters, flamingos may not even need to use their muscles for the task.
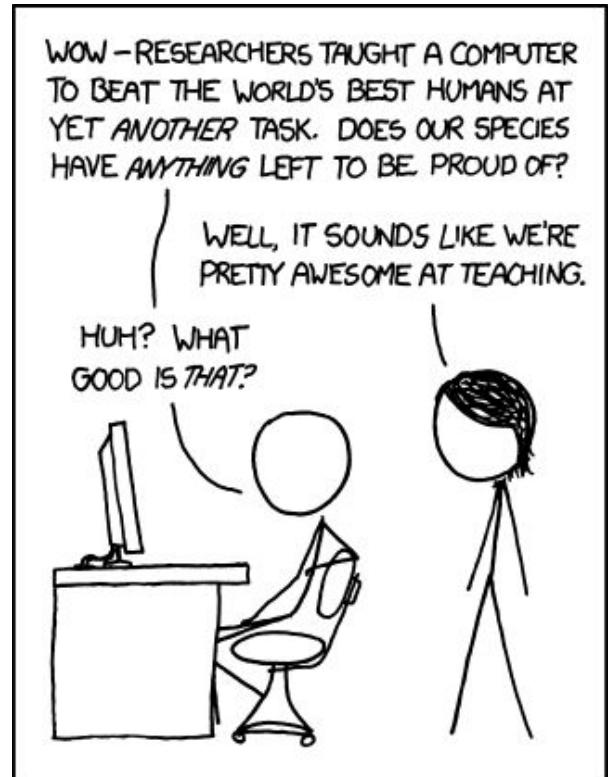
```python
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '    %s [label="%s' % (nodename,label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print '    %s -> (' % nodename,
        for name in children:
            print '%s' % name,
```

In one area, humans still beat computers:

Natural language!

Reading, writing, communicating and otherwise working with *unstructured data*.

Why is NLP so difficult?

All of our examples in the course have thus far involved modeling **structured data.**
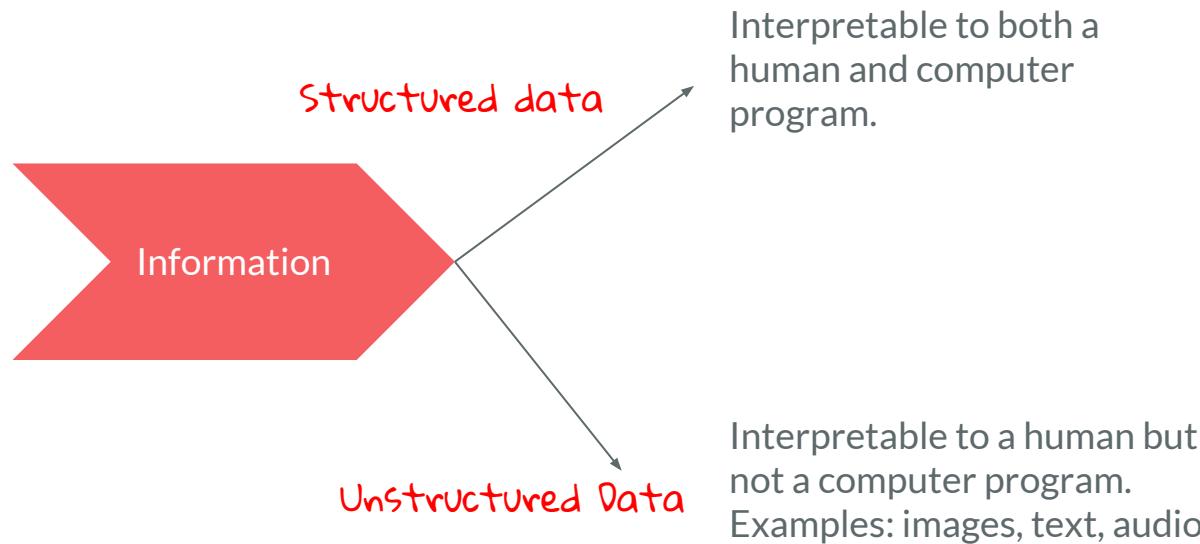
Outstanding Debt	Current Annual Income ($)	Approval for credit card	Predicted Approval for credit card
		Y	Y*
200	12,000	No	Yes
60000	60,000	No	No
0	11,000	Yes	No
10000	200,000	Yes	Yes

Structured data (data frame)

NLP is difficult because text is an example of unstructured data.

Our algorithms know how to pull information from structured data but are unable to interpret unstructured data. **How do we pull information from unstructured data?**

Structured data

Information

Interpretable to both a human and computer program.

Unstructured Data

Interpretable to a human but not a computer program.
Examples: images, text, audio

Focus in NLP is on how to better represent language as structured data!

Represent the data to retain as much of the information as possible. This turns out to be a non-trivial task

Unstructured Data

Interpretable to a human but not a computer program.

data representation

Structured data

Interpretable to both a human and computer program.

music   images   text

Note that you can still pull labels from unstructured data!

Books

Labels:

1. Author
2. Year published
3. Genre
4. Number of pages
5. Title

How do we extract labels from the unstructured data?

# Why else is text difficult to represent?

Let's walk through an example.

Human Intuition

NLP

Sam is trying to build his relationship with Jasmine.

%%$$$asds tasjkshdf adbha sfsfvdf

Unlike all previous models we have looked at, computers <u>cannot understand raw text at all!</u> We first need to represent raw text in a way they can use.

Language difficult to represent to a machine because:

Is your name Wi-fi?
Because I'm really feeling a connection

Language is ambiguous:
- Subtleties
- Cultural references
- Jokes, sarcasm, wordplay
- Phonetic similarities between words

There are certain grammatical rules that we can take advantage of, but this only gets us half the way.

Let's try a short quiz that will demonstrate linguistic ambiguity:

The trophy doesn't fit into the brown suitcase because **it's too small**. What is too small?

The suitcase or the trophy?

The trophy doesn't fit into the brown suitcase because **it's too big**. What is too big?

The suitcase or the trophy?

You can easily tell what "it" is, **even if the sentences are totally different,** using your own understanding of trophies and suitcases. A computer won't have such an easy time.

If NLP is so difficult, why do we bother?
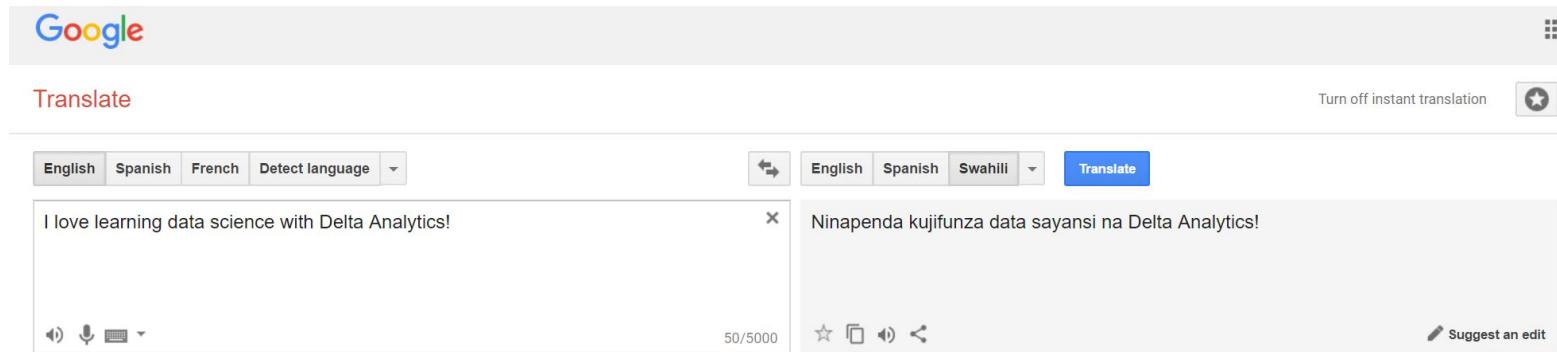
Because NLP is important!

- Most of the data in the world is unrepresented.
- Vast amount of data are stored as text (think about websites, laws, emails, books) and this amount will increase in the future
- Allowing machines to understand text unlocks vast amounts of information. *NLP is incredibly powerful!*

Machine translation & recommendations

1. *Google translate is driven by NLP and deep learning*

2. *New York Times recommends new articles based upon keyword topic extraction*

Source: New York Times - LDA recommendation, Google translate, the great AI awakening

Familiar applications of NLP

Text classification: Adding tags to academic papers

Spell check, autocomplete and Search results

When you type a query like '*natural language processing*'
into Google search it is already doing 3 NLP tasks:

1. spell and grammar check
2. autocompletion of search results
3. Information retrieval

These are all NLP tasks!

Artificial intelligence personal assistants
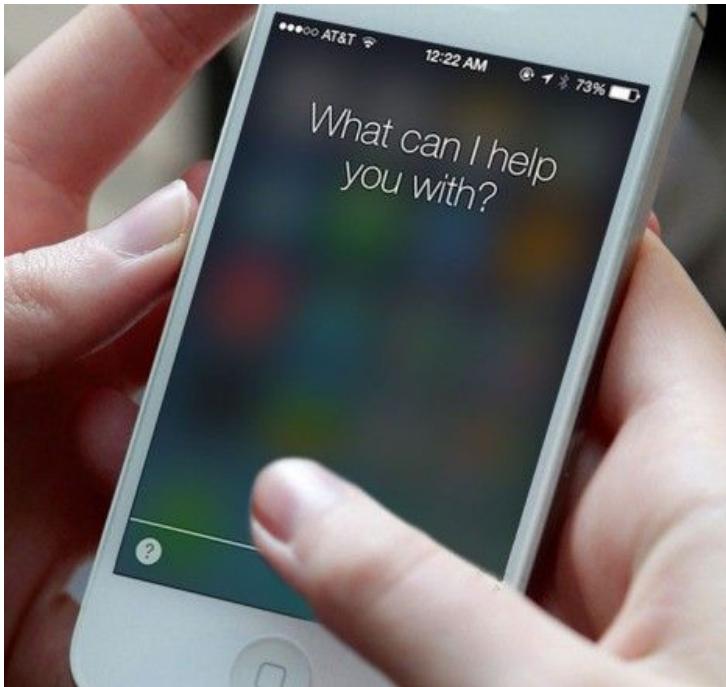
x.ai is a personal assistant who schedules meetings for you

Sign up for your free trial

X.ai relies heavily on NLP to schedule meetings:

1. X.ai extracts information from your emails in order to schedule:
   a. Day of week
   b. Location
   c. Number of people to schedule
2. *If you reply to your non-human assistant, they reply back!* How? X.ai extracts keywords from your initial email and uses the keywords to classify the text as having a certain intent (text classification).

NLP is often an intermediate step in complex processes
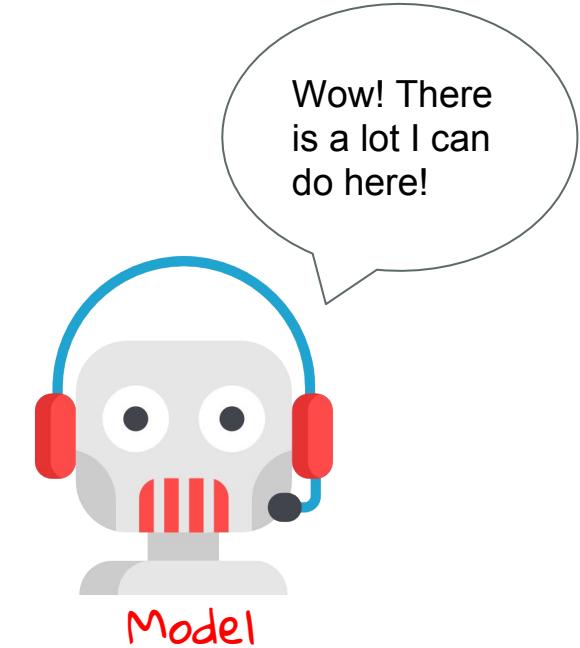
Apple's Siri converts the sound of your voice to words to action.

When converting between words to action, Siri needs to "understand" what you're asking her to do. **This requires NLP!**

NLP applications are very varied! All involve first representing text as data, and then extracting meaning from the text.

NLP applications:

- ✓ Classifiers - spam filter
- ✓ Information retrieval - Google Search
- ✓ Information extraction - AI assistants
- ✓ Machine translation - Google translate
- ✓ Question & answering - automatic response to certain emails
- ✓ Summarization of text - what is this article about?
- ✓ Sentiment analysis - is this document positive or negative?
- ✓ Speech processing - options on phone helplines
- ✓ Optical character recognition - scan cheques at ATMs
- ✓ Spell check, autocomplete - Google Search

Wow! There is a lot I can do here!

Model

How do we convert text into data?

Representing text as data is a big part of our NLP work!

| Gather text | Represent text as data | Modeling Phase | Validation Phase |

What makes NLP different from the other algorithms we have studied so far is how difficult and computationally intensive this step can be.

Represent text as data

Data cleaning

stemming

lemmatization

Stop words

Additional cleaning

Data representation

tokenizing

n-grams

bag-of-words

Not necessarily in this order - depends on your research question

Firstly, we clean the input text and then we represent it using a model.

Let's go through each of these techniques step by step!

Reminder example: Sam sends a text to Jasmine.

First, some vocabulary!

feature   observation   dataset
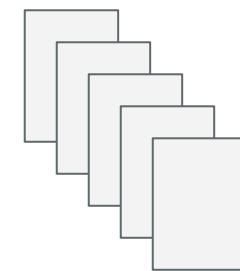
Word

Document
Many words

Corpus
Many documents

In our text exchange between Sam and Jasmine:
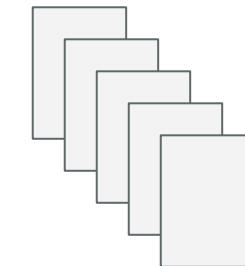
feature   observation   dataset

Word   **Text**   **All texts**
                  Document   *Corpus*

Let's start with *text cleaning:*

The goal of cleaning text data is to standardize the text as much as possible, so that words with the same intent are grouped together.

Represent text as data → Data cleaning → stemming

lemmatization

stop words

Lower case, remove special characters, remove spaces

Represent text as data → Data cleaning → Lemmatization

Lemmatization is similar to stemming, but uses morphological analysis of words to return them to their root.

Original	Stemmed	Lemmatized
walked	walk	walk
are	are	be
natural	natur	natural
processing	process	process

Removing "stop words" removes common words that don't add meaning

Original sentence:
"Hi Jasmine, would you like to go running in the park this weekend? We could catch a movie and dinner afterwards! "

Sentence with stopwords removed:
"Hi Jasmine, would you like ~~to~~ go running ~~in~~ ~~the~~ park ~~this~~ weekend? We could catch ~~a~~ movie ~~and~~ dinner afterwards! "

Two main ways to approach removing stop words!.

predetermined list of words

Using a list of predetermined stopwords (available through nltk package).

adapted from corpus

Discarding words that appear too often in your data (using frequency counts)

Lowercase, remove spaces, remove any special characters (%,?,!)

Lowercase letters

remove special characters

Original sentence:

"Hi Jasmine, would you like to go running in the park this weekend? We could catch a movie and dinner afterwards! "

Cleaned sentence:

"hi jasmine would you like to go running in the park this weekend we could catch a movie and dinner afterwards"

Once text is cleaned, the next step is *tokenization*.

| Data representation | tokenizing | unigrams | bag-of-words |

| bigrams |

| n-grams |

Now we have clean text, we can represent our text as data using a process called tokenization.

Let's take a look!

Breaking up the text into words, phrases, elements (aka "tokens")

Original sentence:
"Hi Jasmine, would you like to go running in the park this weekend? We could catch a movie and dinner afterwards! "
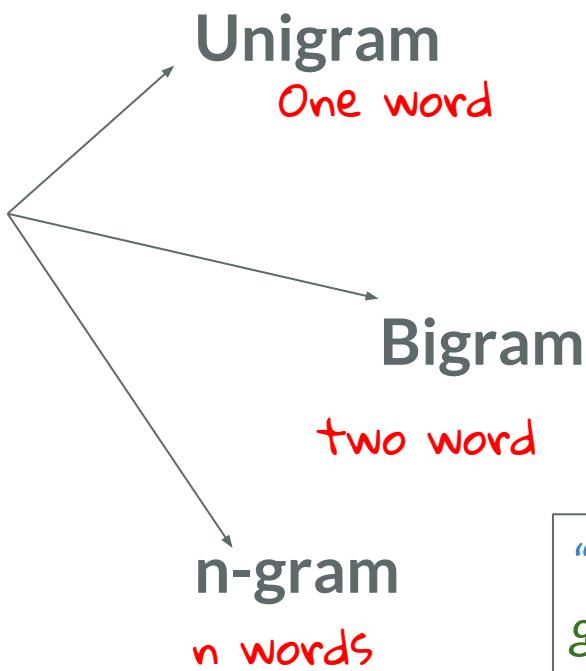
Tokenized sentence:
"Hi", "Jasmine", "would", "you", "like", "to", "go", "running", "in", "the", "park", "this", "weekend", "We", "could", "catch", "a", "movie" "and", "dinner," "afterwards"

Is there a **best** type of tokenization? As with many of our choices in ML, there is a tradeoff!

- Unigram models ignore context of a word, so <u>bias is high.</u>
- Bigram, trigram models take into account context, but need more data to train.

Bag-of-words algorithms use unigram tokenization, so their key disadvantage is that they do not take any word's **context** into account.

Think about it: with bag-of-words, a document that reads "it's bad, not good" would be considered the same as a document that reads "it's good, not bad".

Let's take a closer look at the Bag-of-Words representation.

Bag of words uses unigram tokenization across the corpus of text messages!

Text 1:
"Hi Jasmine, would you like to go running in the park this weekend? We could catch a movie and dinner afterwards! "

Text 2:
"Great dinner Jasmine! Sweet dreams."

Text 1:
"Hi", "Jasmine", "would", "you", "like", "to", "go", "running", "in", "the", "park", "this", "weekend", "We", "could", "go", "to", "a", "movie" "and", "dinner," "afterwards"

Text 2:
"Great", "dinner", "Jasmine", "Sweet", "dreams"

document	corpus
Single text	→ Multiple texts

The first step in bag-of-word representation is Unigram tokenization of each document. Then these words are aggregated using different approaches.
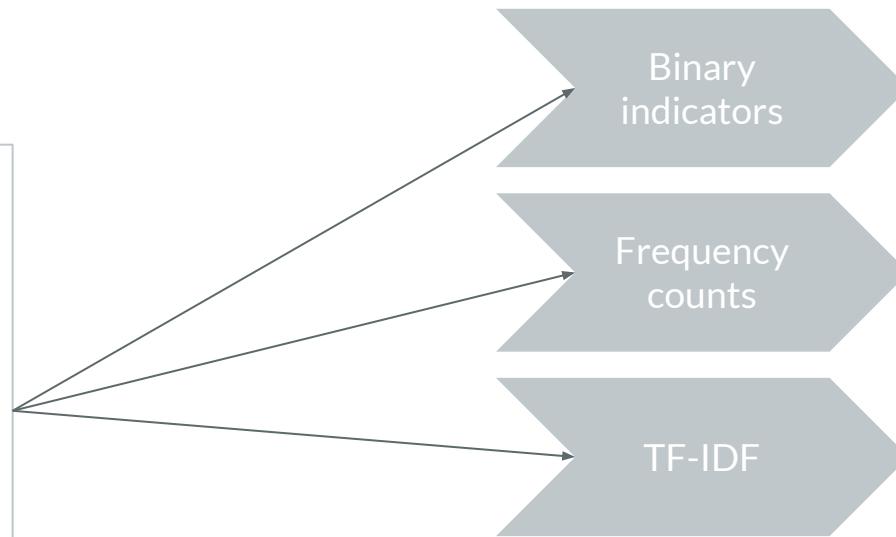
1) Unigram tokenization of each document

Text 1:
"Hi", "Jasmine", "would", "you", "like", "to", "go", "running", "in", "the", "park", "this", "weekend", "We", "could", "go", "to", "a", "movie" "and", "dinner," "afterwards"

Text 2:
"Great", "dinner", "Jasmine", "Sweet", "dreams"

Binary indicators

Frequency counts

TF-IDF

TF-IDF, Binary and frequency counts are very simple aggregation approaches.

bag-of-words → Binary counts

Binary indicators are 1 or 0 dependent on whether the word is present in the document.

Text 1:
"Hi", "Jasmine", "would", "you", "like", "to", "go", "running", "in", "the", "park", "this", "weekend", "We", "could", "go", "to", "a", "movie" "and", "dinner," "afterwards"

Text 2:
"Great", "dinner", "Jasmine", "Sweet", "dreams"

text	Jasmine	dinner	dreams	weekend	...
1	1	1	0	1	...
2	1	1	1	0	...

Bag-of-words → **Frequency counts**
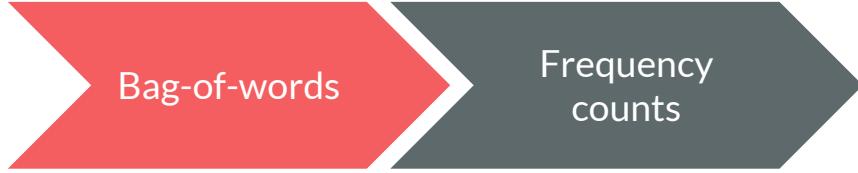
Frequency counts are the number of times each word appears in the document

Text 1:
"Hi", "Jasmine", "would", "you", "like", "to", "go", "running", "in", "the", "park", "this", "weekend", "We", "could", "go", "to", "a", "movie" "and", "dinner," "afterwards"

Text 3:
"Hi", "Jasmine", "how", "is", "your", "week", "going" "I" "am" "going" "bowling" "this" "saturday" "want" "to" "come"

text	Jasmine	dinner	go	going	...
1	1	1	2	0	...
3	1	0	0	2	...

Problem: How important is each word?

| Bag-of-words | → | Frequency counts |

Problem: How important is each word?

Text 1:
"Hi", "Jasmine", "would", "you", "like", "to", "go", "running", "in", "the", "park", "this", "weekend", "We", "could", "go", "to", "a", "movie" "and", "dinner," "afterwards"
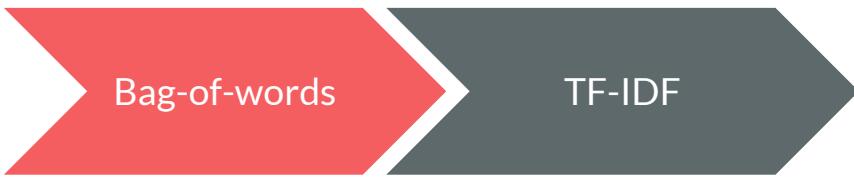
Text 3:
"Hi", "Jasmine", "how", "is", "your", "week", "going" "I" "am" "going" "bowling" "this" "saturday" "want" "to" "come"

text	Jasmine	dinner	go	going	...
1	1	1	2	0	...
3	1	0	0	2	...

Frequency tells us how important a word is to a document. ***But how can we judge how important is that word in the corpus?*** E.g. "Jasmine" probably has a high frequency, but this doesn't give us that much new information - the entire corpus is about her! ***Tf-idf helps determine importance relative to the corpus.***

TF-IDF: Term frequency, inverse document frequency
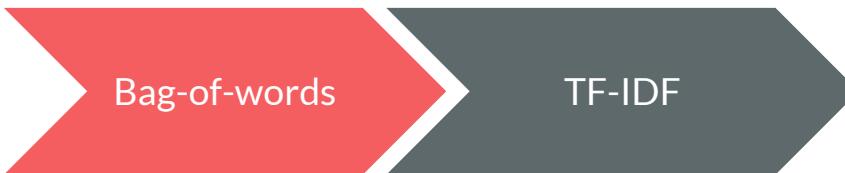
Tf-idf is a number assigned to each word, intended to reflect **how important that word is to a document**.

Tf-idf **increases** proportionally to the number of times a word appears in the document, but is **offset** by the frequency of the word in the corpus.

3 Step Procedure:

1) Compute term frequency

2) Compute inverse document frequency

3) Multiply 1)*2)

TF-IDF vs. frequency

TF-IDF can tell us which words provide the most information about a given document. Let's compare frequency and tf-idf:

Frequency

	Jasmine	Sam	running
Text 1	1	2	0
Text 2	1	1	0
Text 3	2	1	1

Tf-idf

	Jasmine	Sam	running
Text 1	0.2	0.1	0
Text 2	0.01	0.02	0
Text 3	0.1	0.1	0.8

Based on frequencies alone, we see that the words "Jasmine" and "Sam" are important. But we knew that! The tf-idf table provides a little more insight: the word "running" is **relatively more important**. If we relied only on frequencies, we might have missed the relative importance of the word "running"! Let's get into how to calculate tf-idf next.

Computed for every word in a document. Equal to frequency count normalized to account for length of document.

TF: Term Frequency: measures how frequently a term occurs in a document.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

Term frequency table:

	running	Total words in text	TF(T)
Text 1	1	22	0.04

Text 1:
"Hi", "Jasmine", "would", "you", "like", "to", "go", "running", "in", "the", "park", "this", "weekend", "We", "could", "go", "to", "a", "movie" "and", "dinner," "afterwards"

The term frequency of the word running in text 1 is 0.04 (1/22).

Computed <u>across</u> every single document in the corpus

Inverse Document Frequency measures how important a term is.

While computing TF, all terms are considered equally important.
However, certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance.

IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

TF	IDF
All terms considered equally important	Weighs down frequent term while scales up infrequent terms.
'Is' as important as 'girlfriend'	"Vacation" will be more important than "and"

$IDF(t) = \log_e(\text{Total number of documents / Number of documents with term } t \text{ in it})$

Text 1:
"Hi", "Jasmine", "would", "you", "like", "to", "go", "running", "in", "the", "park", "this", "weekend", "We", "could", "go", "to", "a", "movie" "and", "dinner," "afterwards"
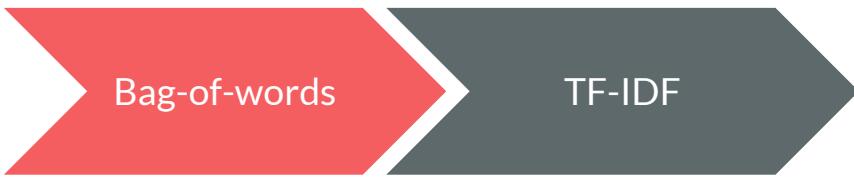
Text 2:
"Great", "dinner", "Jasmine", "Sweet", "dreams"

Total number of documents	running	IDF(T)
2	1	0.69

$IDF(\text{running}) = \log_e(2/1)$

Let's bring it all together!

3 step procedure:

1) Compute term frequency

2) Compute inverse document frequency

3) Multiply 1)*2)

	word	TF(T)	IDF(T)	TF-IDF(T)
Text 1	running	0.04	0.69	0.0276

TF-IDF tells us which words are important and which are not, **relative to the corpus**!

Thus, composition of the corpus can also be an important decision to make as a researcher.

Binary, frequency and TF-IDF all result in sparse matrices.
A sparse matrix is a data frame where most elements=0.

The matrix is "**sparse**" because each feature is a word in the corpus. For a big corpus, this will be a large vocabulary. ***Most elements in the matrix will be 0.***

text	Jasmine	dinner	dreams	weekend
1	1	1	0	1
2	1	1	1	0
3	1	0	0	0

In our example, our corpus is the set of text messages.

Text 3:
"Hey", "Jasmine", "how", "is", "your", "week", "going"

So, where are we?

| Gather text | Represent text as data | Exploratory Analysis | Modeling Phase | Validation Phase |

We put a lot of effort into cleaning our data and explored one simple model (bag-of-words) to represent our text.

Now, we can finally start doing some exploratory analysis.

Exploratory Analysis

with Python package "nltk"

Exploratory Analysis	frequency	Count of times each word appears in the corpus
	conditional frequencies	Count of times each word appears in the corpus dependent on an additional criteria
	concordance	A list showing all the appearances of a text of phrase
	similar	Returns words in a range of contexts similar to that of the input word
	collocations	Sequences of words that usually appear together
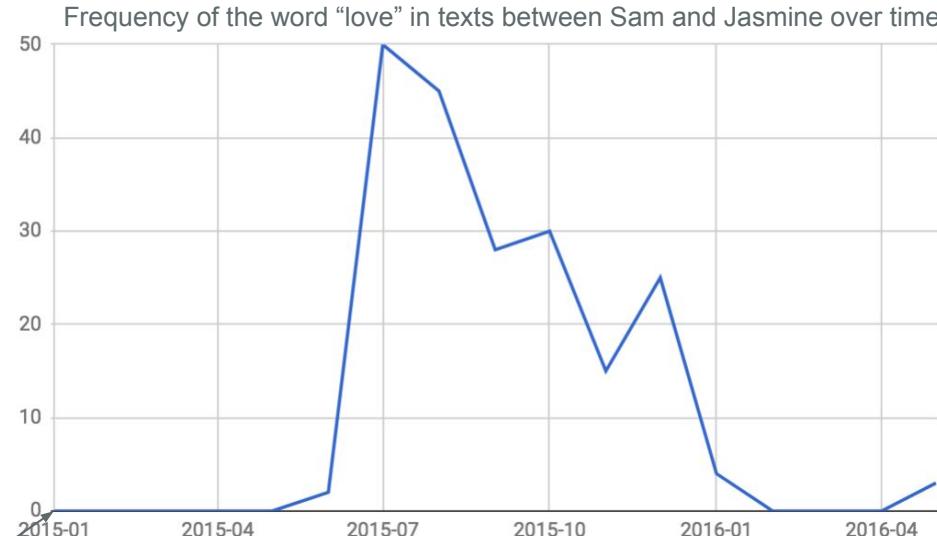
Exploratory Analysis → frequency

Frequency is the count of times each word appears in the corpus

Frequency of the word "love" in texts between Sam and Jasmine over time

Relationship start date

nltk.ConditionalFreqDist()

Count of times each word appears in the corpus subject to additional criteria

For example, we may want to look at the frequency of words by the type of news article. One article is categorized "news" and the other "romance."

Condition: News	
the	ⅢⅢ ⅢⅢ ⅢⅢ ⅢⅢ
cute	
Monday	ⅢⅢ ⅢⅢ
could	I
will	ⅢⅢ Ⅲ

Condition: Romance	
the	ⅢⅢ ⅢⅢ ⅢⅢ
cute	Ⅲ
Monday	I
could	ⅢⅢ ⅢⅢ ⅢⅢ
will	ⅢⅢ

Source: Natural Language Processing with Python

nltk.ConditionalFreqDist()

Exploratory Analysis → conditional frequency

You have already seen an example of conditional frequency in our KIVA data!
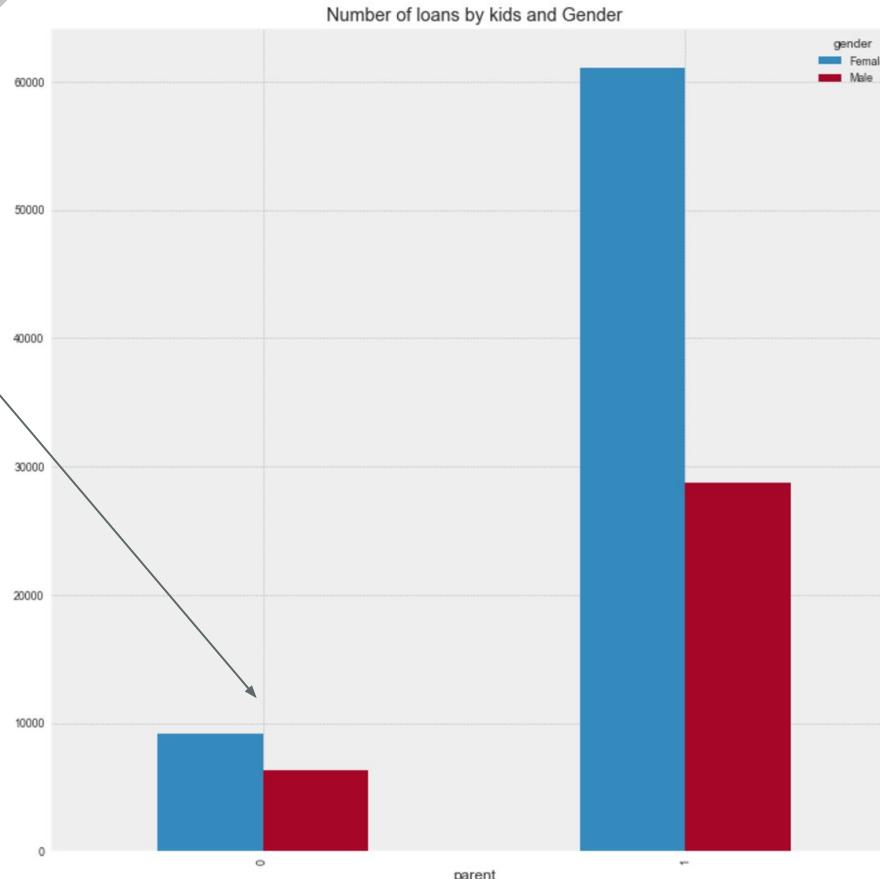
This is an example conditional frequency of a word match by gender.

"Kids" field defined as 1 for any description that had the words "children", "kids", "child", "daughter", "son", "mother", "father".

Number of loans by kids and Gender

A list showing the context surrounding an input word or phrase.

All appearances of the word "woman" in the Kiva data.

```
In [74]:  text_corpus.concordance('woman')

          Displaying 25 of 237 matches:

                                           woman with three kids all of whom attend s
          filling life dorine is a 27-year-old woman with 4 children ranging in age from
          aid successfully habiba is a married woman with four children all of whom atten
          ocation in mombasa asha is a married woman with three children all of whom atte
          the loan promptly nzije is a married woman who has been blessed with five schoo
          thful 28-year-old mrembo ( beautiful woman ) she is a wise farmer whom neighbor
          " she said millicent mobilized other woman in her village and joined juhudi kil
          ease her income level millicent is a woman who is making a change for herself a
          le within 5 years grace is a married woman and owns a house that has piped wate
          cess loans from banks since she is a woman and a smallholder farmer she joined
          ical decisions margaret is a married woman with two children both of whom atten
          e solar lights mwanasha is a married woman with three children all of whom atte
          n shillings ) to buy poultry being a woman accessing funds is very challenging
          ends her dreams to be an independent woman and to have job security can now com
          microfinance bank rose is a married woman with one child who attends school sh
          live happily mwanamgeni is a married woman with four kids all of whom attend sc
```

nltk.concordance()

Collocations are sequences of words that usually appear together

```
In [76]:  text_corpus.collocations()
```

years old; acre fund; one acre; juhudi kilimo; piped water; join yehu;
greatest monthly; school fees; major challenge; primary customers;
married woman; access loans; kadet ltd; first loan; attend school;
solar lights; wheat flour; microfinance bank; monthly expense; three
children

Collocations takes advantage of the fact that there are pairs of words that naturally go together.

This is often particular to the data set, for example Acre Fund is the name of a partner and would unlikely be a collocation in a more broad corpus.

nltk.collocations()

"Similar" in NLTK takes words that appear around our input word as the context and finds other words that have similar context.

The result is a list of similar words to our input word!
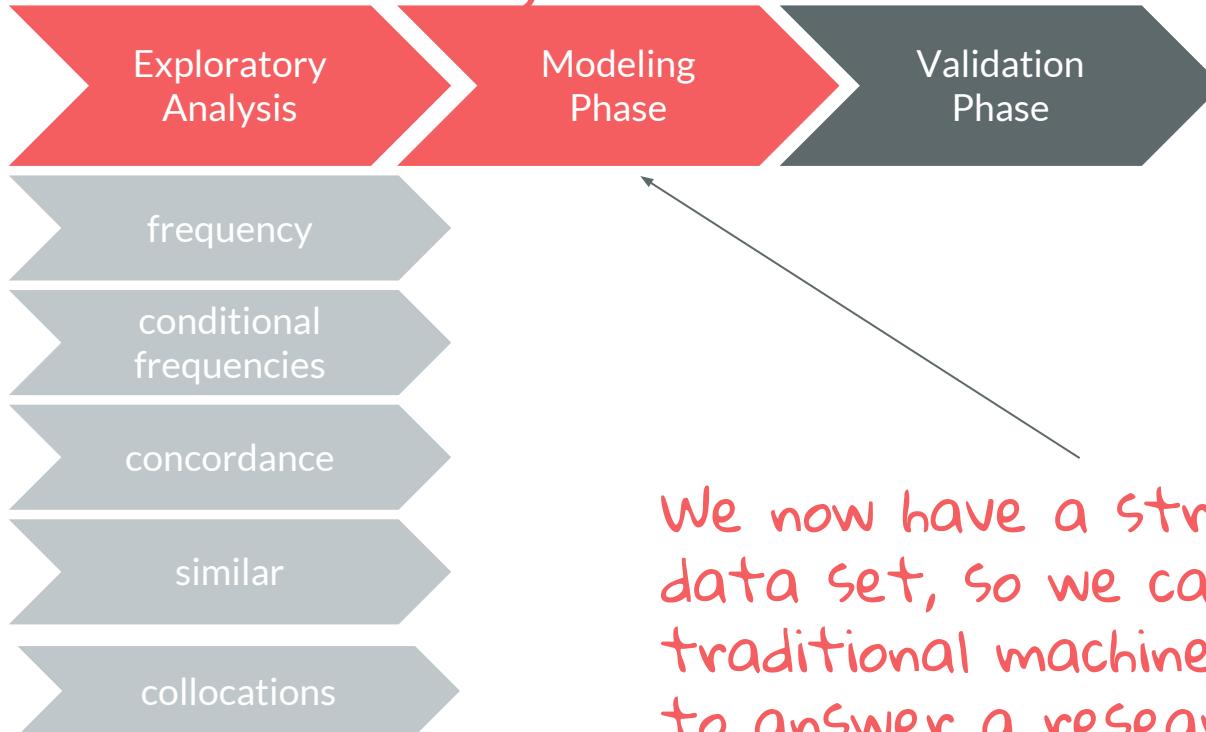
```
In [35]:  text_corpus.similar("mother")

          woman father business loan farmer man house lady and challenge married
          that children friend profit lack living hope life sale
```

```
In [38]:  text_corpus.similar("father")

          mother woman challenge business loan lack farmer way house women
          variety hope life sale stock number lives lady part group
```

Source: Sample of Kiva data

nltk.similar()
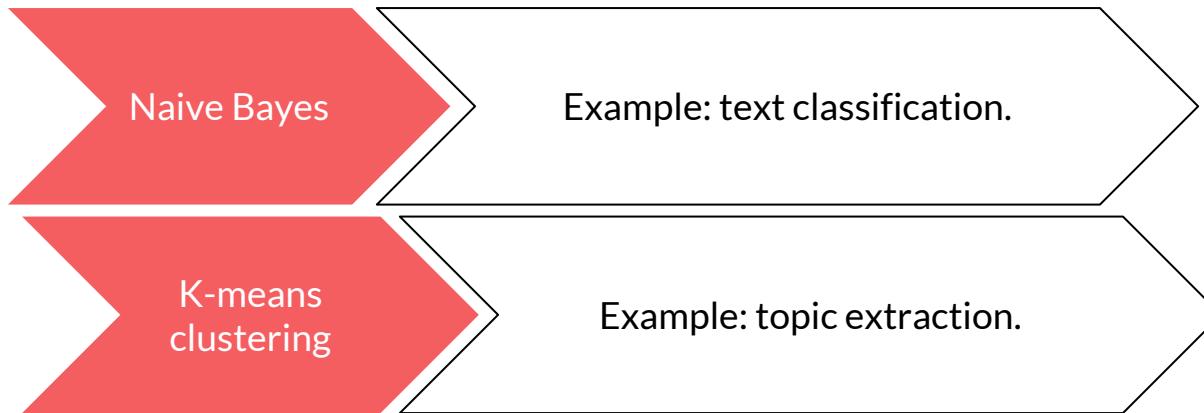
Now we have an idea of what is in the data, let's move into our modeling phase.

| Exploratory Analysis | Modeling Phase | Validation Phase |

frequency

conditional frequencies

concordance

similar

collocations

We now have a structured data set, so we can use traditional machine learning to answer a research question.
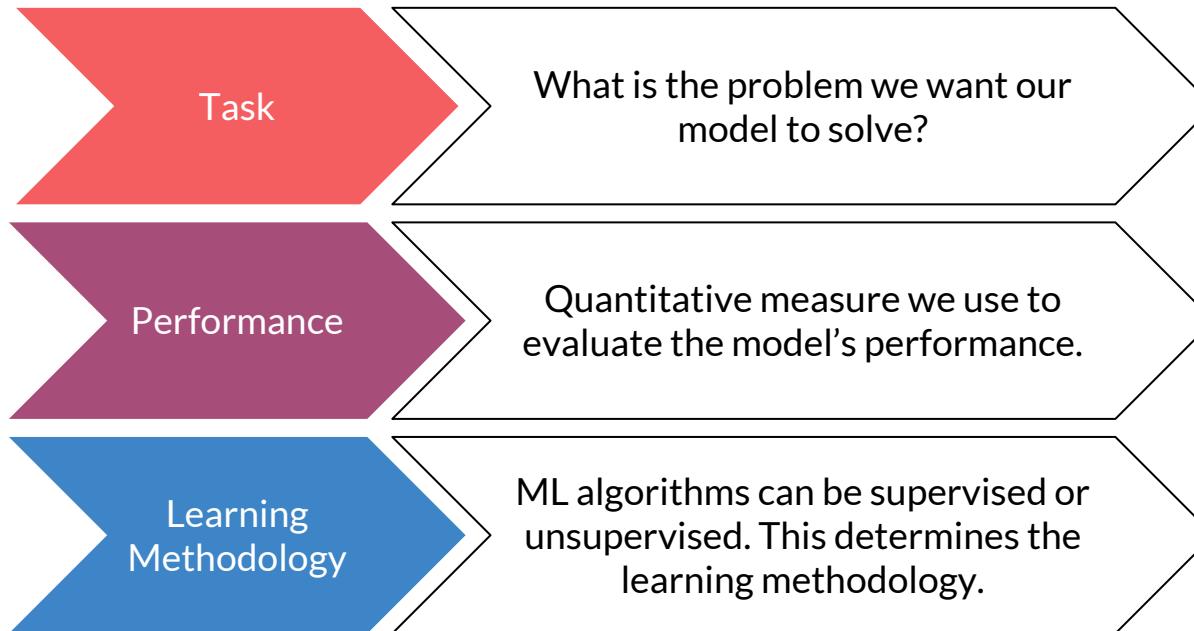
NLP Modeling

We will discuss two algorithms:

| Naive Bayes | Example: text classification. |
| K-means clustering | Example: topic extraction. |

This is just a starting point for your exploration of NLP. Both of these algorithms are fairly intuitive and for some use cases produce powerful results.

We will use our familiar framework to discuss both naive bayes and k-means clustering:

Task	What is the problem we want our model to solve?
Performance	Quantitative measure we use to evaluate the model's performance.
Learning Methodology	ML algorithms can be supervised or unsupervised. This determines the learning methodology.

Remember the difference between supervised and unsupervised?

Supervised Algorithms
- Every observation has an associated label (for every x, there is a true Y).
- Goal is to predict Y using x.
- Since we have the true Y we can tell how far Y* is from the truth.
- **Majority of machine learning is supervised.**

Unsupervised Algorithms
- For every x, there is no Y.
- We do not know the correct answers.
- Instead, we try to model our understanding of the distribution of x to draw inference about Y.

Source: Machine Learning Mastery - Supervised and Unsupervised Learning
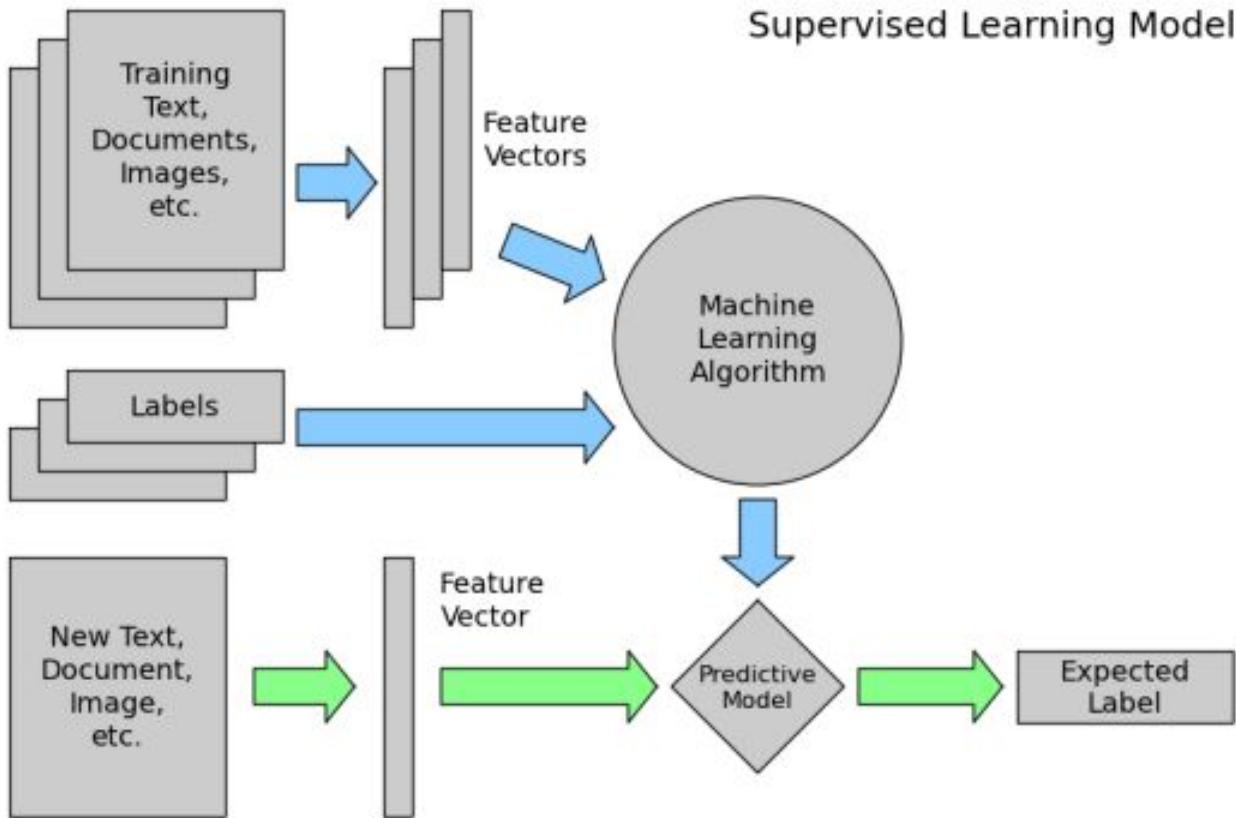
1) Naive Bayes:
   Text classification algorithm

NLP supervised learning algorithms

Supervised Learning Model

Training Text, Documents, Images, etc.

Feature Vectors

Labels

Machine Learning Algorithm

New Text, Document, Image, etc.

Feature Vector

Predictive Model

Expected Label

Naive Bayes: model cheat sheet

Pros

- Simple yet well-performing, especially in document classification.
- Allow every feature or word in a document to contribute to its classification
- Low false positive rate

Cons

- "Naive" - makes assumptions that do not necessarily hold in real life
- Assumes independence between features.

Can we tell whether an email is spam or not? (this is a real email from an inbox!)

FBI INVESTIGATION ON YOUR FUND IN AFRICA - Federal Bureau Of Investigations Headquarters Washington Dc. Building 935 Pennsylvania Ave. NW WAS

Federal Bureau Of Investigations
Headquarters Washington Dc.
Building 935 Pennsylvania Ave.
NW WASHINGTON, D.C. 20535-0001
E-Mail: fbi_govt2@usa.com

NOTICE OF ONGOING INVESTIGATION

Attn Recipient:
This is Agent Dean Marugor, we were sent by the acting Director of Federal Bureau of Investigation (ANDREW G. McCABE), we are currently in Africa as an FBI/ United States delegate that have been delegated to investigate these fraudsters who are in the business of swindling Foreigners that has transactions in Africa.

Be informed that during our investigations we found out that there is a total amount of $4.5 Million that has been assigned in your name as the beneficiary and these fraudsters are busy swindling you without any hope of receiving your fund, these are the works of the fraudsters who needed to extort money from you in the name of this transfer, We have to inform you that we have arrested some men in respect of this delayed overdue fund,We have a very limited time to stay in Africa here so I advise you urgently respond to this message.

These criminals will be caught unaware and we don't want them to know this new development to avoid jeopardizing our investigation, you need to conceal anything that has to do with this exercise to enable us get all the necessary information we required.

I will be expecting your swift response as soon as you receive this email and notify us of any message or phone call you receive from those fraudsters for us to investigate on it before you make any contact with them.

In case if found this message in spam folder, it could be due to your Internet Service Provider, ISP. So kindly move to your inbox before replying.

Regards,
Agent Dean Marugor

Email:fbi_govt2@usa.com
Email:deanmarugor2@gmail.com

Federal Bureau of Investigation
+1(205)340-5968

What is the difference between how a human and machine learning model would decide whether this is spam or not?

Email text:

"Be informed that during our investigations we found out that there is a total amount of $4.5 Million that has been assigned in your name as the beneficiary and these fraudsters are busy swindling you without any hope of receiving your fund."

Human Intuition   Naive Bayes

- I don't have a fund in Africa.
- Would the FBI email me?
- How would I not know about $4.5 million dollars?

Each word in the email contributes to the probability the email is spam, based upon historical frequency in spam messages.

- **Same goal! Both humans & ML are assessing the probability the email is true.**
- **Key difference: Naive Bayes use frequency of words present in the email in historical examples of spam to assign a probability that it is spam.**
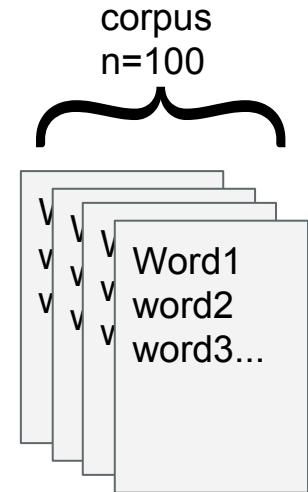
We train our model by assessing how each word of the email **contributes** to a classification of Spam or Not Spam, over our entire labelled sample of emails.

Compute probability of spam **for each word in an email**

Multiply probabilities across all words.

If the total probability is >0.5 **then we classify as spam.**

corpus
n=100

Word1
word2
word3...

Repeat for each email in the corpus of labelled data

Predicting whether an email is spam is an example of a classification task.

Email	"millions"	"FBI"	spam
1	1	1	1
2	0	0	0
3	1	0	0
4	0	0	1
5	1	0	1
6	1	1	1

Classification

IMPORTANT: Notice that our data frame is a bag-of-words count. We did unigram tokenization to transform unstructured data to structured data.

What we want is the probability that an email is spam, GIVEN the words that are present in the email.

This is **conditional probability.**

EMAIL TEXT: "We found out that there is a total amount of $4.5 Million that has been assigned in your name as the beneficiary and these fraudsters are busy swindling you without any hope of receiving your fund..."

Formalized as an equation, our conditional probability for the sample email above is as follows:

P(Spam | We found out that there is a total amount of $4.5 million...)

Conditional probability is a concept that can be seen in our daily lives. For example:

P(Rain|April)

Probably varies quite a bit, depending on where in the world you are!

This equation formalizes the probability that it will rain today, given that it is April.

For a primer on this concept, see:
https://www.khanacademy.org/math/statistics-probability/probability-library/conditional-probability-independence/v/calculating-conditional-probability

Bayes Theorem allows us to calculate conditional probability:

$$P(Rain|April) = \frac{P(April|Rain) * P(Rain)}{P(April)}$$

For a primer on this concept, see:
https://www.khanacademy.org/math/statistics-probability/probability-library/conditional-probability-independence/v/calculating-conditional-probability

How can we use words to predict spam?

The Naive Bayes algorithm will calculate the **conditional probability that an email is spam,** conditioned on the text of the email.

Using our bag-of-words representation, we evaluate the probability of each word separately and sum them up.

P(Spam | We found out that there is a total amount of $4.5 million…)

= P(Spam|we) * P(Spam|found) * P(Spam|total) * P(Spam|amount) * P(Spam|million) …

NOTE: Why did we leave out words like "that", "is", "a", and "of"? These are examples of **stopwords** that would have been removed while we were cleaning and preparing our data.

Each word in an email contributes to the probability of spam.

Let's use the Bayes theorem to calculate the probabilities below, starting with **P(Spam|million)**. This will tell us **how predictive** the word "millions" is in determining spam.

P(Spam | We found out that there is a total amount of $4.5 million…)

= P(Spam|we) * P(Spam|found) * P(Spam|total) * P(Spam|amount) * P(Spam|million) …

NOTE: Why did we leave out words like "that", "is", "a", and "of"? These are examples of **stopwords** that would have been removed while we were cleaning and preparing our data.

Let's use the Bayes Theorem to calculate this probability:

$$\boxed{P(Spam|million)} = \frac{P(million|Spam) * P(Spam)}{P(million)}$$

We don't know the probability of spam given "million", but thanks to our labelled training data, we do know the probability of "million" given spam!

We will calculate each of the 3 right-hand pieces, starting with P(million|Spam)

P(million|Spam)

What is the probability that "millions" occurs in a spam email?

How many spam emails in our training data set contain the word "millions"?

Probability of the word "millions" being in a spam email

$$= \frac{\text{sum(millions=1 \& spam=1)}}{\text{sum(spam=1)}}$$

$$P(\text{million}|\text{Spam}) = \frac{3}{4}$$

Email	millions	spam
1	1	1
2	0	0
3	1	0
4	0	1
5	1	1
6	1	1

P(Spam)

What is the probability that an email is spam?

How many emails are spam?

Email	millions	spam
1	1	1
2	0	0
3	1	0
4	0	1
5	1	1
6	1	1

Probability of an email being spam $= \dfrac{\text{sum(spam=1)}}{\text{sum(spam=1 or spam=0)}}$

$$P(\text{Spam}) = \frac{4}{6}$$

P(million)

What is the probability that an email contains the word "million"?

How many emails contain the word "million"?

Probability of an email containing the word "million" $= \dfrac{\text{sum(million=1)}}{\text{sum(million=1 or million=0)}}$

$\text{P(million)} = \dfrac{4}{6}$

Email	millions	spam
1	1	1
2	0	0
3	1	0
4	0	1
5	1	1
6	1	1

Now let's bring it all together!

$$P(\text{Spam}|\text{million}) = \frac{P(\text{million}|\text{Spam}) * P(\text{Spam})}{P(\text{million})}$$

$$= \frac{3/4 * 4/6}{4/6} = 0.75$$

Probability of this email being spam, given it has the word "millions" = 0.75

Email	millions	spam
1	1	1
2	0	0
3	1	0
4	0	1
5	1	1
6	1	1

Now we can predict spam!

Now we can predict spam, using a single word! Our model predicts that if the word "millions" is present, it is 75% spam. Since this is a classification problem, we round this up to 1.

Email	millions	spam	predicted
1	1	1	1
2	0	0	0
3	1	0	1
4	0	1	0
5	1	1	1
6	1	1	1

What about the other words?

But this model is simplistic. We should use the other words in the email too! We can repeat the process of calculating conditional probability for all the words in an email.

P(Spam | We found out that there is a total amount of $4.5 million…)

= P(Spam|we) * P(Spam|found) * P(Spam|total) * P(Spam|amount) * P(Spam|million) …

Email	"millions"	"FBI"	"amount"	...
1	1	1	0	
2	0	0	0	
3	1	0	1	
4	0	0	1	
5	1	0	1	
6	1	1	1	

What about multiple words?

Once we have conditional probabilities for all the words in an email, we **multiply** them together to get the probability that this email is spam, given all the words it contains.

P(Spam | We found out that there is a total amount of $4.5 million…)

= P(Spam|we) * P(Spam|found) * P(Spam|total) * P(Spam|amount) * P(Spam|million) …

Email	"millions"	"FBI"	"amount"	...
1	1	1	0	
2	0	0	0	
3	1	0	1	
4	0	0	1	
5	1	0	1	
6	1	1	1	

NOTE: What if a probability is 0? Then the total probability would be 0! To avoid this problem, Naive Bayes uses "**Laplace smoothing**," to ensure the probability is never 0. Read more here: https://en.wikipedia.org/wiki/Laplace_smoothing

Now you know how the Naive Bayes Algorithm works, let's examine some of its disadvantages and assumptions.

Naive Bayes "*naively*" assumes each word is **independent** to every other word.

This is inherited from our representation of the text as a **"bag-of-words"**. Remember, an email that said "it's not good, it's bad" would be considered the same as "it's not bad, it's good" to the Naive Bayes algorithm.

As a result, you can end up double-counting highly correlated words, like "dollars" and "millions," which would incorrectly push up the error rate.

How can we quantify algorithm performance?

Evaluating algorithm performance

Interpreting our results for a single word "millions"
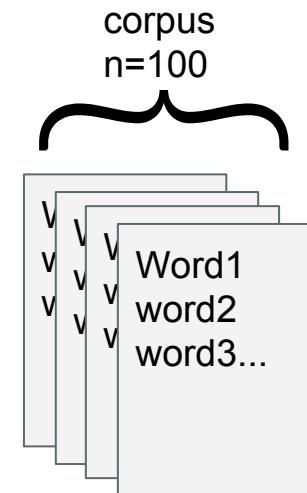
Classification

$P(S|W) = 75\%$

corpus
n=100

Word1
word2
word3...

Probability of email being spam given presence of word "millions" is 75%.

What is the accuracy of our model?

We turn to our training data to evaluate how our model did.

Accuracy:

$$\frac{\text{\# labels predicted correctly}}{\text{\# labels predicted}}$$

Email	millions	spam
1	1	1
2	0	0
3	1	0
4	0	1
5	1	1
6	1	1

What is the accuracy of our model?

Accuracy:

$$\frac{\text{\# labels predicted correctly}}{\text{\# labels predicted}}$$

$$= \frac{4}{6} = 66\%$$

Email	millions	spam	predicted
1	1	1	1
2	0	0	0
3	1	0	1
4	0	1	0
5	1	1	1
6	1	1	1

Recap: false positive & false negative

Spam classification task: is the email spam?

False positive:

Email is classified as spam when it is not.

False negative:

Email is not classified as spam when it is.

	They say you **did**	They say you **didn't**
You really did	*They are right!*	**"False Negative"**
You really didn't	**"False Positive"**	*They are right!*

In any classification model we work on, we will evaluate performance on FP, FN in addition to accuracy.

What is the false negative rate?

False negative rate:

incorrectly predicted as spam
—————————————————————
total predictions

Email	millions	spam	predicted
1	1	1	1
2	0	0	0
3	1	0	1
4	0	1	0
5	1	1	1
6	1	1	1

False negative rate

False negative rate:

$$\frac{\text{\# incorrectly predicted as spam}}{\text{\# total predictions}}$$

$$= \frac{1}{6} = 16\%$$

Email	millions	spam	predicted
1	1	1	1
2	0	0	0
3	1	0	1
4	0	1	0
5	1	1	1
6	1	1	1

What is the false positive rate?

False positive rate:

incorrectly predicted as non-spam
----
total predictions

Email	millions	spam	predicted
1	1	1	1
2	0	0	0
3	1	0	1
4	0	1	0
5	1	1	1
6	1	1	1

False positive rate

False positive rate:

$$\frac{\text{\# incorrectly predicted as non-spam}}{\text{\# total predictions}}$$

$$= \frac{1}{6} = 16\%$$

Email	millions	spam	predicted
1	1	1	1
2	0	0	0
3	1	0	1
4	0	1	0
5	1	1	1
6	1	1	1

We've just automated spam-detection (thank goodness!)

In the next module, we will look at another NLP algorithm.

End of theory