Outline

- Word similarity
- Thesaurus methods (path length, information content)
- Distributional methods

Reading

- Reading:
  Chapter 20 of Jurafsky and Martin (2nd ed)
  Sections 20.6 and 20.7

Word Similarity

- Several techniques have been developed for determining the how close two words are in meaning, referred to as the word similarity or semantic distance
  - For example, car and automobile are similar while car and printer are not
- Word similarity has a role in many applications:
  - Information Retrieval and Question answering: retrieve documents with similar meanings to query words
  - Summarization, generation and machine translation: similar words could potentially be substituted for one another
  - language modelling cluster similar words in class-based models
  - Automatic essay grading determine whether essay is similar to correct answer

Word Similarity - Main Approaches

Two main approaches to computing word similarity:

- Thesaurus-based which rely on a dictionary, like WordNet
- Distributional which compare the occurrences of words in a corpus

Word Similarity – Thesaurus Methods

- The simplest thesaurus-based algorithms are based on the idea that similarity can be computed by counting the number of links between words or senses in the thesaurus; the closer two words/senses are together the higher their similarity.
- This can be defined as follows:

$$\mathit{pathlen}(c_1, c_2) = \text{number of edges between nodes } c_1 \text{ and } c_2$$

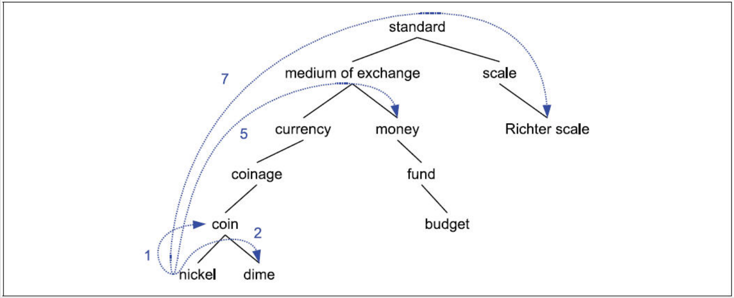
  - ◇ Note that this measure assigns low scores to similar pairs of words (since the path is short) and high scores to pairs that are not similar

- Often converted into a similarity measures as follows:

$$\mathit{sim}(c_1, c_2) = \frac{1}{\mathit{pathlen}(c_1, c_2)}$$

Thesaurus Methods – Example

Fragment of WordNet hypernym hierarchy



word 1	word 2	path length	sim
nickel	coin	1	1
nickel	dime	2	0.5
nickel	money	5	0.2
nickel	Richter scale	7	0.14

Word Similarity vs Sense Similarity

- In reality similarity applies to word senses rather than words. For example, bank^2 (= 'edge of river') is similar to slope but bank^1 (= financial institution) is not

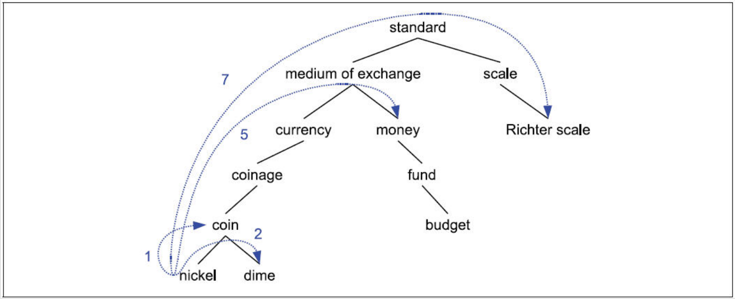
- For most applications being able to identify the similarity between pairs of words is more useful than being able to do so for senses

- We can do this by computing the similarity for all possible pairs of senses for two words and choosing the highest score:

$$wordsim(w_1, w_2) = \underset{c_1 \in senses(w_1), c_2 \in senses(w_2)}{\arg\max} sim(c_1, c_2)$$

Word Similarity – Problems with Path length

- The path length measure is simple to understand and normally easy to compute
- It assumes that each link in the hierarchy represents the same unit of distance which may not be the case, eg in WordNet
  - Similarity between nickel and coin seems higher than medium of exchange and standard

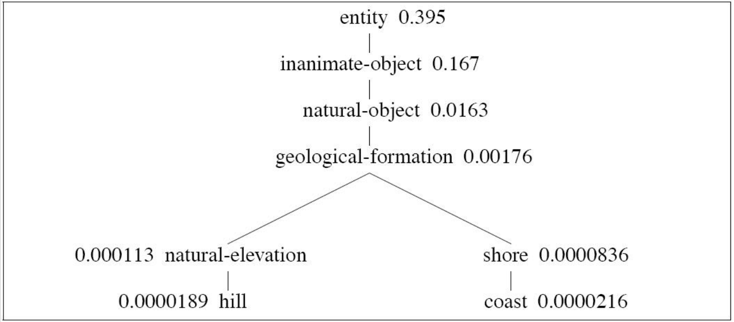


Information-Content Word-similarity

- Another set of thesaurus-based algorithms attempts to avoid this problem
- Information-content approaches add corpus probabilities to the thesaurus
- $P(c)$ is the probability that a word is an example of a concept.
- Words in a corpus are considered instances of any concepts that subsume them
  - dime counts as an occurrence of the concepts dime, coin, coinage, ...
- $P(c)$ is computed as $P(c) = \frac{\sum_{w \in words(c)} count(w)}{N}$, where $words(c)$ is the set of words subsumed by a concept and N is the number of words in the corpus

Example

- Fragment of WordNet hierarchy with probabilities attached to each concept
- More general concepts (e.g. entity) have higher probabilities than more specific ones (e.g. hill)



Information-Content Word-similarity

- Two additional definitions are required:
  - The Information Content (IC) of a concept is defined as $IC(c) = -\log(P(c))$.
    - This is a standard definition from Information Theory
    - High probabilities lead to low Information Content (and vice versa). The intuition here is that seeing something rare provides more information that seeing something common
  - The lowest common subsumer (LCS) of two nodes in a hierarchy, c_1 and c_2 is the lowest node that subsumes both c_1 and c_2.
    - The LCS is the lowest shared hypernym of c_1 and c_2

Information-Content Measures

- Several similarity measures that used these concepts have been proposed:
  - Resnik $sim_{resnik}(c_1, c_2) = IC(LCS(c_1, c_2))$
  - Lin $sim_{lin}(c_1, c_2) = \frac{2 \times IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)}$
  - Jiang and Conrath
    $dist_{JC} = IC(c_1) + IC(c_2) - 2 \times IC(LCS(c_1, c_2))$
- Note that the Jiang and Conrath measure is a distance measure rather than a similarity measure (common trick: $sim_{JC} = \frac{1}{dist_{JC}}$)
- It has been shown to work as well as or better than other thesaurus based measures

Information-Content Measures

- Note that Jurafsky and Martin express these in a slightly different (but equivalent) way:
- Resnik $sim_{resnik}(c_1, c_2) = -\log P(LCS(c_1, c_2))$
- Lin $sim_{lin}(c_1, c_2) = \frac{2 \times \log P(LCS(c_1, c_2)}{\log P(c_1) + \log P(c_2)}$
- Jiang and Conrath
  $dist_{JC} = 2 \times \log P(LCS(c_1, c_2) - (\log P(c_1) + \log P(c_2))$

Distributional Similarity Methods

- Distributional similarity methods use the context in which words appear to compute similarity between them
- Have advantage of not requiring a thesaurus or dictionary which may not be available for all languages and/or domains
- "You shall know a word by the company it keeps" (Firth, 1957)
- We can learn a lot about the meaning of words from their context:

  A bottle of tezguino is on the table
  Everybody likes tezguino
  tezguino makes you drunk
  We make tezguino out of corn

- Suggests tezguino is an alcoholic drink made out of corn and is similar to words like beer, liquor and tequila.

Distributional Similarity - Basic approach

- The basic approach behind approaches to distributional similarity is to represent each word as a feature vector (similar to the approach used in supervised approaches to WSD).
- Estimate similarity between words by applying a vector distance measure
- Simple example feature vectors for apricot, pineapple, digital and information

	arts	boil	data	function	large	sugar	summarized	water
apricot	0	1	0	0	1	1	0	1
pineapple	0	1	0	0	1	1	0	1
digital	0	0	1	1	1	0	1	0
information	0	0	1	1	1	0	1	0

  - ◇ In real life context words are unlikely to be as good discriminators and vectors would be much more sparse

Defining Feature Vectors

- Feature vectors for distributional similarity can be constructed from:
- Bag of words The vector is constructed from the words which appear in some predefined context around w (contexts vary from $\pm$ 1 to $\pm$ 500).
- Grammatical relations or dependencies Vector constructed from words in a grammatical relation with w and note the relation.
  - Grammatical relations can be extracted from the output of a parser, e.g. noun-subject, noun-object, noun-adjective
  - Example Everybody likes tezguino → likes (subject everybody), likes (object tezguino), everybody (subject-of likes), tezguino (object-of likes)
- Vectors created using grammatical relations will be much more sparse than those using bag of words
- Compare with supervised WSD where feature vectors can be constructed from co-occurrence and collocational features

Weighting vectors

- Choice of approaches to weighting vectors
- Binary values and raw counts used in examples so far
- Another approach is to use probability that feature occurs with target word
- For a word w each element of its concurrence vector is a feature f consisting of a relation r and a related word w' (so $f = (r, w')$).
- The probability of a feature given a target word, $P(f|w)$, can be estimated as $\frac{count(f,w)}{count(w)}$
- The probability can be used as a measure of association and is defined as follows $assoc_{prob}(w, f) = P(f|w)$
- Turns out that these approaches don't work very well
- Define some more more probabilities, $P(w, f) = \frac{count(f,w)}{\sum_{w'} count(w')}$

Weighting vectors (cont.)

- Some better approaches to weighting vectors (see Jurafsky and Martin for motivations behind each):
  - $assoc_{PMI} = \log_2 \frac{P(w,f)}{P(w)P(f)}$
  - $assoc_{Lin} = \log_2 \frac{P(w,f)}{P(w)P(r|w)P(w'|w)}$
  - $assoc_{t-test} = \frac{P(w,f)-P(w)P(f)}{\sqrt{P(f)P(w)}}$
- Example of object of verb drink with weights assigned by $assoc_{PMI}$

Object	**Count**	**PMI Assoc**	**Object**	**Count**	**PMI Assoc**
bunch beer	2	12.34	wine	2	9.34
tea	2	11.75	water	7	7.65
Pepsi	2	11.75	anything	3	5.15
champagne	4	11.75	much	3	5.15
liquid	2	10.53	it	3	1.25
beer	5	10.20	<SOME AMOUNT>	2	1.22

Comparing vectors

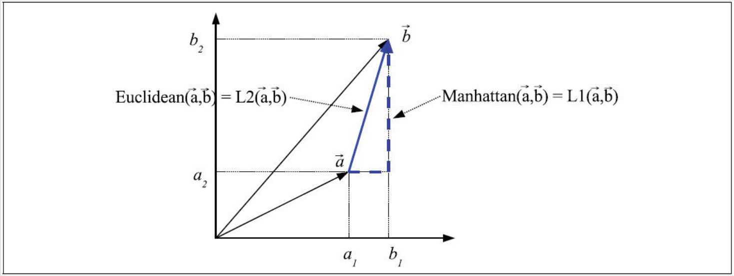
- Standard approach to comparing vectors are the Manhattan and Euclidian distance
- Manhattan distance (Levenstein distance or L1 norm)

$$distance_{manhattan(\vec{x},\vec{y})} = \sum_{i=i}^{N} |x_i - y_i|$$

- Euclidian distance (L2 norm)

$$distance_{euclidian(\vec{x},\vec{y})} = \sum_{i=i}^{N} \sqrt{(x_i - y_i)^2}$$

Comparing vectors



- These measures are sensitive to extreme values and are not suitable for word similarity so measures from Information Retrieval and Information Theory are preferred

Comparing vectors (cont.)

- Cosine metric Normalised dot product of two vectors, equivalent to the cosine of the angle between them

$$sim_{cosine(\vec{v},\vec{w})} = \frac{\vec{v}.\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

- Jaccard measure For each element divide the smaller value by the larger

$$sim_{Jaccard(\vec{v},\vec{w})} = \frac{\sum_{i=1}^{N} min(v_i, w_i)}{\sum_{i=1}^{N} max(v_i, w_i)}$$

- Dice measure Similar to Jaccard measure but divide by the sum of the two elements

$$sim_{Dice(\vec{v},\vec{w})} = \frac{2 \times \sum_{i=1}^{N} min(v_i, w_i)}{\sum_{i=1}^{N}(v_i + w_i)}$$

Summary of Association and Similarity Measures for Distributional Similarity

See Jurafsky and Martin for more details on these measures and some others

$$\text{assoc}_{\text{prob}}(w,f) = P(f|w) \tag{20.35}$$

$$\text{assoc}_{\text{PMI}}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(f)} \tag{20.38}$$

$$\text{assoc}_{\text{Lin}}(w,f) = \log_2 \frac{P(w,f)}{P(w)P(r|w)P(w'|w)} \tag{20.39}$$

$$\text{assoc}_{\text{t-test}}(w,f) = \frac{P(w,f)-P(w)P(f)}{\sqrt{P(f)P(w)}} \tag{20.41}$$

$$\text{sim}_{\text{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \tag{20.47}$$

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)} \tag{20.48}$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} (v_i+w_i)} \tag{20.49}$$

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2}) \tag{20.52}$$

Example of distributional word similarity output

- hope (N) optimism 0.141, chance 0.137, expectation 0.137, prospect 0.126, dream 0.119, desire 0.118, fear 0.116, effort 0.111, confidence 0.109, promise 0.108
- hope (V) would like 0.158, wish 0.140, plan 0.139, say 0.137, believe 0.135, think 0.133, agree 0.130, wonder 0.130, try 0.127, decide 0.125
- brief (N) legal brief 0.139, affidavit 0.103, filing 0.0983, petition 0.865, document 0.0835, argument 0.0832, letter 0.0786, rebuttal 0.0778. memo 0.0768, article 0.0758
- brief (V) lengthy 0.256, hour-long 0.191, short 0.174, extended 0.163, frequent 0.163, recent 0.158, short-lived 0.155, prolonged 0.149, week-long 0.149, occasional 0.146

(Generated using approach from Lin (2007))

Lexical Similarity Summary

- Lexical similarity methods automatically estimate the similarity between pairs of words
- Two main approaches: thesaurus or dictionary based and distributional
- Thesaurus based methods:
  - Path length
  - Information-content based
- Distributional methods:
  - Represent word as feature vector
  - Compare feature vectors