CS626: Speech, NLP and the Web

Expectation Maximization, WSD

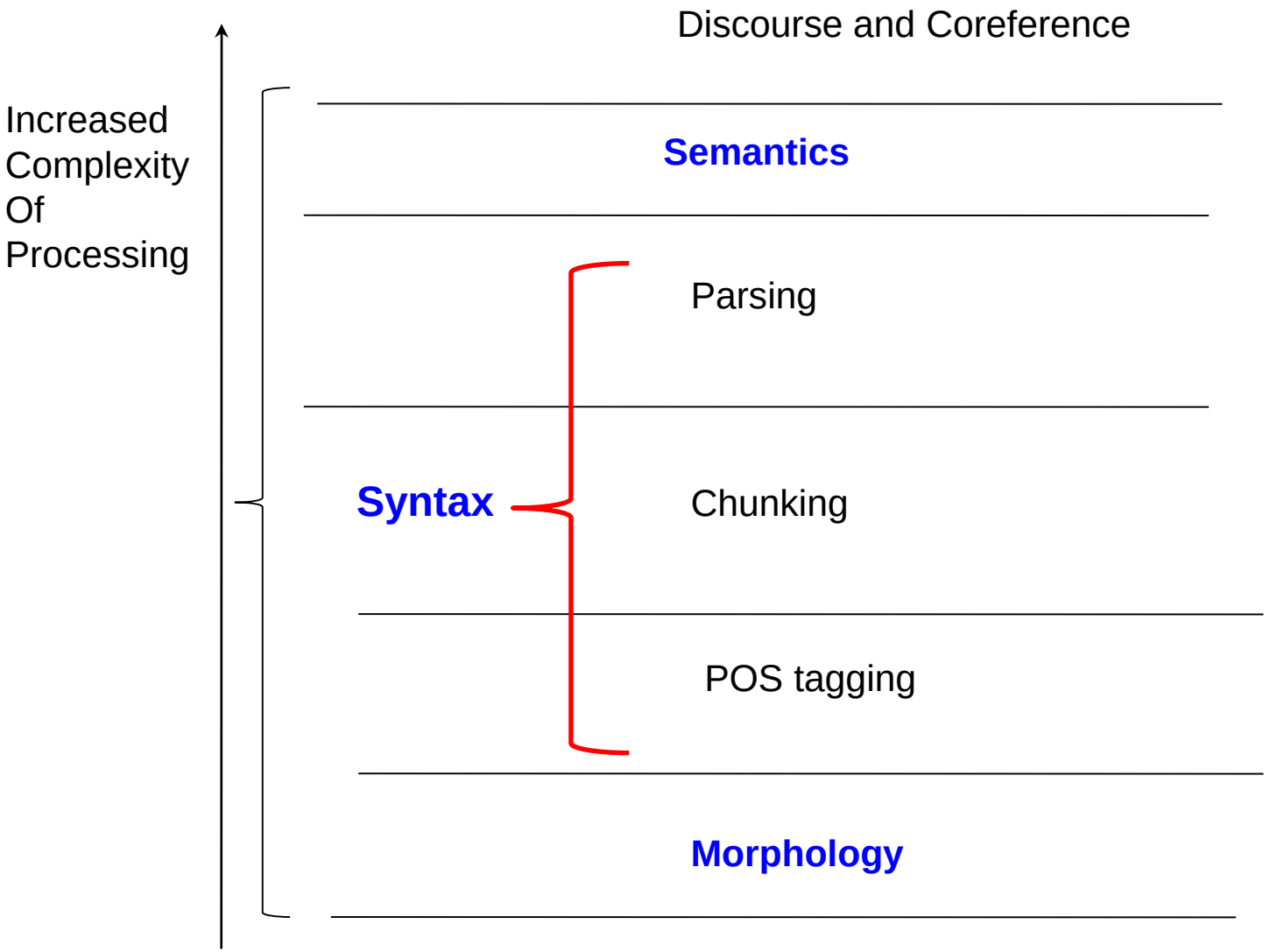
Pushpak Bhattacharyya

Computer Science and Engineering Department

IIT Bombay

Week 11 of 3rd October, 2022

NLP Layer and Linguistics



Expectation Maximization

- A very important technique for parameter estimation in presence of hidden variables
- Application in
  - Machine Translation- word alignment
  - HMM- combined transition and emission probabilities
  - PCFG- probabilities of CFG rules

Mathematics of EM

From

Pushpak Bhattacharyya, *Machine Translation*, CRC Press, 2015

Maximum Likelihood of Observations

- ***Situation 1: Throw of a Single Coin***
- The parameter is the probability *p* of getting heads in a single toss. Let *N* be the number of tosses. Then the observation *X* and the data or observation likelihood *D* respectively are:

$$X :< x_1, x_2, x_3, ..., x_{N-1}, x_N >$$

$$D = \prod_{i=1}^{N} p^{x_i} (1-p)^{1-x_i}, \text{ s.t. } x_i = 1 \text{ or } 0, \text{ and } 0 \le p \le 1$$

where x_i is an indicator variable assuming values 1 or 0 depending on the *ith* observation being heads or tail. Since there are *N* identically and independently distributed (*i.i.d.*) observations, *D* is the product of probabilities of individual observations each of which is a Bernoulli trial.

Single coin

Since exponents are difficult to manipulate mathematically, we take log of *D*, also called log likelihood of data, and maximize with regard to *p*. This yields

$$p = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{M}{N}; \; M = \#Heads, N = \#tosses$$

Throw of 2 coins

- Three parameters: probabilities p_1 and p_2 of heads of the two coins and the probability p of choosing the first coin (automatically, $1-p$ is the probability of choosing the second coin).
- N tosses and observations of heads and tails. Only, we do not know which observation comes from which coin.
- Indicator variable z_i is introduced to capture coin choice ($z_i=1$ if coin 1 is chosen, else 0). This variable is hidden, *i.e.*, we do not know its values.
- However, without it the likelihood expression would have been very cumbersome.

Data Likelihood

Data Likelihood,

$D = P_{<p1,p2,p>}(X) = P_\theta(X),\ \theta = <p, p_1, p_2>$

$= \Sigma_Z P_\theta(X,Z))$

$$X :< x_1, x_2, x_3, ..., x_{N-1}, x_N >$$

$$Z :< z_1, z_2, z_3, ..., z_{N-1}, z_N >$$

$$P_\theta(X,Z) = \prod_{i=1}^{N} \left[\left(p p_1^{\,x_i} (1-p_1)^{1-x_i} \right)^{z_i} \left((1-p) p_2^{\,x_i} (1-p_2)^{1-x_i} \right)^{1-z_i} \right],$$

$$\text{s.t. } z_i, x_i = 1 \text{ or } 0, \text{ and } 0 \le p, p_1, p_2 \le 1$$

Invoke Jensen Inequality

We would like to work with $logP_{\theta}(X)$. However, there will be a Σ inside *log*. Fortunately, *log* is a concave function, so that

$$\log\left(\sum_{i=1}^{K} \lambda_i y_i\right) \geq \left(\sum_{i=1}^{K} \lambda_i \log(y_i)\right); \sum_{i=1}^{K} \lambda_i = 1$$

Log likelihood of Data

$LL(D)$= log likelihood of data

$= log(P_\theta(X))= log(\Sigma_Z P_\theta(X,Z))$

$= log[\Sigma_Z \lambda_Z (P_\theta(X,Z)/\lambda_Z)]; \Sigma_Z \lambda_Z = 1$

$>= \Sigma_Z [\lambda_Z log[(P_\theta(X,Z)/\lambda_Z)]$

After a number of intricate mathematical steps

$LL(D) >= E_{Z|X,\theta} log(P_\theta(X,Z))$, where $E(.)$ is the expectation function; note that the expectation is conditional on X.

Expectation of log likelihood

$$E_{Z|X}[\log(P_\theta(X,Z)]$$

$$= E_{Z|X}\left[\log\prod_{i=1}^{N}\left[\left(pp_1{}^{x_i}\left(1-p_1\right)^{1-x_i}\right)^{z_i}\left((1-p)\,p_2{}^{x_i}\left(1-p_2\right)^{1-x_i}\right)^{1-z_i}\right]\right]$$

$$= E_{Z|X}\left[\sum_{i=}^{N} z_i\left(\log p + x_i\log p_1 + (1-x_i)\log\left(1-p_1\right)\right) + (1-z_i)\left(\log(1-p) + x_i\log p_2 + (1-x_i)\log\left(1-p_2\right)\right)\right]$$

$$= \sum_{i=1}^{N}\left[E(z_i\mid x_i)\left(\log p + x_i\log p_1 + (1-x_i)\log\left(1-p_1\right)\right) + (1-E(z_i\mid x_i))\left(\log(1-p) + x_i\log p_2 + (1-x_i)\log\left(1-p_2\right)\right)\right]$$

$$\text{s.t. } z_i, x_i = 1 \text{ or } 0, \text{ and } 0 \le p, p_1, p_2 \le 1$$

Derivation of E and M steps for 2 coin problem (1/2)- M step

Take partial derivative of $E_{Z|X,\theta}(.)$ (prev. slide) wrt p, p_1, p_2 and equate to 0.

$$p = \frac{\sum_{i=1}^{N} E(z_i \mid x_i)}{N}$$

$$p_1 = \frac{\sum_{i=1}^{N} E(z_i \mid x_i) x_i}{\sum_{i=1}^{N} E(z_i \mid x_i)}$$

$$p_2 = \frac{M - \sum_{i=1}^{N} E(z_i \mid x_i) x_i}{N - \sum_{i=1}^{N} E(z_i \mid x_i)}; M = \#Heads, N = \#tosses$$

Derivation of E and M steps for 2 coin problem (2/2)- E step

$E(z_i|x_i) = 1.P(z_i=1|x_i) + 0.P(z_i=0|x_i)$

$= P(z_i=1|x_i)$

$$P(z_i = 1 | x_i) = \frac{P(z_i = 1, x_i)}{P(x_i)}$$

$$= \frac{pp_1^{x_i}(1-p_1)^{1-x_i}}{P(x_i, z_i = 1) + P(x_i, z_i = 0)}$$

$$= \frac{pp_1^{x_i}(1-p_1)^{1-x_i}}{pp_1^{x_i}(1-p_1)^{1-x_i} + (1-p)p_2^{x_i}(1-p_2)^{1-x_i}}$$

Generalization into N “throws” using M “things” each having L outcomes

From

Pushpak Bhattacharyya, *Machine Translation*, CRC Press, 2015

Multiple outcomes from multiple entities

- “Throw” of “something” where that something has more than 2 outcomes, e.g., throw of multiple dice
- The observation sequence has a sequence of 1 to 6s
- But we do not know which observation came from which dice
- Gives rise to a multinomial that is extremely useful in NLP ML.

Observation Sequence

- N ‘throws’, 1 of L outcomes from each throw, 1 of the M ‘things’ (called ‘sources’) chosen
- $\Sigma_{k=1,L} x_{ik}=1$, since each x_{ik} is either 1 or 0 and one and only one of them is1.
- D (data):

$$<x_{11}/x_{12}/\ldots x_{1L}>, <x_{21}/x_{22}/\ldots x_{2L}>, \ldots$$
$$<x_{N1}/x_{N2}/\ldots x_{NL}>$$

Hidden Variable

- Hidden variable for M sources
- $\Sigma_{j=1,M} z_{ij} = 1$, since each z_{ij} is either 1 or 0 and one and only one of them is 1.
- Z:

$$<z_{11}/z_{12}/\ldots z_{1M}>, <z_{21}/z_{22}/\ldots z_{2M}>, \ldots$$
$$<z_{N1}/z_{N2}/\ldots z_{NM}>$$

Parameters

- Parameter set θ:
  - π_j: probability of choosing source j
  - p_{jk}: probability of observing k^{th} outcome from the j^{th} source

This will be elaborated next week; only expressions are given now

M-step

M-Step:

$$\pi_j = \frac{\sum_{i=1}^{N} E(z_{ij})}{\sum_{j=1}^{M} \sum_{i=1}^{N} E(z_{ij})}$$

$$p_{jk} = \frac{\sum_{i=1}^{N} E(z_{ij}) x_{ik}}{\sum_{i=1}^{N} E(z_{ij})}$$

E-step

E-Step:

$$E(z_{ij}) = \frac{\pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})}{\sum_{j=1}^{M} \pi_j \prod_{k=1}^{L} (p_{jk}^{x_{ik}})}$$

Word Sense Disambiguation

OVERLAP BASED APPROACHES

- Require a ***Machine Readable Dictionary*** *(MRD).*
- Find the overlap between the features of different senses of an ambiguous word (sense bag) and the features of the words in its context (context bag).
- These features could be sense definitions, example sentences, hypernyms etc.
- The features could also be given weights.
- The sense which has the maximum overlap is selected as the contextually appropriate sense.

LESK'S ALGORITHM

Sense Bag: *contains the words in the definition of a candidate sense of the ambiguous word.*

Context Bag: *contains the words in the context.*

E.g. "On burning ***coal*** we get ***ash***."

From Wordnet

- The noun ash has 3 senses (first 2 from tagged texts)
- 1. (2) ash -- (the residue that remains when something is burned)
- 2. (1) ash, ash tree -- (any of various deciduous pinnate-leaved ornamental or timber trees of the genus Fraxinus)
- 3. ash -- (strong elastic wood of any of various ash trees; used for furniture and tool handles and sporting goods such as baseball bats)
- The verb ash has 1 sense (no senses from tagged texts)
- 1. ash -- (convert into ashes)

LESK'S ALGORITHM (contd..)

- Note the **importance of lower layer tasks** in NLP stack for a higher layer task like **Word Sense Disambiguation**
  - **Morphological Analysis**: Comparing the root words while finding overlap could be useful
  - Ex: 'burned' and 'burning' have the same root word in the previous example
  - **POS Tagging:** Identifying the POS tag of a word would reduce the search space while finding its sense
  - Ex: Finding out POS of 'ash' as noun reduces the

CRITIQUE

- Many times there may not be any overlap: sparsity problem
  - The ash from the combustion
- Overlap may be spurious leading to “drift”
  - *The ash tree was burned*
- Proper nouns as as strong disambiguators, but not present in WN

  E.g. **“Sachin Tendulkar”** will be a strong indicator of the category **“sports”.**

  Sachin Tendulkar plays **cricket.**

- Typical Accuracy
  - 50% when tested on 10 highly polysemous English words.

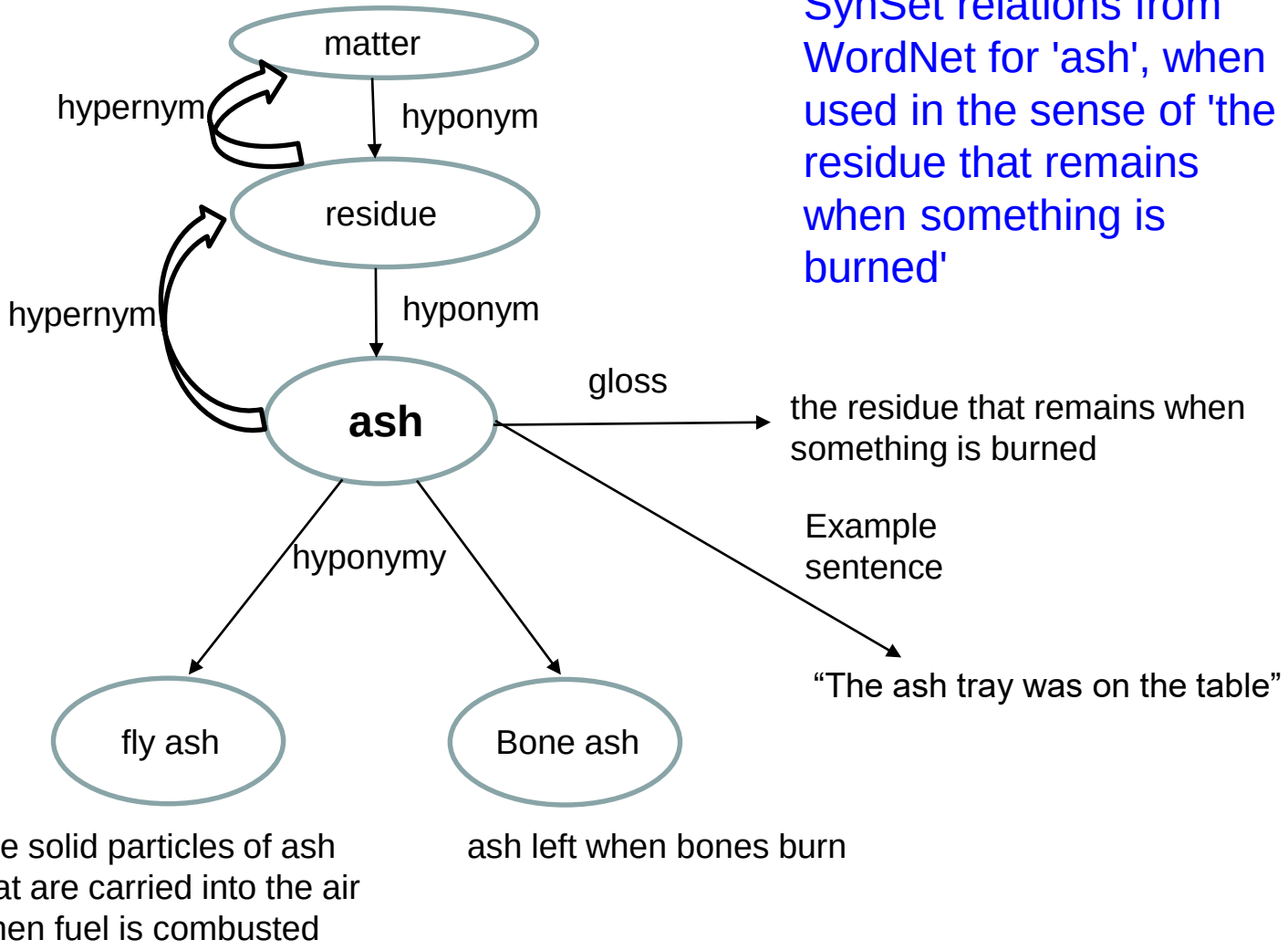
Extended Lesk's algorithm

- Extension includes glosses of semantically related senses from WordNet (e.g. *hypernyms*, *hyponyms*, etc.).

- The scoring function now computes the overlap of context bag with not only the words local to the synset but also words occurring in neighjboring synsets

- Vide next slide

WordNet Sub-graph



SynSet relations from WordNet for 'ash', when used in the sense of 'the residue that remains when something is burned'

Example: Extended Lesk

- *"On combustion of coal we get ash"*

From Wordnet

- The noun ash has 3 senses (first 2 from tagged texts)
- 1. (2) ash -- (the residue that remains when something is burned)
- 2. (1) ash, ash tree -- (any of various deciduous pinnate-leaved ornamental or timber trees of the genus Fraxinus)
- 3. ash -- (strong elastic wood of any of various ash trees; used for furniture and tool handles and sporting goods such as baseball bats)
- The verb ash has 1 sense (no senses from tagged texts)
- 1. ash -- (convert into ashes)

Example: Extended Lesk (cntd)

- *"On combustion of coal we get ash"*

From Wordnet (through hyponymy)

- ash -- (the residue that remains when something is burned)

  => fly ash -- (fine solid particles of ash that are carried into the air when fuel is combusted)

  => bone ash -- (ash left when bones burn; high in calcium phosphate; used as fertilizer and in bone china)

Critique of Extended Lesk

- Larger region of matching in WordNet
  - Increased chance of Matching

    BUT
  - Increased chance of Topic Drift

  - E.g. for “there were some bones under the ash tree”→ Spurious overlap with bone under “bone ash”

What if overlaps tie?

- There is “tree” also in the context
- Both “bone” and “tree” will contribute equally to overlap
- Then we will invoke other factors like PROXIMITY which is also called SANNIDHI in Indian linguistic tradition (SANNIDHI means “proximity”)
- AKANGJSHA (desire), YOGYATA (suitability) and SANNIDHI (proximity) are fundamental disambiguators
- Since “tree” is *CLOSER* to “ash”, ash tree will be the winner sense

Argument Frame Selection Preference

- “eat” and “rice”
- Eat needs an object→ akangksha (argument)
- Object should be edible, rice is edible→ yogyata (selectional preference)

WSD using Sense Embedding

- We will create the **sense embedding** by averaging the word vector for each word in the Gloss.

  E.g. “On burning *coal* we get ***ash***.”

- We have three senses from Wordnet

> **1.** ash -- (the residue that remains when something is burned)
>
> **2.** ash, ash tree -- (any of various deciduous pinnate-leaved ornamental or timber trees of the genus Fraxinus)
>
> **3.** ash -- (strong elastic wood of any of various ash trees; used for furniture and tool handles and sporting goods such as baseball bats)

- sense_emb = sum of word vector of each word in Gloss /# of words in Gloss
- context_emb = sum of word vector of each word in input /# of words in input

WSD using Sense Embedding (cont'd…)

- sense_emb = sum of word vector of each word in Gloss /# of words in Gloss
- context_emb = sum of word vector of each word in input /# of words in input
- Compute the cosine similarity between each sense embedding and context embedding:

  *similarity_with_sense_1 = cosine_similarity(sense_emb_1, context_emb)=**0.4675***

  similarity_with_sense_2 = cosine_similarity(sense_emb_2, context_emb) =0.4315

  similarity_with_sense_3 = cosine_similarity(sense_emb_3, context_emb)=0.4019
- The sense having the maximum cosine similarity will be the disambiguated sense for the given context word.

$$best\_sense = argmax\ (\ similarity\_with\_sense\_i\ )\ \ \forall\, i$$

Best sense: ash -- (the residue that remains when something is burned)

WALKER'S ALGORITHM

- A Thesaurus Based approach.
- ***Step 1:*** *For each sense of the target word find the thesaurus category to which that sense belongs.*
- ***Step 2:*** *Calculate the score for each sense by using the context words. A context word will add 1 to the score of the sense if the thesaurus category of the word matches that of the sense.*

  - *E.g. The money in this **bank** fetches an interest of 8% per annum*
  - Target word: ***bank***
  - Clue words from the context: ***money, interest, annum, fetch***

	Sense1: Finance	Sense2: Location
Money	+1	0
Interest	+1	0
Fetch	0	0
Annum	+1	0
Total	3	0

Context words add 1 to the sense when the topic of the word matches that of the sense

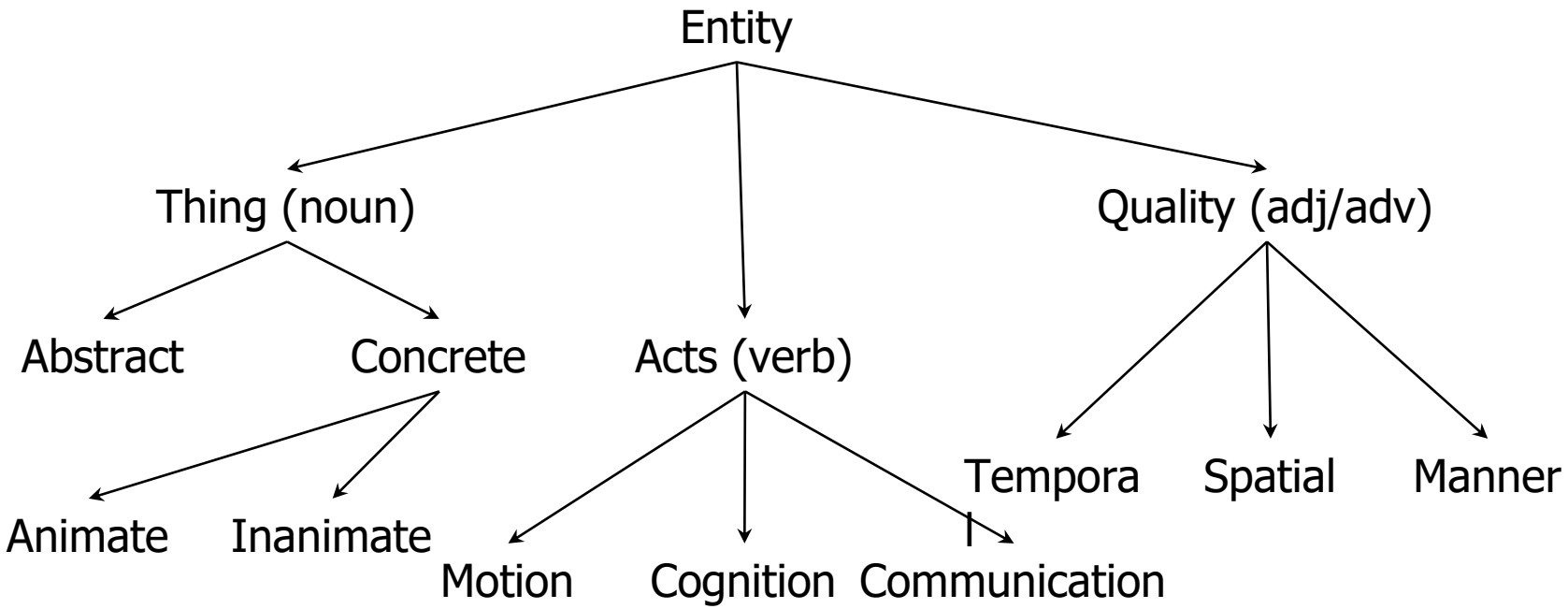
Walker Algo cntd.

- Thesaurus is a systematic organization of concepts
- “bank”, “interest”, “annum” etc. appear in the finance domain and contribute to each others count in the walker algo
- Lesk insists on local exact symbol match
- Extended lesk on inside and outside synset matches
- Walker insists on domain (concept category) matching

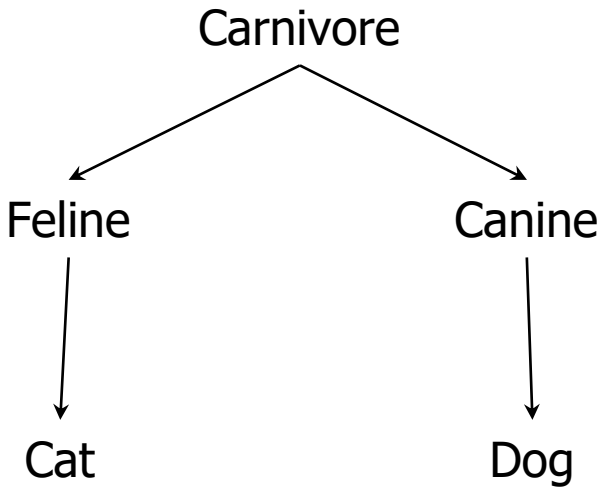
WSD USING CONCEPTUAL DENSITY *(Agirre and Rigau, 1996)*

- Select a sense based on the *relatedness* of that word-sense to the context.
- Relatedness is measured in terms of conceptual distance
  - (i.e. how close the concept represented by the ***word*** and the concept represented by its ***context words*** are)
- This approach uses a structured hierarchical semantic net (*WordNet*) for finding the conceptual distance.
- Smaller the conceptual distance higher will be the conceptual density.
  - (i.e. if all words in the context are strong indicators of a particular concept then that concept will have a higher density.)

Fundamental ontology (starting part)



Path length and concept “height”



path_length(cat, dog) = 4

path_length(animate, inanimate) = 2

Animate and inanimate are more similar?

- Higher the concept, less specific it is
- Feature vector has less number of components
- Child concept inherits everything of parent plus adds its own
- Entropy is higher at higher levels of conceptual hierarchy (more heterogeneity)
- Semantic similarity will reduce at higher levels

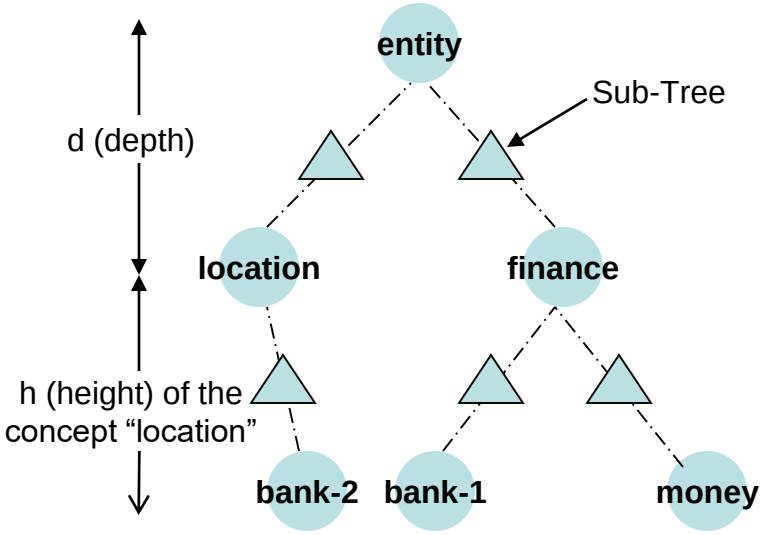
Relevance in the era of DL-NLP

- The notion of conceptual density is important for DL-NLP too
- Similarity in DL-NLP is computed by cosine similarity of word vectors
- Word vectors are created exploiting SYNTAGMATIC relations (coming from corpus)
- Ontology based similarity is computed using PARADIGMATIC relations

CONCEPTUAL DENSITY FORMULA

Wish list

- The conceptual distance between two word senses should be proportional to the length of the path between the two words in the hierarchical tree (WordNet).
- The conceptual distance between two word senses should be inversely proportional to the depth of the common ancestor concept in the hierarchy.



$$CD(c,m) = \frac{\sum_{i=0}^{m-1} nhyp^{i^{0.20}}}{descendants_c}$$

where,

c= concept

nhyp = mean number of hyponyms

h= height of the sub-hierarchy

m= no. of senses of the word and senses of context words contained in the sub-hierarchy

CD= Conceptual Density

and 0.2 is the smoothing factor

CONCEPTUAL DENSITY (cntd)

- The dots in the figure represent the senses of the word to be disambiguated or the senses of the words in context.
- The CD formula will yield highest density for the sub-hierarchy containing more senses.
- The sense of W contained in the sub-hierarchy with the highest CD will be chosen.

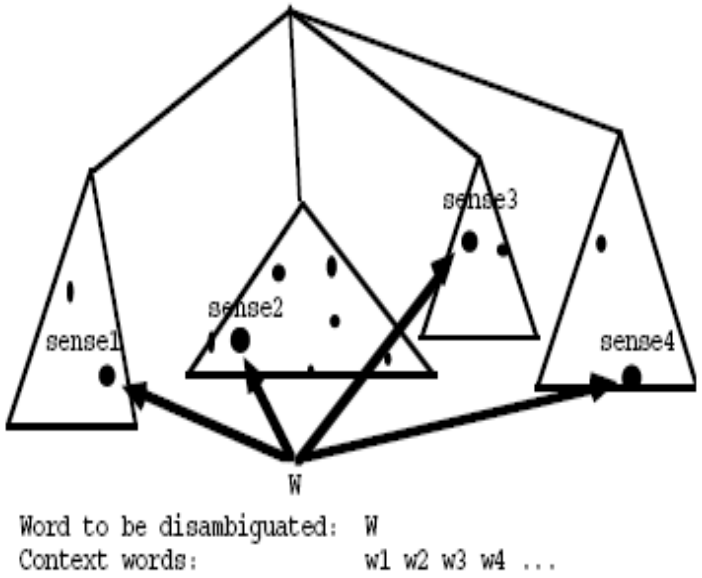
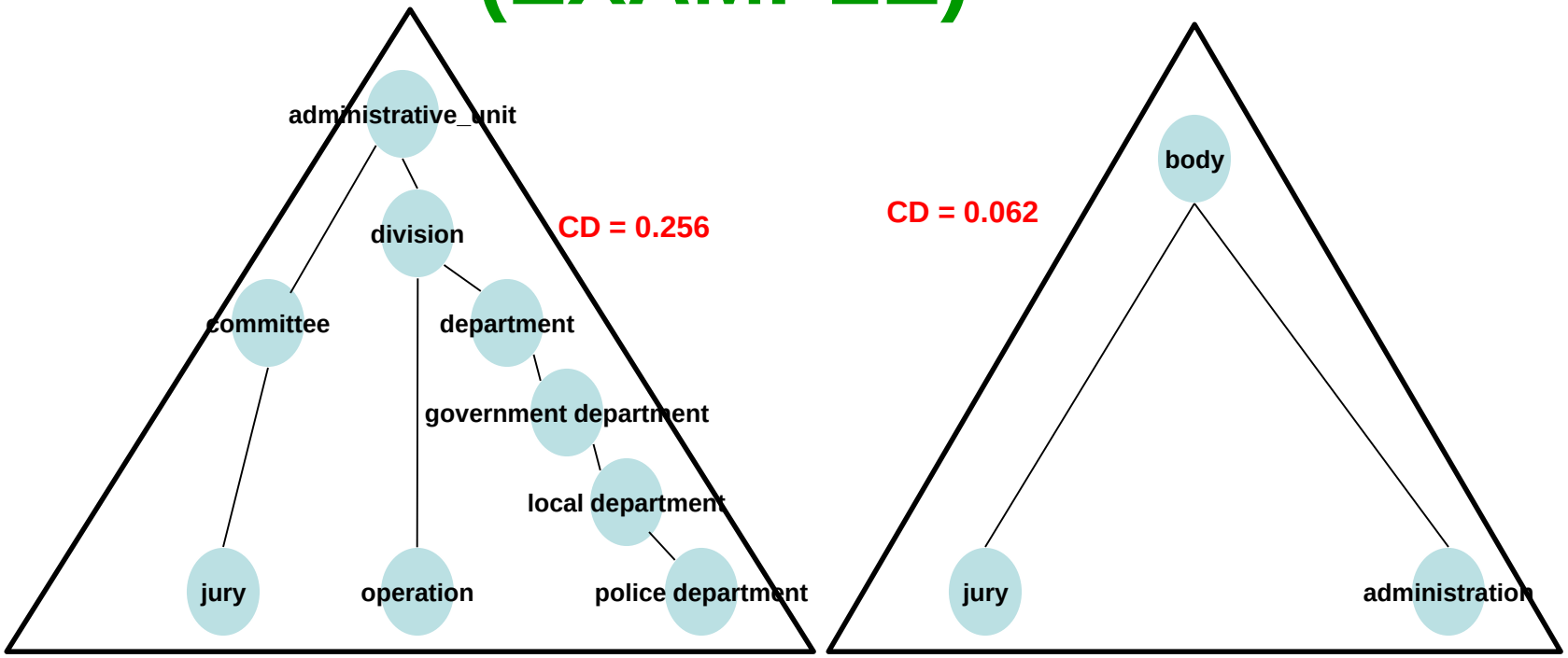


Figure 1: senses of a word in WordNet

CONCEPTUAL DENSITY (EXAMPLE)



The jury(2) praised the administration(3) and **operation** (8) of Atlanta Police Department(1)

Step 1: Make a lattice of the nouns in the context, their senses and hypernyms.

Step 2: Compute the conceptual density of resultant concepts (sub-hierarchies).

Step 3: The concept with the highest CD is selected.

Step 4: Select the senses below the selected concept as the correct sense for the respective words.

CRITIQUE

- Resolves lexical ambiguity of ***nouns*** by finding a combination of senses that maximizes the total Conceptual Density among senses.
- The Good
  - Does not require a tagged corpus.
- The Bad
  - Fails to capture the strong clues provided by proper nouns in the context.
- Accuracy
  - 54% on Brown corpus.

WSD USING RANDOM WALK ALGORITHM (Page Rank) *(sinha and Mihalcea, 2007)*

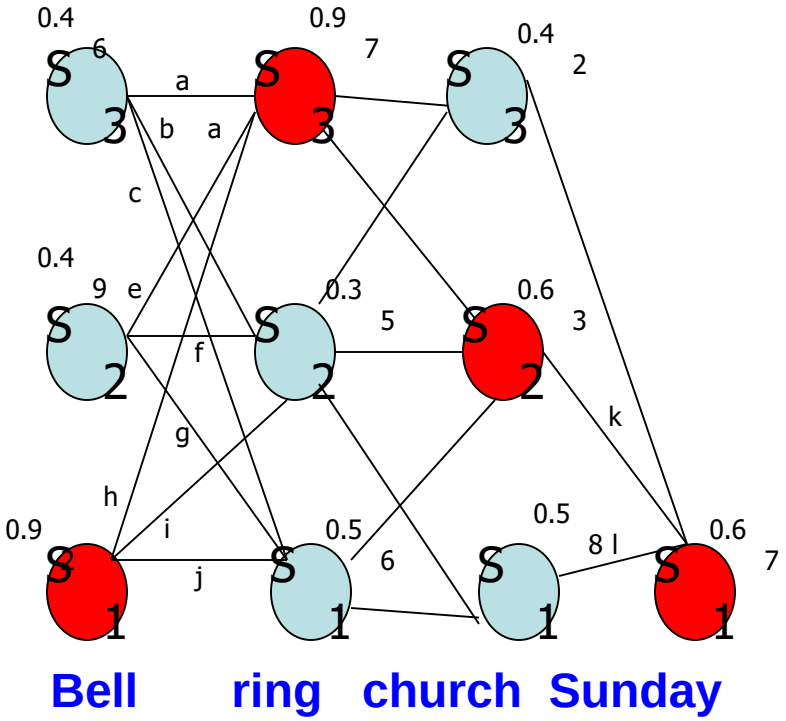
The **church bells** no longer **rung** on **Sundays**.

church
1: one of the groups of Christians who have their own beliefs and forms of worship
2: a place for public (especially Christian) worship
3: a service conducted in a church

bell
1: a hollow device made of metal that makes a ringing sound when struck
2: a push button at an outer door that gives a ringing or buzzing signal when pushed
3: the sound of a bell

ring
1: make a ringing sound
2: ring or echo with sound
3: make (bells) ring, often for the purposes of musical edification

Sunday
1: first day of the week; observed as a day of rest and worship by most Christians



Step 1: Add a vertex for each possible sense of each word in the text.

Step 2: Add weighted edges using definition based semantic similarity (Lesk's method).

Step 3: Apply graph based ranking algorithm to find score of each vertex (i.e. for each word sense).

Step 4: Select the vertex (sense) which has the highest score.

A look at Page Rank (from Wikipedia)

Developed at Stanford University by Larry Page (hence the name *Page*-Rank) and Sergey Brin as part of a research project about a new kind of search engine

The first paper about the project, describing PageRank and the initial prototype of the Google search engine, was published in 1998

Shortly after, Page and Brin founded Google Inc., the company behind the Google search engine

While just one of many factors that determine the ranking of Google search results, PageRank continues to provide the basis for all of Google's web search tools

A look at Page Rank (cntd)

PageRank is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page.

Assume a small universe of four web pages: **A**, **B**, **C** and **D**.

The initial approximation of PageRank would be evenly divided between these four documents. Hence, each document would begin with an estimated PageRank of 0.25.

If pages **B**, **C**, and **D** each only link to **A**, they would each confer 0.25 PageRank to **A**. All PageRank **PR( )** in this simplistic system would thus gather to **A** because all links would be pointing to **A**.

PR(A)=PR(B)+PR(C)+PR(D)

This is 0.75.

A look at Page Rank (cntd)

Suppose that page **B** has a link to page **C** as well as to page **A**, while page **D** has links to all three pages

The *value of the link-votes is divided among all the outbound links on a page*.

Thus, page **B** gives a vote worth 0.125 to page **A** and a vote worth 0.125 to page **C**.

Only one third of **D**'s PageRank is counted for A's PageRank (approximately 0.083).

PR(A)=PR(B)/2+PR(C)/1+PR(D)/3

In general,

PR(U)= $\sum_{V \varepsilon B(U)}$ PR(V)/L(V), where B(u) is the set of pages u is linked to, and L(V) is the number of links from V

A look at Page Rank (damping factor)

The PageRank theory holds that even an imaginary surfer who is randomly clicking on links will eventually stop clicking.

The probability, at any step, that the person will continue is a damping factor *d*.

$$PR(U) = (1-d)/N + d.\sum_{V \varepsilon B(U)} PR(V)/L(V),$$

N=size of document collection

For WSD: Page Rank

- Given a graph G = (V,E)
  - $In(V_i)$ = predecessors of V_i
  - $Out(V_i)$ = successors of V_i

$$S(V_i) = \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

- In a weighted graph, the walker randomly selects an outgoing edge with higher probability of selecting edges with higher weight.

$$WS(V_i) = \sum_{j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

Other Link Based Algorithms

- *HITS* algorithm invented by Jon Kleinberg (used by Teoma and now Ask.com)
- IBM *CLEVER project*
- *TrustRank* algorithm.

CRITIQUE

- Relies on random walks on graphs encoding label dependencies.
- The Good
  - Does not require any tagged data (a wordnet is sufficient).
  - The weights on the edges capture the definition based semantic similarities.
  - Takes into account global data recursively drawn from the entire graph.
- The Bad
  - Poor accuracy
- Accuracy
  - 54% accuracy on SEMCOR corpus which has a baseline accuracy of 37%.

KB Approaches– Comparisons

Algorithm	Accuracy
WSD using Selectional Restrictions	44% on Brown Corpus
Lesk's algorithm	50-60% on short samples of *"Pride and Prejudice"* and some "*news stories"*.
Extended Lesk's algorithm	32% on Lexical samples from Senseval 2 (Wider coverage).
WSD using conceptual density	54% on Brown corpus.
WSD using Random Walk Algorithms	54% accuracy on SEMCOR corpus which has a baseline accuracy of 37%.
Walker's algorithm	50% when tested on 10 highly polysemous English words.

KB Approaches– Summary

- Drawbacks of WSD using Selectional Restrictions
  - Needs exhaustive Knowledge Base.

- Drawbacks of Overlap based approaches
  - Dictionary definitions are generally very small.
  - Dictionary entries rarely take into account the distributional constraints of different word senses (e.g. selectional preferences, kinds of prepositions, etc. ***cigarette*** and ***ash*** never co-occur in a dictionary).
  - Suffer from the problem of sparse match.
  - Proper nouns are not present in a MRD. Hence these approaches fail to capture the strong clues provided by proper nouns.