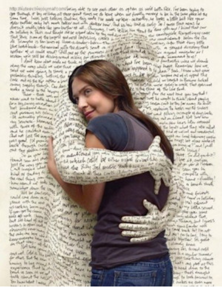
Words

and their abstractions

Jon Dehdari

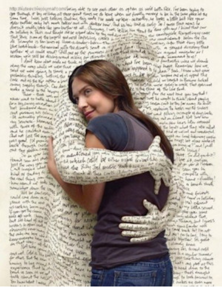
November 22, 2015

Too Many Words!



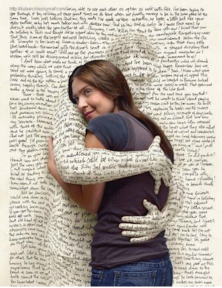
- Languages have too many words for statistical models of language

Too Many Words!



- Languages have too many words for statistical models of language
- We need some way to generalize them

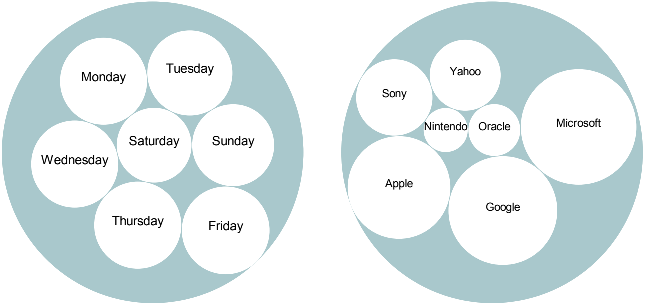
Too Many Words!



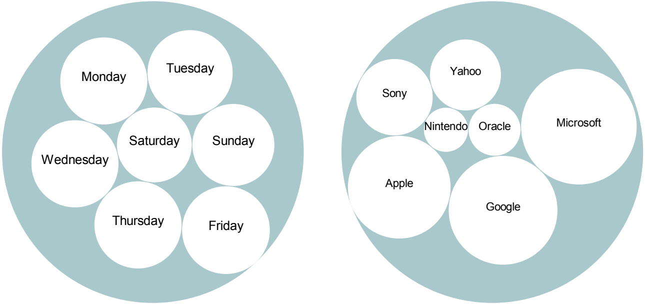
- Languages have too many words for statistical models of language
- We need some way to generalize them
- Let's treat some words like other words

- Words can be grouped together into equivalence classes to help reduce data sparsity and better generalize the data.

- Words can be grouped together into equivalence classes to help reduce data sparsity and better generalize the data.



- Words can be grouped together into equivalence classes to help reduce data sparsity and better generalize the data.



- Hand-crafted equivalence classes are called **part-of-speech tags**, and automatically induced equivalence classes are usually called **word classes** or **word clusters**

Parts of Speech and Word Clusters

- Part-of-speech example:

Pierre	Vinken	,	61	years	old	,	will	join	the	board
NNP	NNP	,	CD	NNS	JJ	,	MD	VB	DT	NN

- Word cluster example:

Pierre	Vinken	,	61	years	old	,	will	join	the	board
344	0	283	94	348	274	283	367	360	71	390

Parts of Speech and Word Clusters

- Part-of-speech example:

Pierre	Vinken	,	61	years	old	,	will	join	the	board
NNP	NNP	,	CD	NNS	JJ	,	MD	VB	DT	NN

- Word cluster example:

Pierre	Vinken	,	61	years	old	,	will	join	the	board
344	0	283	94	348	274	283	367	360	71	390

Differences:

- Parts of speech have human-readable labels (eg. NN, VB), while word clusters usually just have numbers
- A word can have more than one part of speech (which depends on the context), while a word usually has just one word class

Supervised, Unsupervised, Semi-supervised Learning

- **Supervised learning** uses **manually-annotated data**

Supervised, Unsupervised, Semi-supervised Learning

- **Supervised learning** uses **manually-annotated data**
    - It's usually evaluated on **accuracy** (or related idea) of 'correct' label, given unannotated test input

Supervised, Unsupervised, Semi-supervised Learning

- **Supervised learning** uses **manually-annotated data**
  - It's usually evaluated on **accuracy** (or related idea) of 'correct' label, given unannotated test input
  - The data's[1] usually expensive and small

Supervised, Unsupervised, Semi-supervised Learning

- **Supervised learning** uses **manually-annotated data**
  - It's usually evaluated on **accuracy** (or related idea) of 'correct' label, given unannotated test input
  - The data's[1] usually expensive and small

- **Unsupervised learning** uses **unannotated data**

Supervised, Unsupervised, Semi-supervised Learning

- **Supervised learning** uses **manually-annotated data**
    - It's usually evaluated on **accuracy** (or related idea) of 'correct' label, given unannotated test input
    - The data's[1] usually expensive and small

- **Unsupervised learning** uses **unannotated data**
    - It's usually evaluated on the probability of the unannotated test input ($\propto$ **perplexity**), or a downstream task

Supervised, Unsupervised, Semi-supervised Learning

- **Supervised learning** uses **manually-annotated data**

  - It's usually evaluated on **accuracy** (or related idea) of 'correct' label, given unannotated test input
  - The data's[1] usually expensive and small

- **Unsupervised learning** uses **unannotated data**
  - It's usually evaluated on the probability of the unannotated test input ($\propto$ **perplexity**), or a downstream task
  - The data's usually big and noisy. Just like the world around us.

Supervised, Unsupervised, Semi-supervised Learning

- **Supervised learning** uses **manually-annotated data**

  - It's usually evaluated on **accuracy** (or related idea) of 'correct' label, given unannotated test input
  - The data's[1] usually expensive and small

- **Unsupervised learning** uses **unannotated data**
  - It's usually evaluated on the probability of the unannotated test input ($\propto$ **perplexity**), or a downstream task
  - The data's usually big and noisy. Just like the world around us.
- **Semi-supervised learning** uses **both** unannotated and annotated data
  - It's usually evaluated just like supervised learning tasks

[1] That's right: "data is".

Words *vs.* Word Classes

- Using word classes reduces the number of parameters in language models, which means **generalization**

Words *vs.* Word Classes

- Using word classes reduces the number of parameters in language models, which means **generalization**
- But, some sequences of words are **lexicalized**, meaning they are based on the specific words involved

Words *vs.* Word Classes

- Using word classes reduces the number of parameters in language models, which means **generalization**
- But, some sequences of words are **lexicalized**, meaning they are based on the specific words involved
- For example: "It's raining cats and ____"

Words *vs.* Word Classes

- Using word classes reduces the number of parameters in language models, which means **generalization**
- But, some sequences of words are **lexicalized**, meaning they are based on the specific words involved
- For example: "It's raining cats and ____"
- Using word-based language models, the next word will probably be '*dogs*'

Words *vs.* Word Classes

- Using word classes reduces the number of parameters in language models, which means **generalization**
- But, some sequences of words are **lexicalized**, meaning they are based on the specific words involved
- For example: "It's raining cats and ____"
- Using word-based language models, the next word will probably be '*dogs*'
- But class-based LMs only see something like "PRP VBZ VBG NNS CC ____"
- So they would predict something like '*shares*', if they were trained on the WSJ corpus

How Many Word Classes Should I Use?

The more word classes you use, the closer you get to a word-based model

How Many Word Classes Should I Use?

The more word classes you use, the closer you get to a word-based model

If you use a class-based LM by itself, more word classes is usually better, especially if you have a lot of training data

How Many Word Classes Should I Use?

The more word classes you use, the closer you get to a word-based model

If you use a class-based LM by itself, more word classes is usually better, especially if you have a lot of training data

However if you interpolate a class-based LM with a word-based LM, fewer word classes is usually better, because you get complementary information

How Can You Cluster Words?

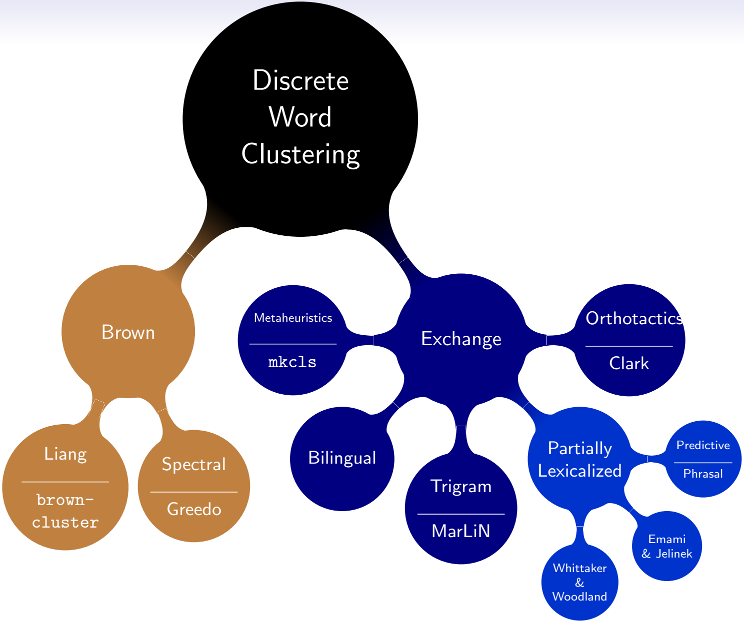
- If you represent words as vectors of real numbers, you can use general clustering algorithms like k-means clustering agglomerative clustering

How Can You Cluster Words?

- If you represent words as vectors of real numbers, you can use general clustering algorithms like *k*-means clustering agglomerative clustering
- You can also use discrete versions of these two algorithms, to cluster words directly from plaintext

How Can You Cluster Words?

- If you represent words as vectors of real numbers, you can use general clustering algorithms like k-means clustering agglomerative clustering
- You can also use discrete versions of these two algorithms, to cluster words directly from plaintext
- Discrete agglomerative word clustering is usually called **Brown clustering**
- Discrete k-means word clustering is usually called **exchange algorithm clustering**


Discrete Word Clustering
Brown
Liang
brown-cluster
Spectral
Greedo
Exchange
Metaheuristics
mkcls
Orthotactics
Clark
Bilingual
Trigram
MarLiN
Partially Lexicalized
Predictive
Phrasal
Emami & Jelinek
Whittaker & Woodland

Brown Clustering (hierarchical clusters)

1. Assign the most frequent words to their own class

Brown Clustering (hierarchical clusters)

1. Assign the most frequent words to their own class
2. For the remaining words, assign the next most frequent word to the class giving the best likelihood of the training data

Brown Clustering (hierarchical clusters)

1. Assign the most frequent words to their own class
2. For the remaining words, assign the next most frequent word to the class giving the best likelihood of the training data
3. There is no step 3.

Brown Clustering (hierarchical clusters)

1. Assign the most frequent words to their own class
2. For the remaining words, assign the next most frequent word to the class giving the best likelihood of the training data
3. There is no step 3.
4. Optionally recursively merge the remaining classes, again based on likelihood

Brown Clustering (hierarchical clusters)

1. Assign the most frequent words to their own class
2. For the remaining words, assign the next most frequent word to the class giving the best likelihood of the training data
3. There is no step 3.
4. Optionally recursively merge the remaining classes, again based on likelihood

- Above is Percy-style Brown clustering, which can be more efficient than the original algorithm

Brown Clustering (hierarchical clusters)

1. Assign the most frequent words to their own class
2. For the remaining words, assign the next most frequent word to the class giving the best likelihood of the training data
3. There is no step 3.
4. Optionally recursively merge the remaining classes, again based on likelihood

- Above is Percy-style Brown clustering, which can be more efficient than the original algorithm
- The time complexity is $\mathcal{O}(|V| \times |C|^2)$
- $|V|$ is the size of the vocabulary
  $|C|$ is the number of word classes

Brown Clustering (hierarchical clusters)

1. Assign the most frequent words to their own class
2. For the remaining words, assign the next most frequent word to the class giving the best likelihood of the training data
3. There is no step 3.
4. Optionally recursively merge the remaining classes, again based on likelihood

- Above is Percy-style Brown clustering, which can be more efficient than the original algorithm
- The time complexity is $\mathcal{O}(|V| \times |C|^2)$
- $|V|$ is the size of the vocabulary
  $|C|$ is the number of word classes
- Thus it's fairly fast for small clusters (< 400), but slow for large clusters (> 800)

Exchange Algorithm (flat clusters)

1. Randomly assign each word to a class

Exchange Algorithm (flat clusters)

1. Randomly assign each word to a class
2. For each word, change its word class to the one giving the best likelihood of the training data

Exchange Algorithm (flat clusters)

1. Randomly assign each word to a class
2. For each word, change its word class to the one giving the best likelihood of the training data
3. Do the last step for a few times (maybe 10–20 iterations)

Exchange Algorithm (flat clusters)

1. Randomly assign each word to a class
2. For each word, change its word class to the one giving the best likelihood of the training data
3. Do the last step for a few times (maybe 10–20 iterations)

- Thus the basic time complexity is $\mathcal{O}(|V| \times |C| \times i)$
- $|V|$ is the size of the vocabulary
  $|C|$ is the number of word classes
  i is the number of iterations

Exchange Algorithm (flat clusters)

1. Randomly assign each word to a class
2. For each word, change its word class to the one giving the best likelihood of the training data
3. Do the last step for a few times (maybe 10–20 iterations)

- Thus the basic time complexity is $\mathcal{O}(|V| \times |C| \times i)$
- $|V|$ is the size of the vocabulary
  $|C|$ is the number of word classes
  i is the number of iterations
- There's a little more added complexity is how you calculate training-set likelihood

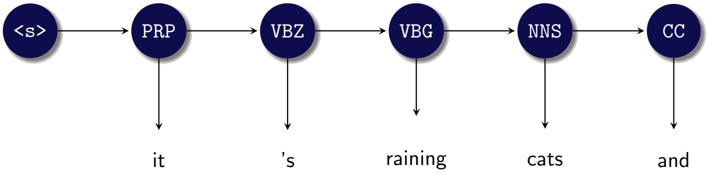
Class-based Language Models

- So how do we use word classes as a language model?

Class-based Language Models

- So how do we use word classes as a language model?
- The most common form (c_i is the word class of word w_i):

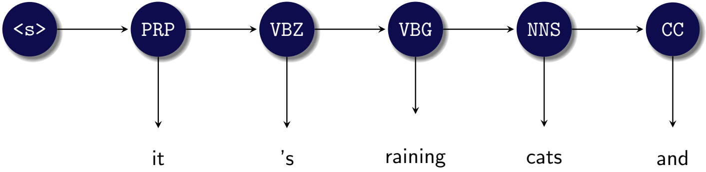
$$P(w_i|w_{i-1}) \triangleq P(w_i|c_i)\,P(c_i|c_{i-1})$$



Class-based Language Models

- So how do we use word classes as a language model?
- The most common form (c_i is the word class of word w_i):

$$P(w_i|w_{i-1}) \triangleq P(w_i|c_i)\,P(c_i|c_{i-1})$$



- Notice c_i, which is called a **bottleneck variable**
- The history is 'squeezed' through this point, in order to summarize and generalize the history

Predictive Exchange and Conditional Exchange

- The previous model is used in both Brown clustering and exchange algorithm clustering to determine the likelihood of the training set. We can use different models as well.

Predictive Exchange and Conditional Exchange

- The previous model is used in both Brown clustering and exchange algorithm clustering to determine the likelihood of the training set. We can use different models as well.
- The *predictive exchange algorithm* uses this model:

$$P(w_i|w_{i-1}) \triangleq P(w_i|c_i)\, P(c_i|w_{i-1})$$

- The *conditional exchange algorithm* uses this model:

$$P(w_i|w_{i-1}) \triangleq P(w_i|c_{i-1})$$