Natural Language Processing

Joe

galvanize

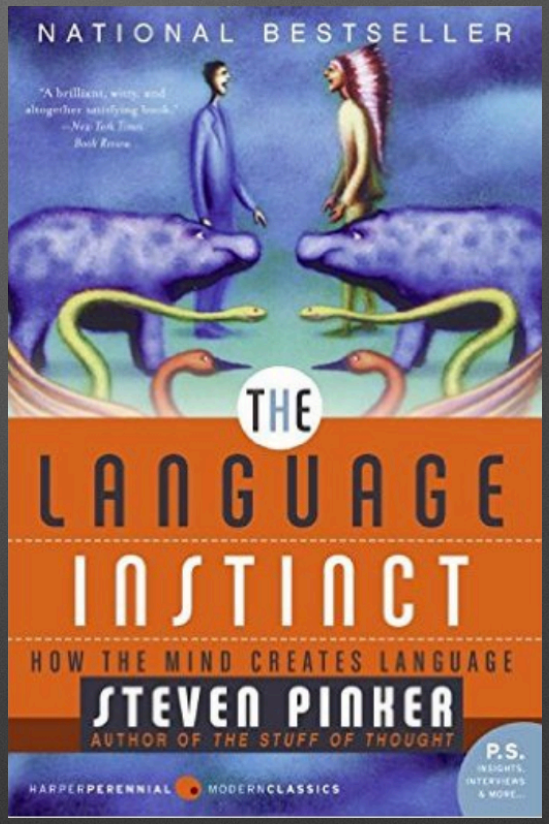
Morning Objectives

1. Define the NLP Problem
2. Contextualize Machine Learning within NLP
3. Describe basic dimensionality reduction techniques

Yet Another Book Recommendation…..

- Steven Pinker's excellent book, *The Language Instinct,* surveys a multitude of reasons why language is such an odd subject to learn, and I consider it a must read for anyone who hopes to specialize in NLP

- Added bonus: Steven Pinker is a phenomenal writer so it's a real pleasure to read.

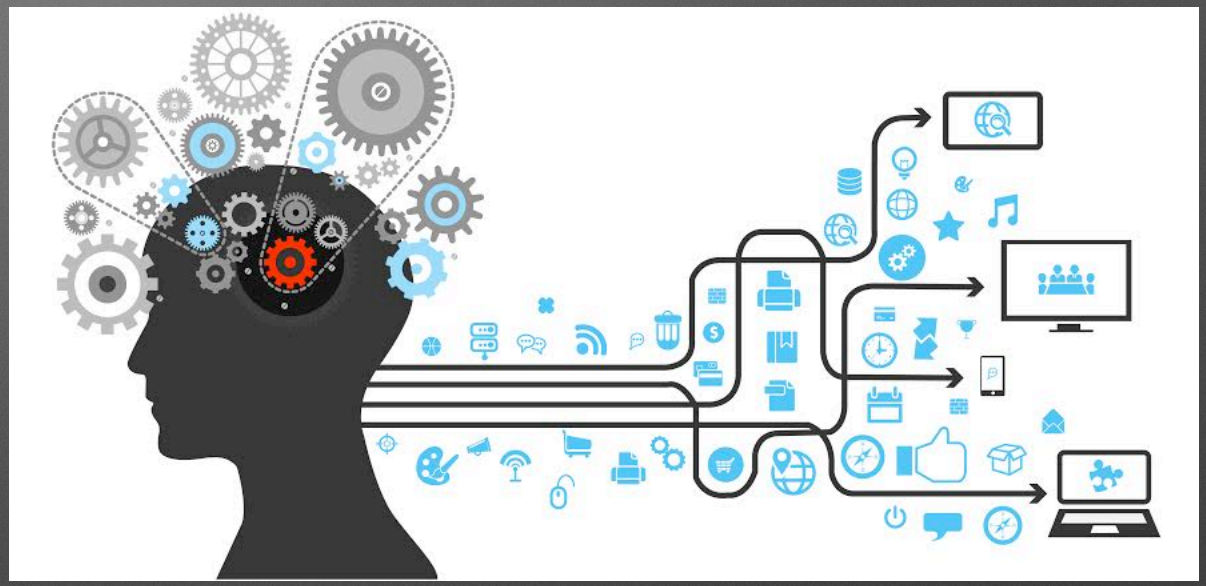


galvanize

The NLP Problem

Thinking Machines

- Early ‘Artificial Intelligence’:
  - Rapid Calculations
  - Memorization/Recall
  - Ex. playing chess



- Artificial Intelligence has undergone ‘winters’ of favor due to it’s inability to tackle human-easy, machine difficult tasks

- Such tasks, like NLP and object recognition in pictures, are particularly difficult because the heuristics commonly associated with ML do particularly poorly when applied to such tasks.

Differing Views of AI

- Embodied Cognition Hypothesis - cognition can only come from machines with sensory and motor skills
  - This means that machines must be able to understand nuance like context for them to be able to 'understand' text
  - Previously, this has failed due to the lack of computers ability to mimic human instinct

- An alternate view favors 'Logical Propositions', or high level mental entities associated with groups of words
  - This can be achieved through Linear Algebra, as we will do later in the week with PCA, SVD (called LSA in this context), and NMF

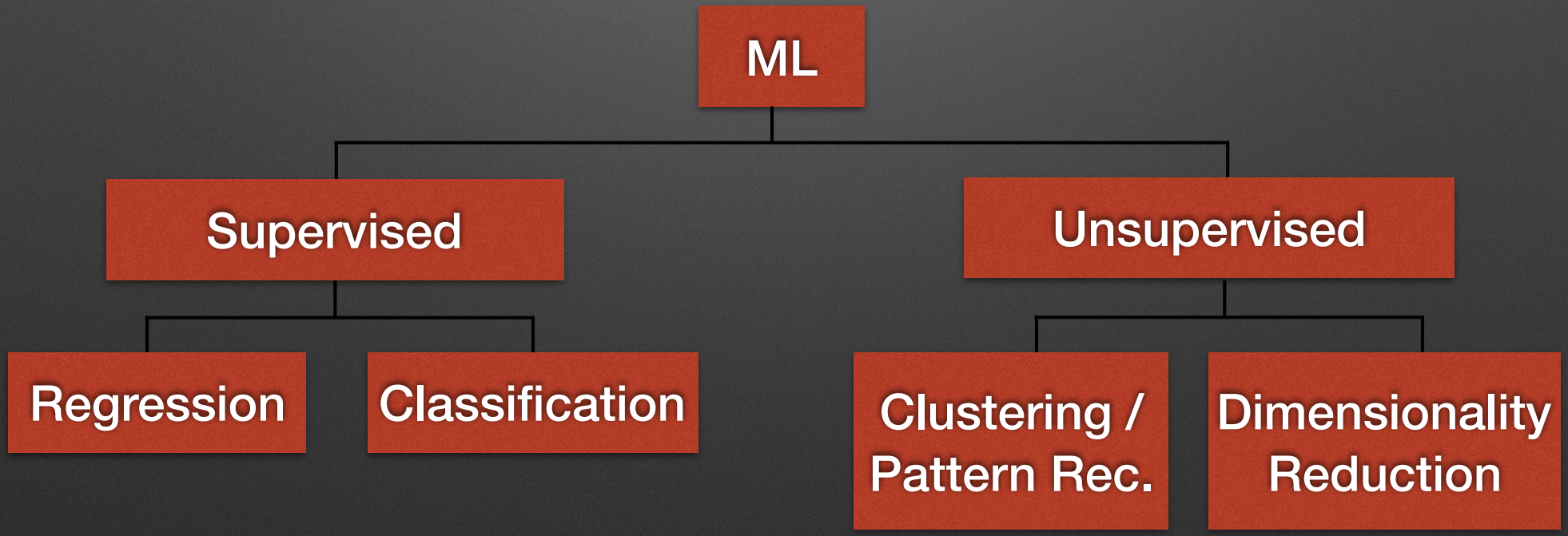
- Neural Networks have promise of fulfilling both representations

How ML Surfaces in NLP

NLP Can be Many Things

We started by categorizing machine learning into several groups of supervised and unsupervised techniques. NLP, as it turns out, spans all of these groups.

NLP is simply ML on human text, and we'll touch upon a small portion of it in the next few days.

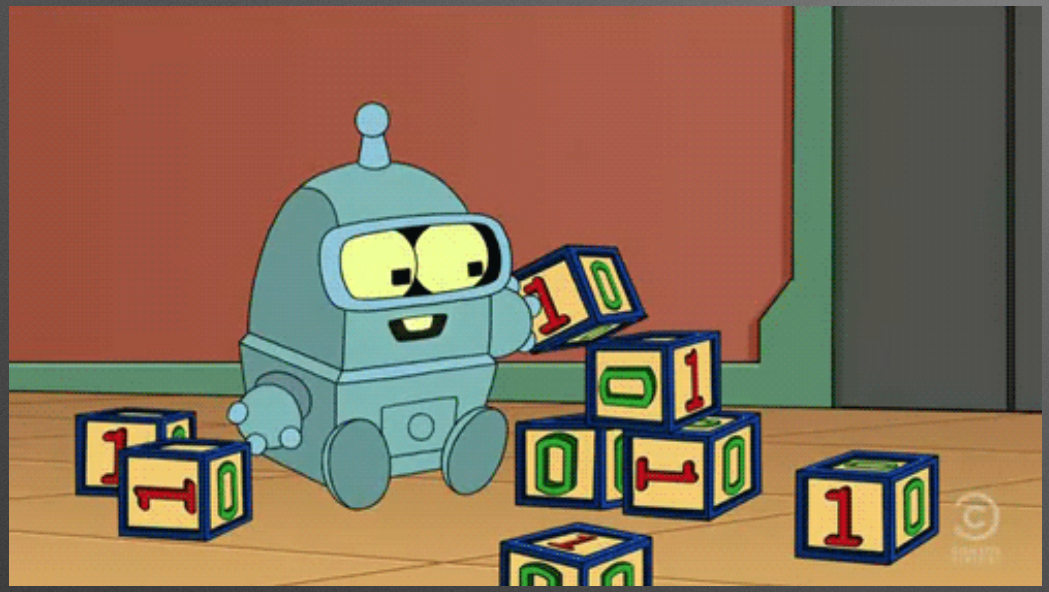


Common NLP Task by Category

- Regression - Sentiment Analysis, Subjectivity Analysis
- Classification - Grouping documents for news, flagging documents as relevant or not for search (much larger subject of document retrieval)
- Pattern Recognition - Event detection in document streams, part of speech recognition
- Dimensionality Reduction - Simple tasks like lemmatization, very challenging task like LSA, LDA, PCA, ect.

Sparse Document Representation

Text Representations



- When computers learn to speak, they only learn 1's and 0's.
- Despite this, there are many ways of representing text that vary based on the task you hope to accomplish.

- As such, even more of the heavy lifting associated with NLP is done in the feature engineering phase.
- Much of NLP is based on the "semantic hypothesis":

"Statistical patterns of human word usage can be used to figure out what people mean"

Document Vectorization

- Consider the following sentences.

```
l1 = "I am a data scientist teaching at Galvanize"
l2 = "You are students learning at Galvanize"
l3 = "Dan is also a data scientist teaching at Galvanize"
```

- We can transform these to a vector representing the lines in a count-vectorizer or a “bag of words” representation.

```
>>> X.toarray()
array([[0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0],
       [0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0]], dtype=int64)
```

- The term comes from the mathematical object ‘bag’ which is a multi-set (a set that allows duplicates)

Bag of Words

- The bag of word heuristics:
  - Word frequency vectors can be used to assess semantic relevance of documents
  - Frequencies of individual words are sufficiently indicative of the semantic association between two documents

- Problems - consider the following:

```
>>> d1 = "Jack is quicker than Jill"
>>> d2 = "Jill is quicker than Jack"
```

1. BOW sees these as the same sentence despite them conveying opposite meaning
2. BOW will have vectors with greater magnitude for longer documents
3. Every word in this sentence is of equal importance

N-Grams

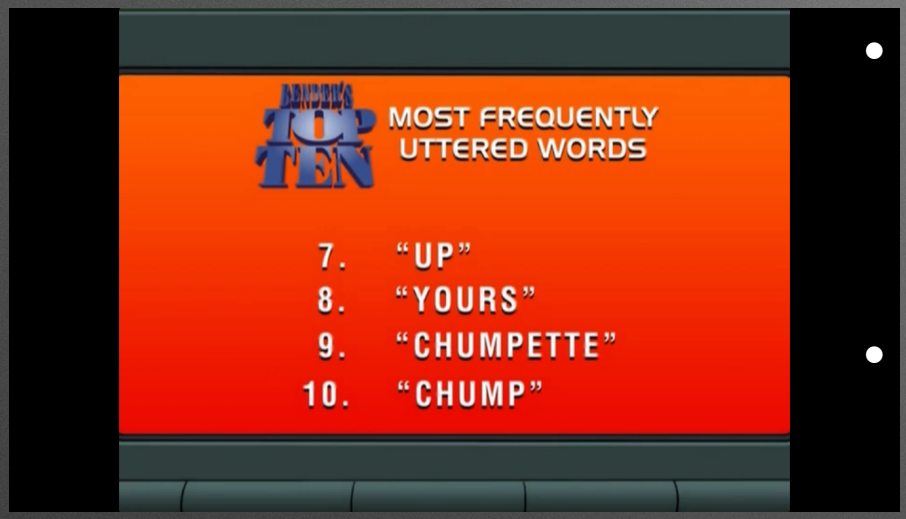
- N-Grams aim to retrieve some of the contextual clues lost in bag of words by not thinking of single words as the primary tokens

- Consider instead multiple tokens:

```
>>> d2 = "Jill is quicker than Jack"
>>> list(zip(*[d2.lower().split()[i:] for i in range(2)]))
[('jill', 'is'), ('is', 'quicker'), ('quicker', 'than'), ('than', 'jack')]
>>> list(zip(*[d2.lower().split()[i:] for i in range(3)]))
[('jill', 'is', 'quicker'), ('is', 'quicker', 'than'), ('quicker', 'than', 'jack')]
```

- Drawbacks are that it will expand the dimensionality of the dataset (which is already a big problem in NLP)

TF-IDF

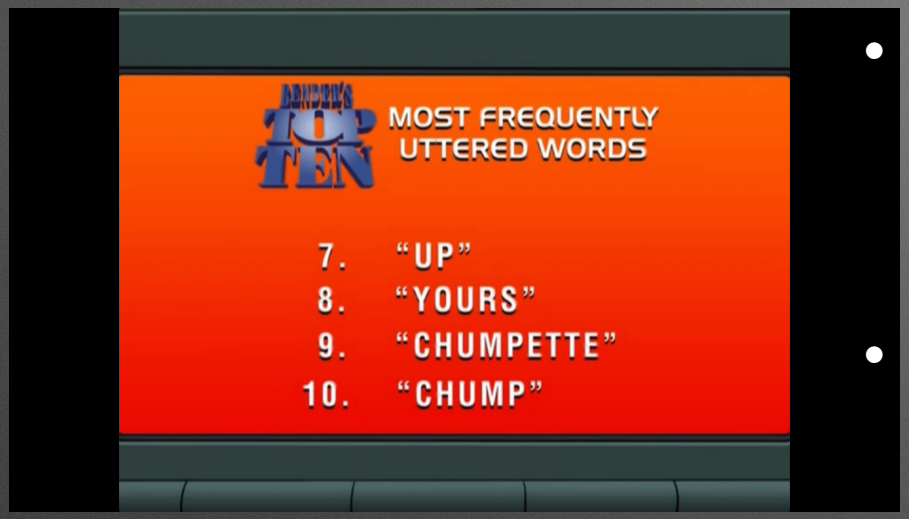


- Consider the following line, that may be in a menu:

```
ml = "Savory Rutabaga and Chicken Tacos"
```

- If I'm trying to perform a task like information retrieval, I want to identify which words are most important.

TF-IDF



- Consider the following line, that may be in a menu:

```
ml = "Savory Rutabaga and Chicken Tacos"
```

- If I'm trying to perform a task like information retrieval, I want to identify which words are most important.

$tf_{t,d} = n_{t,d}$

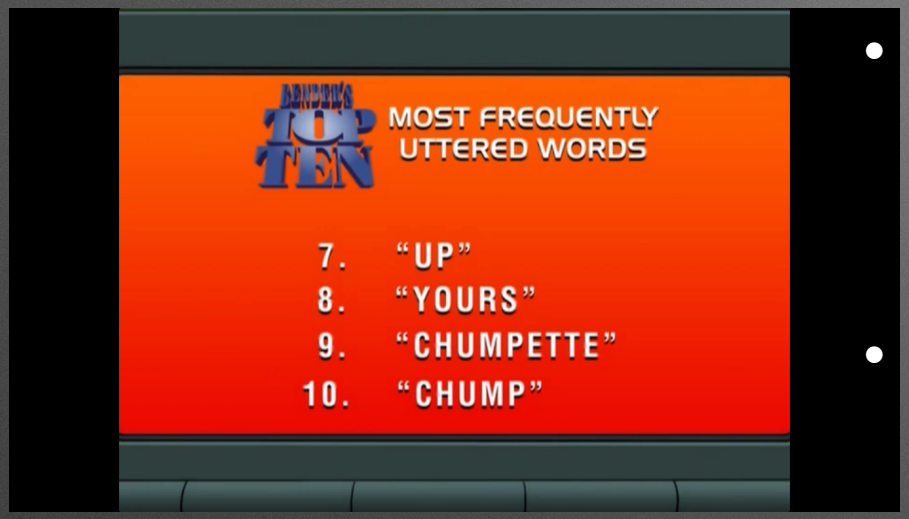
or normalized

$$tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}}$$

$tf_{t,d}$ is the term frequency for term t in document d

$n_{t/k,d}$ is the number of times t/k exists in d

TF-IDF



- Consider the following line, that may be in a menu:

```
ml = "Savory Rutabaga and Chicken Tacos"
```

- If I'm trying to perform a task like information retrieval, I want to identify which words are most important.

$tf_{t,d} = n_{t,d}$

or normalized

$$tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}}$$

$tf_{t,d}$ is the term frequency for term t in document d

$n_{t/k,d}$ is the number of times t/k exists in d

$$idf_t = \log(\frac{N}{n_t})$$

or smoothed

$$idf_t = \log(1 + \frac{N}{n_t})$$

Basic Dimensionality Reduction

Information Overload

- If we look at text vectors, we end up with huge (O(10^{5-8}) depending on corpus size) vocabulary sizes. It's useful for many reasons to reduce this. Additionally, under the semantic hypothesis, words like 'are' and 'is' convey the same meaning, so for many reasons, we put forth a good bit of effort into the cleaning of text.

  See N.B. 1 for some basics.

- Note exact processes vary: http://text-processing.com/demo/tokenize/

Morning Objectives

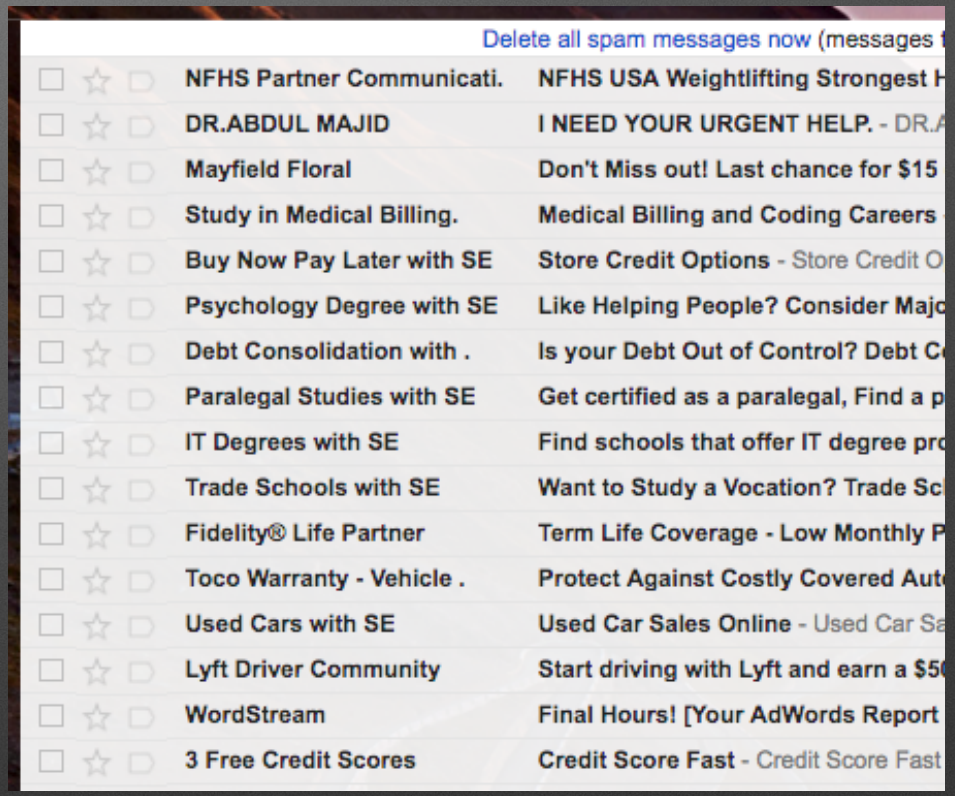
1. Define the NLP Problem
2. Describe efficient word document representations
3. Describe basic dimensionality reduction techniques

Afternoon Objectives

1. Outline what make the Naive Bayes algorithm Naive
2. Demonstrate how Laplace smoothing make Naive Bayes better
3. Compare to Random Forest on a *simple* text classification task

Naive Bayes Motivation

- The fundamental issue we will grapple with all week is the curse of dimensionality
- Particularly in the case of text classification two problems will be very prevalent:
  - Imbalanced Classes
  - feature >> observations



Bayesian Probability

$$P(spam|email) = \frac{P(email|spam)P(spam)}{P(email)}$$

Bayesian Probability

$$P(spam|email) = \frac{P(email|spam)P(spam)}{P(email)}$$

This ends up just being the number of spam/total number of email.

Bayesian Probability

$$P(spam|email) = \frac{P(email|spam)P(spam)}{P(email)}$$

The question becomes, how do we estimate the probability of observing an email, given we have a spam message?

The 'Naive' Assumption in Naive Bayes

$$P(email|spam) = \prod_{i=1}^{p} P(word_i|spam)$$

$$P(word_i|spam) = \frac{\#\ of\ word_i\ in\ spam\ emails}{\#of\ total\ words\ in\ spam\ emails}$$

The 'Naive' Assumption in Naive Bayes

$$P(email|spam) = \prod_{i=1}^{p} P(word_i|spam)$$

$$P(word_i|spam) = \frac{\#\ of\ word_i\ in\ spam\ emails}{\#of\ total\ words\ in\ spam\ emails}$$

The Naive Assumption ends up being…OK. Certain phrases are certainly not independent, but these effects are minimal.

The 'Naive' Assumption in Naive Bayes

$$P(email|spam) = \prod_{i=1}^{p} P(word_i|spam)$$

$$P(word_i|spam) = \frac{\#\ of\ word_i\ in\ spam\ emails}{\#of\ total\ words\ in\ spam\ emails}$$

What happens if a word has not been encountered?

Laplace Smoothing

$$P(word_i|spam) = \frac{\#\ of\ word_i\ in\ spam\ emails + \alpha}{\#\ of\ words\ in\ spam\ emails + \alpha p}$$

- *p* is the number of features (i.e. number of words in your corpus)
- α ~ 0.01

Afternoon Objectives

1. Outline what make the Naive Bayes algorithm Naive
2. Demonstrate how Laplace smoothing make Naive Bayes better
3. Compare to Random Forest on a *simple* text classification task