CS 4650/7650: Natural Language Processing

Vector Semantics

Diyi Yang

Slides from Dan Jurafsky and Michael Collins, and many others
Announcements

- HW1 Regrade Due Jan 29th
- HW2 Due on Feb 3rd, 3pm ET
What are various ways to represent the meaning of a word?
Q: What’s the meaning of life?
A: LIFE
Lexical Semantics

How to represent the meaning of a word?

- Words, lemmas, senses, definitions

http://www.oed.com
Lemma “Pepper”

- Sense 1:
  - Spice from pepper plant

- Sense 2:
  - The pepper plant itself

- Sense 3:
  - Another similar plant (Jamaican pepper)

- Sense 4:
  - Another plant with peppercorns (California pepper)

- Sense 5:
  - Capsicum (i.e., bell pepper, etc)

A sense or “concept” is the meaning component of a word.
Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definitions
  - Relationships between words or senses
Relation: Synonymity

- Synonyms have the same meaning in some or all contexts.
  - Filbert/hazelnut
  - Couch/sofa
  - Big/large
  - Automobile/car
  - Vomit/throw up
  - Water/H20
Synonyms have the same meaning in some or all contexts.

Note that there are probably no examples of perfect synonymy:

- Even if some aspects of meaning are identical
- Still may not preserve the acceptability based on notions of politeness, slang, register, genre, etc.
Relation: Antonymy

- Senses that are opposites with respect to one feature of meaning
  - Otherwise, they are very similar!
    - Dark/light short/long fast/slow rise/fall
    - Hot/cold up/down in/out
  - Many formally: antonyms can
    - Define a binary opposition or be at opposite ends of a scale
      - Long/short, fast/slow
    - Be reverse:
      - Rise/fall, up/down
Relation: Similarity

- Words with similar meanings
- Not synonyms, but sharing some element of meaning
  - Car, bicycle
  - Cow, horse
### Ask Humans How Similar 2 Words Are

<table>
<thead>
<tr>
<th>Word 1</th>
<th>Word 2</th>
<th>similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanish</td>
<td>disappear</td>
<td>9.8</td>
</tr>
<tr>
<td>behave</td>
<td>obey</td>
<td>7.3</td>
</tr>
<tr>
<td>belief</td>
<td>impression</td>
<td>5.95</td>
</tr>
<tr>
<td>muscle</td>
<td>bone</td>
<td>3.65</td>
</tr>
<tr>
<td>modest</td>
<td>flexible</td>
<td>0.98</td>
</tr>
<tr>
<td>hole</td>
<td>agreement</td>
<td>0.3</td>
</tr>
</tbody>
</table>

SimLex-999 dataset (Hill et al., 2015)
Relation: Word Relatedness

- Also called “word association”
- Words be related in any way, perhaps via a **semantic field**

A semantic field is a set of words which cover a particular semantic domain and bear structured relations with each other.
A semantic field is a set of words which cover a particular semantic domain and bear structured relations with each other.
Relation: Word Relatedness

- Also called “word association”
- Words be related in any way, perhaps via a semantic field
  - Car, bicycle: similar
  - Car, gas: related, not similar
  - Coffee, cup: related, not similar
Relation: Superordinate/Subordinate

- One sense is a **subordinate** of another if the first sense is more specific, denoting a subclass of the other
  - *Car is a subordinate of vehicle*
  - *Mango is a subordinate of fruit*
- Conversely superordinate
  - *Vehicle is a superordinate of car*
  - *Fruit is a superordinate of mango*
Taxonomy

Superordinate  Basic  Subordinate

- chair
  - office chair
  - piano chair
  - rocking chair

- furniture

- lamp
  - torchiere
  - desk lamp

- table
  - end table
  - coffee table
Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definitions
  - Relationships between words or senses
  - Taxonomy relationships
  - Word similarity, word relatedness
Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definitions
  - Relationships between words or senses
  - Taxonomy relationships
  - Word similarity, word relatedness
  - Semantic frames and roles
Semantic Frame

- A set of words that denote perspectives or participants in a particular type of event
  - “buy” (the event from the perspective of the buyer)
  - “sell” (from the perspective of the seller)
  - “pay” (focusing on the monetary aspect)
  - John hit Bill
  - Bill was hit by John

- Frames have semantic roles (like buyer, sellers, goods, money) and words in a sentence can take on those roles
Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definitions
  - Relationships between words or senses
  - Taxonomy relationships
  - Word similarity, word relatedness
  - Semantic frames and roles
  - Connotation and sentiment
Connotation and Sentiment

- Connotations refer to the aspects of a word’s meaning that are related to a writer or reader’s emotions, sentiment, opinions, or evaluations.
  - happy vs. sad
  - great, love vs. terrible, hate
- Three dimensions of affective meaning
  - **Valence**: the pleasantness of the stimulus
  - **Arousal**: the intensity of emotion
  - **Dominance**: the degree of control exerted by the stimulus

<table>
<thead>
<tr>
<th></th>
<th>Valence</th>
<th>Arousal</th>
<th>Dominance</th>
</tr>
</thead>
<tbody>
<tr>
<td>courageous</td>
<td>8.05</td>
<td>5.5</td>
<td>7.38</td>
</tr>
<tr>
<td>music</td>
<td>7.67</td>
<td>5.57</td>
<td>6.5</td>
</tr>
<tr>
<td>heartbreak</td>
<td>2.45</td>
<td>5.65</td>
<td>3.58</td>
</tr>
<tr>
<td>cub</td>
<td>6.71</td>
<td>3.95</td>
<td>4.24</td>
</tr>
<tr>
<td>life</td>
<td>6.68</td>
<td>5.59</td>
<td>5.89</td>
</tr>
</tbody>
</table>
Lexical Semantics

- How should we represent the meaning of the word?
  1. Words, lemmas, senses, definitions
  2. Relationships between words or senses
  3. Taxonomy relationships
  4. Word similarity, word relatedness
  5. Semantic frames and roles
  6. Connotation and sentiment
Electronic Dictionaries

WordNet

```python
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

(here, for good):

*S:* (adj) full, good
*S:* (adj) estimable, good, honorable, respectable
*S:* (adj) beneficial, good
*S:* (adj) good, just, upright
*S:* (adj) adept, expert, good, practiced, proficient, skillful
*S:* (adj) dear, good, near
*S:* (adj) good, right, ripe
...
*S:* (adv) well, good
*S:* (adv) thoroughly, soundly, good
*S:* (n) good, goodness
*S:* (n) commodity, trade good, good
Problems with Discrete Representation

- Too coarse
  - Expert $\rightarrow$ skillful
- Sparse
  - Wicked, badass, ninja
- Subjective
- Expensive
- Hard to compute word relationships

\[
\begin{align*}
\text{expert} & \quad [0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \\
\text{skillful} & \quad [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \textbf{1} \ 0 \ 0 \ 0 \ 0]
\end{align*}
\]
Vector Semantics
Distributional Hypothesis

- “The meaning of a word is its use in the language” [Wittgenstein PI 43]
- “You shall know a word by the company it keeps” [Firth 1957]
- “If A and B have almost identical environments we say that they are synonyms” [Harris 1954]
Example: What does OngChoi Mean?

- Suppose you see those sentences:
  - Ongchoi is delicious sautéed with garlic
  - Ongchoi is superb over rice
  - Ongchoi leaves with salty sauces

- And you’ve also seen these:
  - ... spinach sautéed with garlic over rice
  - Chard stems and leaves are delicious
  - Collard greens and other salty leafy greens
Example: What does OngChoi Mean?

- Suppose you see those sentences:
  - Ongchoi is delicious *sautéed with garlic*
  - Ongchoi is superb *over rice*
  - Ongchoi *leaves* with salty sauces
- And you’ve also seen these:
  - ... spinach *sautéed with garlic over rice*
  - Chard stems and *leaves* are delicious
  - Collard greens and other *salty* leafy greens
Word Embedding Representations

- Count-based
  - Tf-idf, PPMI
- Class-based
  - Brown Clusters
- Distributed prediction-based embeddings
  - Word2vec, FastText
- Distributed contextual (token) embeddings from language models
  - Elmo, BERT
- + many more variants
  - Multilingual embeddings, multi-sense embeddings, syntactic embeddings, etc ...
### Term-Document Matrix

<table>
<thead>
<tr>
<th>Term</th>
<th>As You Like It</th>
<th>Twelfth Night</th>
<th>Julius Caesar</th>
<th>Henry V</th>
</tr>
</thead>
<tbody>
<tr>
<td>battle</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>solider</td>
<td>2</td>
<td>80</td>
<td>62</td>
<td>89</td>
</tr>
<tr>
<td>fool</td>
<td>36</td>
<td>58</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>clown</td>
<td>20</td>
<td>15</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Context = appearing in the same document.
### Term-Document Matrix

<table>
<thead>
<tr>
<th>Term</th>
<th>As You Like It</th>
<th>Twelfth Night</th>
<th>Julius Caesar</th>
<th>Henry V</th>
</tr>
</thead>
<tbody>
<tr>
<td>battle</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>17</td>
</tr>
<tr>
<td>solider</td>
<td>2</td>
<td>80</td>
<td>62</td>
<td>89</td>
</tr>
<tr>
<td>fool</td>
<td>36</td>
<td>58</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>clown</td>
<td>20</td>
<td>15</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

**Vector Space Model:**
Each document is represented as a column vector of length four
Two words are “similar” in meaning if their context vectors are similar.
• Similarity == relatedness
## Count-Based Representations

<table>
<thead>
<tr>
<th></th>
<th>As You Like It</th>
<th>Twelfth Night</th>
<th>Julius Caesar</th>
<th>Henry V</th>
</tr>
</thead>
<tbody>
<tr>
<td>battle</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>good</td>
<td>114</td>
<td>80</td>
<td>62</td>
<td>89</td>
</tr>
<tr>
<td>fool</td>
<td>36</td>
<td>58</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>wit</td>
<td>20</td>
<td>15</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

**Counts: term-frequency**
- Remove stop words
- Use $\log_{10}(tf)$
- Normalize by document length
TF-IDF

What to do with words that are evenly distributed across many documents?

\[ tf_{t,d} = \log_{10}(\text{count}(t,d) + 1) \]

\[ idf_i = \log_{10}\left(\frac{N}{df_i}\right) \]

- Total # of docs in collection
- # of docs that have word i
TF-IDF

- What to do with words that are evenly distributed across many documents?
  \[ t_{f,t,d} = \log_{10}(\text{count}(t,d) + 1) \]
  \[ idf_i = \log_{10}\left(\frac{N}{df_i}\right) \]
  - Total # of docs in collection
  - # of docs that have word \(i\)

- Words like “the” or “good” have very low idf

\[ w_{t,d} = t_{f,t,d} \times idf_i \]
Pointwise Mutual Information (PMI)

- Do word $w$ and $c$ co-occur more than if they were independent?

$$\text{PMI}(w, c) = \log_2 \frac{p(w, c)}{p(w)p(c)}$$
Positive Pointwise Mutual Information (PPMI)

\[ \text{PPMI}(w, c) = \max (\log_2 \frac{p(w, c)}{p(w)p(c)}, 0) \]
Positive Pointwise Mutual Information (PPMI)

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values
  - Give rare words slightly higher probabilities $\alpha=0.75$

\[
\text{PPMI}_\alpha(w, c) = \max(\log_2 \frac{p(w, c)}{p(w)p_\alpha(c)}, 0)
\]

\[
P_\alpha(c) = \frac{\text{count}(c)^\alpha}{\sum_c \text{count}(c)^\alpha}
\]
Sparse versus Dense Vectors

- PPMI vectors are
  - **Long** (length $|V| = 20,000$ to $50,000$)
  - **Sparse** (most elements are zero)

- Alternative: learn vectors which are
  - **Short** (length 200-1000)
  - **Dense** (most elements are non-zero)
Why Dense Vectors

- Short vectors may be easier to use as features (less weights to tune)
- Dense vectors may generalize better than storing explicit counts
- They may do better at capturing synonymy
  - *Car and automobile are synonyms, but are represented as distinct dimensions; this fails to capture similarity between a word with car as a neighbor and a word with automobile as a neighbor.*

- In practice, they work better
Three Methods for Getting Short Dense Vectors

- Singular Value Decomposition (SVD)
  - A special case of this is called LSA – Latent Semantic Analysis

- Brown Clustering

- “Neural Language Model” – inspired predictive models
  - Skip-grams and CBOW
Dense Vectors via SVD

- Intuition
  - Approximate an N-dimensional dataset using fewer dimensions
  - By first rotating the axes into a new space
  - The highest order dimension captures the most variance in the original dataset
  - And the next dimension captures the next most variance, etc
  - Many such (related) methods:
    - PCA - principle components analysis
    - Factor analysis
    - SVD
Dimensionality reduction

PCA dimension 1

PCA dimension 2
Singular Value Decomposition (SVD)

\[ X \times W = \begin{bmatrix} S & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \mathbf{C} \times \mathbf{W} \]

- **Words** \( w \times c \)
- **Contexts** \( m \times m \)
- **S** \( m \times m \)
- **C** \( m \times c \)
Singular Value Decomposition (SVD)

\[ \mathbf{X} \mathbf{W} \mathbf{S} = \mathbf{W} \mathbf{S} \mathbf{C} \]

Words: rows corresponding to original but m columns represents a dimension in a new latent space, such that (1) m column vectors are orthogonal to each other, and (2) columns are ordered by the amount of variance in the dataset each new dimension accounts for.
Singular Value Decomposition (SVD)

\[ X = W S C \]

Words: \( w \times c \)

Contexts: \( m \times m \)

Words: \( w \times m \)

- \( S \): diagonal \( m \times m \) matrix of singular values expressing the importance of each dimension

S: diagonal \( m \times m \) matrix of singular values expressing the importance of each dimension
Singular Value Decomposition (SVD)

\[ X = W \times S \times C \]

- **Words** matrix: \( W \times C \)
- **Contexts** matrix: \( W \times m \)
- **S** matrix: \( m \times m \)
- **C** matrix: \( m \times c \)

**C:** columns corresponding to original but \( m \) rows corresponding to singular values.
SVD Applied to Term-Document Matrix: Latent Semantic Analysis

- If instead of keeping all \( m \) dimensions, we just keep the top \( k \) singular values. Let’s say 300.

- The result is a least-square approximation to the original \( X \)
- But instead of multiplying, we’ll just make use of \( W \)
Truncated SVD

\[ X = W S C \]

Contexts

Words

\[ W \times C \]

\[ W \times m \times k \]

\[ m \times k \times c \]
Truncated SVD Produces Embeddings

- Each row of $W$ is a $k$-dimensional representation of each word $w$
- $K$ might range from 50 to 100
- Generally we keep the top $k$ dimensions, but some experiments suggest that getting rid of the top 1 dimension or even the top 50 dimensions is helpful
Embeddings versus Sparse Vectors

- Dense SVD embeddings sometimes work better than sparse PPMI matrices at tasks like word similarity
  - Denoising: low-order dimensions may represent unimportant information
  - Truncation may help the models generalize better to unseen data
  - Having a smaller number of dimensions may make it easier for classifiers to properly weight the dimensions for the task
  - Dense models may do better at capturing higher order cooccurrence
Word Similarity

$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$
Word Embedding Representations

- Count-based
  - Tf-idf, PPMI
- Class-based
  - Brown Clusters
- Distributed prediction-based embeddings
  - Word2vec, Fasttext
- Distributed contextual (token) embeddings from language models
  - Elmo, BERT
- + many more variants
  - Multilingual embeddings, multi-sense embeddings, syntactic embeddings, etc …
The Brown Clustering Algorithm

- Input: a large collection of words
- Output 1: a partition of words into word clusters
- Output 2 (generalization of 1): a hierarchical word clustering
The Brown Clustering Algorithm

- An agglomerative clustering algorithm that clusters words based on which words precede or follow them
- These word clusters can be turned into a kind of vector
- We’ll give a very brief sketch here
Brown Clustering Algorithm

- Each word is initially assigned to its own cluster.
- We now consider merging each pair of clusters. Highest quality merge is chosen.
  - Quality = merges two words that have similar probabilities of preceding and following words
- Clustering proceeds until all words are in one big cluster
Brown Clusters as Vectors

- By tracing the order in which clusters are merged, the model builds a binary tree from bottom to top.
- Each word represented by binary string = path from root to leaf
- Each intermediate node is a cluster
- Chairman = 0010, “months” = 01, and verbs = 1
## Brown Clustering Example

A Sample Hierarchy (from Miller et al., NAACL 2004)

<table>
<thead>
<tr>
<th>Word</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>lawyer</td>
<td>1000001101000</td>
</tr>
<tr>
<td>newspaperman</td>
<td>100000110100100</td>
</tr>
<tr>
<td>stewardess</td>
<td>100000110100101</td>
</tr>
<tr>
<td>toxicologist</td>
<td>100000110100111</td>
</tr>
<tr>
<td>slang</td>
<td>100000110110</td>
</tr>
<tr>
<td>babysitter</td>
<td>10000011011100</td>
</tr>
<tr>
<td>conspirator</td>
<td>100000110111010</td>
</tr>
<tr>
<td>womanizer</td>
<td>100000110111011</td>
</tr>
<tr>
<td>mailman</td>
<td>1000001101111</td>
</tr>
<tr>
<td>salesman</td>
<td>10000011100000</td>
</tr>
<tr>
<td>bookkeeper</td>
<td>100000111000010</td>
</tr>
<tr>
<td>troubleshooter</td>
<td>100000111000110</td>
</tr>
<tr>
<td>bouncer</td>
<td>100000111001111</td>
</tr>
<tr>
<td>technician</td>
<td>100000111010000</td>
</tr>
<tr>
<td>janitor</td>
<td>100000111010101</td>
</tr>
<tr>
<td>saleswoman</td>
<td>100000111010110</td>
</tr>
<tr>
<td>Nike</td>
<td>1011011100100101011100</td>
</tr>
<tr>
<td>Maytag</td>
<td>10110111010010111010</td>
</tr>
<tr>
<td>Generali</td>
<td>10110111010111101111</td>
</tr>
<tr>
<td>Gap</td>
<td>101101110101111110</td>
</tr>
<tr>
<td>Harley-Davidson</td>
<td>1011011101011111101</td>
</tr>
<tr>
<td>Enfield</td>
<td>101101110101111111</td>
</tr>
<tr>
<td>genus</td>
<td>1011011101011111111</td>
</tr>
<tr>
<td>Microsoft</td>
<td>1011011101011111101</td>
</tr>
<tr>
<td>Ventrillex</td>
<td>101101110101100100</td>
</tr>
<tr>
<td>Tractebel</td>
<td>101101110101100111</td>
</tr>
<tr>
<td>Synopsys</td>
<td>101101110101100111</td>
</tr>
<tr>
<td>WordPerfect</td>
<td>1011011101011000100</td>
</tr>
<tr>
<td>John</td>
<td>101110010000000000</td>
</tr>
<tr>
<td>Consuelo</td>
<td>101110010000000010</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>101110010000000010</td>
</tr>
<tr>
<td>Kenneth</td>
<td>101110010000001100</td>
</tr>
<tr>
<td>Phillip</td>
<td>101110010000011010</td>
</tr>
<tr>
<td>WILLIAM</td>
<td>101110010000011110</td>
</tr>
<tr>
<td>Timothy</td>
<td>101110010000111110</td>
</tr>
<tr>
<td>Terrence</td>
<td>101110010000111110</td>
</tr>
</tbody>
</table>
Brown Clustering Example

from Brown et al., 1992
Intuition

Similar words appear in similar contexts

Similar words have similar distribution of words to their immediate left and right
Brown Clustering

- \( \mathcal{V} \) is a vocabulary
- \( C : \mathcal{V} \rightarrow \{1, 2, \ldots, k\} \) is a partition of the vocabulary into \( k \) clusters
- \( q(C(w_i)|C(w_{i-1})) \) is a probability of cluster \( w_i \) of to follow the cluster of \( w_{i-1} \)
- \( e(w_i|C(w_i)) = \frac{\text{count}(w_i)}{\sum_{x \in C(w_i)} \text{count}(x)} \)

\[
p(w_1, w_2, ..., w_T) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1}))
\]
Brown Clustering

- $\mathcal{V}$ is a vocabulary
- $C : \mathcal{V} \to \{1, 2, \ldots, k\}$ is a partition of the vocabulary into $k$ clusters
- $q(C(w_i)|C(w_{i-1}))$ is a probability of cluster $w_i$ of to follow the cluster of $w_{i-1}$
- $e(w_i|C(w_i)) = \frac{\text{count}(w_i)}{\sum_{x \in C(w_i)} \text{count}(x)}$

Quality($C$) = $\prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1}))$
An Example

\[ p(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1})) \]
An Example

\[
p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i | C(w_i))q(C(w_i) | C(w_{i-1}))
\]

\[
C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3
\]
An Example

\[ p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i|C(w_i)) q(C(w_i)|C(w_{i-1})) \]

\[ C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3 \]

\[ e(\text{the}|1) = 1, \quad e(\text{cat}|2) = e(\text{dog}|2) = 0.5, \quad e(\text{saw}|3) = 1 \]
An Example

\[ p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1})) \]

\[ C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3 \]
\[ e(\text{the}|1) = 1, \quad e(\text{cat}|2) = e(\text{dog}|2) = 0.5, \quad e(\text{saw}|3) = 1 \]
\[ q(1|0) = 0.2, \quad q(2|1) = 0.4, \quad q(3|2) = 0.3, \quad q(1|3) = 0.6 \]
An Example

\[ p(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1})) \]

\[ C(\text{the}) = 1, \quad C(\text{dog}) = C(\text{cat}) = 2, \quad C(\text{saw}) = 3 \]

\[ e(\text{the}|1) = 1, \quad e(\text{cat}|2) = e(\text{dog}|2) = 0.5, \quad e(\text{saw}|3) = 1 \]

\[ q(1|0) = 0.2, \quad q(2|1) = 0.4, \quad q(3|2) = 0.3, \quad q(1|3) = 0.6 \]

\[ p(\text{the dog saw the cat}) = \]
The Brown Clustering Model

- $\mathcal{V}$ is a vocabulary
- $C : \mathcal{V} \rightarrow \{1, 2, \ldots k\}$ is a partition of the vocabulary into $k$ clusters
- $q(C(w_i)|C(w_{i-1}))$ is a probability of cluster $w_i$ to follow the cluster of $w_{i-1}$
- $e(w_i|C(w_i)) = \frac{\text{count}(w_i)}{\sum_{x \in C(w_i)} \text{count}(x)}$

Quality($C$) = $\prod_{i=1}^{n} e(w_i|C(w_i))q(C(w_i)|C(w_{i-1}))$
How to Measure the Quality of C?

How do we measure the quality of a partition C?

\[
\text{Quality}(C') = \sum_{i=1}^{n} \log e(w_i|C(w_i))q(C(w_i)|C(w_{i-1}))
\]

\[
= \sum_{c=1}^{k} \sum_{c' = 1}^{k} p(c, c') \log \frac{p(c, c')}{p(c)p(c')} + G
\]

Where \( p(c, c') = \frac{n(c, c')}{\sum_{c,c'} n(c, c')} \quad p(c) = \frac{n(c)}{\sum_c n(c)} \)

Here, \( n(c) \) is the number of times class c occurs in the corpus, \( n(c, c') \) is the number of times c’ is seen following c, under the function C

A First Algorithm

- Start with $|V|$ clusters: each word gets its own cluster
- The goal is to get $k$ clusters
- We run $|V| - k$ merge steps:
  - Pick 2 clusters and merge them
  - Each step picks the merge maximizing $\text{Quality}(C)$
- Cost?
  - $O(|V| - k) \times O(|V|^2) \times O(|V|^2) = O(|V|^5)$

# iters  # pairs  compute $\text{Quality}(C)$
A Second Algorithm

- m: a hyper-parameter, sort words by frequency
- Take the top m most frequent words, put each of them in its own cluster $c_1, c_2, c_3, \ldots c_m$
- For $i = (m + 1) \ldots |V|$
  - Create a new cluster $c_{m+1}$ (we have $m + 1$ clusters)
  - Choose two clusters from $m + 1$ clusters based on quality(C) and merge (back to $m$ clusters)
- Carry out $m - 1$ final merges (full hierarchy)
- Running time $O(|V|m^2 + n)$, $n=$#words in corpus
Next Class

- Word2vec, FastText
- Elmo, BERT, XLNet
- Multilingual Embeddings
Additional Notes On Brown Clustering
How to Measure the Quality of C?

- $n(w)$ be the number of times word $w$ appears in the text.
- $n(w, w')$ be the number of times the bigram $(w, w')$ occurs in the text.
- $n(c) = \sum_{w \in c} n(w)$ be the number of times a word in a cluster $c$ appears in the text.
- $n(c, c') = \sum_{w \in c, w' \in c'} n(w, w')$
- $n$ is simply the length of the text.
How to Measure the Quality of C?

\[
\text{Quality}(C) = \frac{1}{n} \sum_{i=1}^{n} \log P(C(w_i)|C(w_{i-1}))P(w_i|C(w_i))
\]

\[
= \sum_{w,w'} \frac{n(w, w')}{n} \log P(C(w')|C(w))P(w'|C(w'))
\]

\[
= \sum_{w,w'} \frac{n(w, w')}{n} \log \frac{n(C(w), C(w'))}{n(C(w))n(C(w'))} \frac{n(w')}{n(C(w'))}
\]

\[
= \sum_{w,w'} \frac{n(w, w')}{n} \log \frac{n(C(w), C(w'))}{n(C(w))n(C(w'))} + \sum_{w,w'} \frac{n(w, w')}{n} \log \frac{n(w')}{n}
\]

\[
= \sum_{c,c'} \frac{n(c, c')}{n} \log \frac{n(c, c')n}{n(c)n(c')} + \sum_{w'} \frac{n(w')}{n} \log \frac{n(w')}{n}
\]
How to Measure the Quality of C?

Define

\[ P(w) = \frac{n(w)}{n} \quad P(c) = \frac{n(c)}{n} \quad P(c, c') = \frac{n(c, c')}{n} \]

\[
\text{Quality}(C) = \sum_{c, c'} P(c, c') \log \frac{P(c, c')}{P(c)P(c')} + \sum_w P(w) \log P(w)
\]

\[ = I(C) - H \]

mutual information between adjacent clusters

entropy of the word distribution