Semantic Search

What is Search?

- We are given a set of objects and a query. We want to retrieve some of the objects.
- Interesting questions:
  - What information is associated with every object?
    - Single piece of data or multiple pieces of data?
    - Are the objects homogenous?
    - Is the information textual or structured data?
  - How is the query posed?
    - Do we use natural language text or do we have a query language with strict grammar?
  - How is the result calculated?
    - How do we rank the result?
    - What are data structures we can use for efficient query answering? (pre-computed results and indexes)

The set of objects

- Key-value pairs
- A set of homogenous objects with the same attributes (e.g., a set of employees with ssn, name, age, and departmentID)
- A set of heterogeneous objects. For example, we can put employees, children and department in the same set. Every object can have different attributes and we do not have a data schema.
- The information that is associated with every object can be hierarchical (e.g., an XML document).
- We can have a formal description for every object in a logical language (e.g., OWL)
- Or maybe we just have a textual description associated with every object (e.g., white sneakers size 10).

Key-value stores

cassandra

ORACLE
NOSQL DATABASE

Berkeley DB

Bigtable

mongoDB®

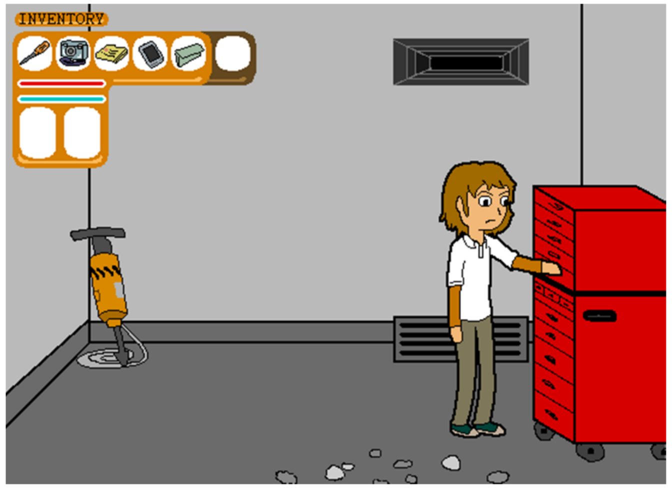
riak

OPENLINK
VIRTUOSO

Cloudify
Any App. On Any Cloud. Your Way.

Key-value Store Implementation

- We are given a key-value combination.
- We can store it in a hash table (possibly distributed over many servers).
- The keys must be unique.
- The query specifies the key and retrieves the value.
- Usually implemented using a hash table.



Homogenous Objects

ssn	name	age	salary
234324465	Bob	23	$40K
242343223	John	43	$120K
643325243	Ann	33	$150K
235342567	Suzan	62	$66K

Employee Table

```
select *
from Employee
where salary > $100K
```

s1

Semi-Structured (Heterogeneous) Objects

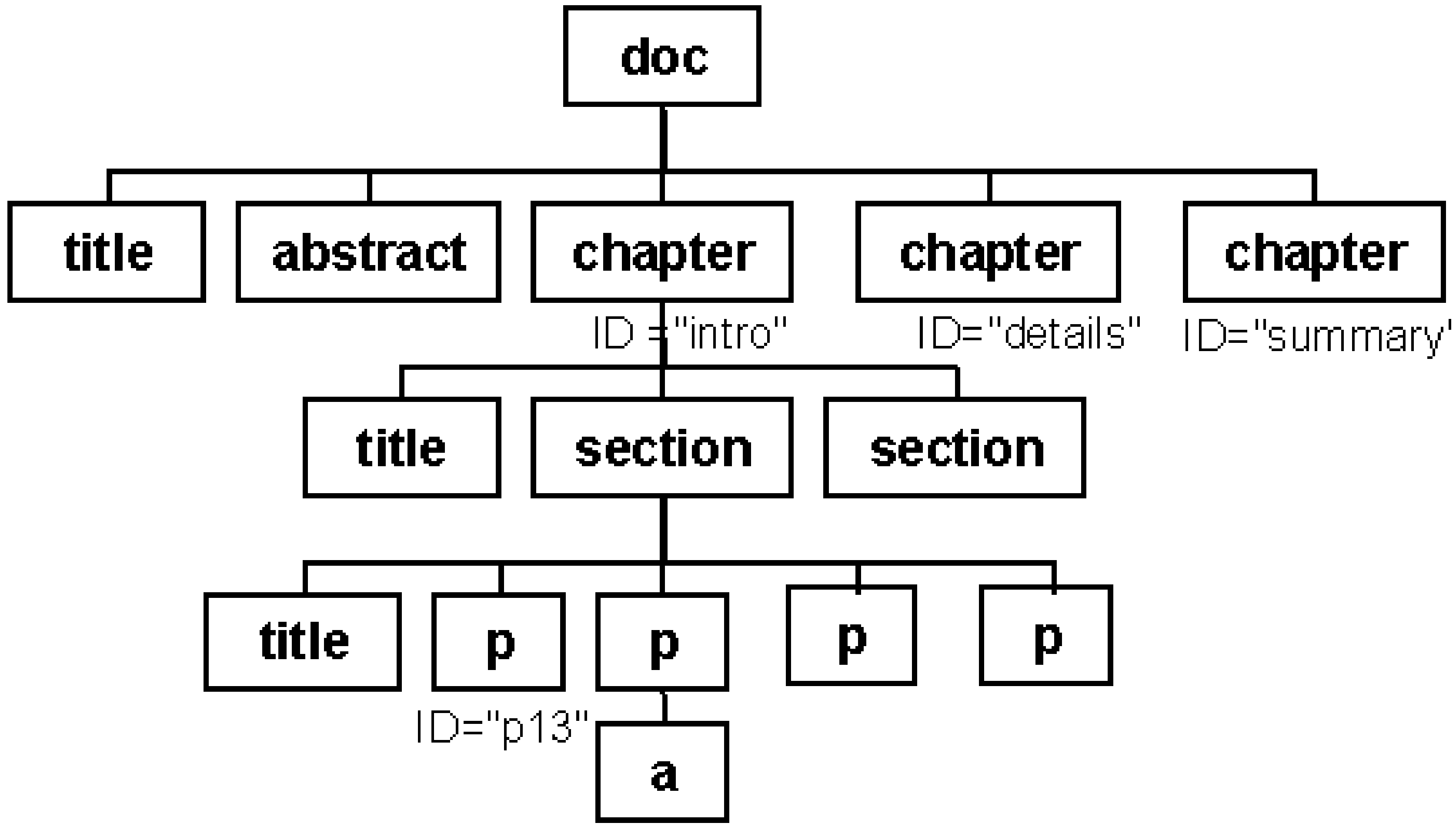
```
<!-- reviewedbooks.dtd --> (document type definition)

<!DOCTYPE reviewedbooks
[
<!ELEMENT reviewedbooks ( book+)>
<!ELEMENT book ( title, author+, publisher, isbn,
review*)>
<!ELEMENT title ( #PCDATA)>
<!ELEMENT author ( firstname, lastname, flag)>
<!ELEMENT firstname ( #PCDATA)>
<!ELEMENT lastname ( #PCDATA)>
<!ATTLIST flag gender ( M | F ) “F”>
<!ELEMENT publisher ( #PCDATA)>
<!ELEMENT isbn ( #PCDATA)>
<!ELEMENT review ( #PCDATA)>
]
>
```

Slide 7

s1 stanchev, 2/10/2016

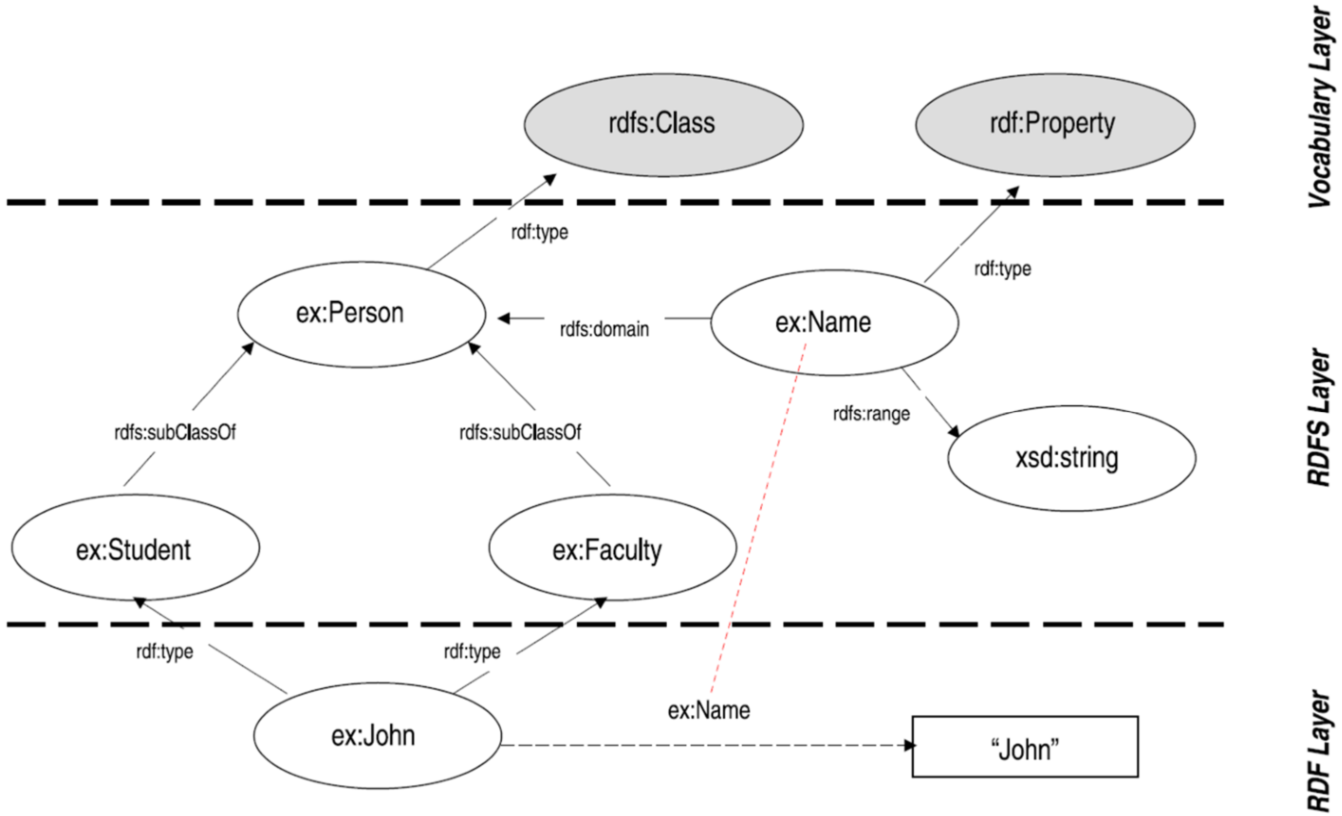
Actual XML

```
<book>
  <title>Gone with the Wind</title>
  <author>Margaret Mitchell</author>
  <publisher>Warner Books</publisher>
  <isbn>0446365386</isbn>
  <review>
    Sometimes only remembered for the epic motion
    picture and "Frankly ... I don't give a damn,"
    Gone with the Wind was initially a compelling and
    entertaining novel. …
  </review>
</book>
```


doc
title
abstract
chapter
chapter
chapter
ID ="intro"
ID="details"
ID="summary"
title
section
section
title
p
p
p
p
ID="p13"
a

(Resource Description Framework)

RDF Objects



Ontology Web Language (OWL)

pepperoni pizza

```
Pizza and (some topping
pepperoni) and
(some ingredient cheese)
```

Hawaiian pizza

```
DeepDishPizza
and (some topping
pineapple) and
(some ingredient tomato)
```

SPARQL Query

```
SELECT * WHERE
{
  ?pizza rdfs:subClassOf [
    owl:onProperty :hasTopping;
    owl:someValuesFrom :MozzarellaTopping ] .
  ?pizza rdfs:subClassOf [
    owl:onProperty :hasTopping;
    owl:someValuesFrom :PeperonSausageTopping ] .
  ?pizza rdfs:subClassOf [
    owl:onProperty :hasTopping;
    owl:someValuesFrom :TomatoTopping ] .
}
```

Text Descriptions

White sneakers size 8

Air Jordan Shoes

Query: "Running footwear"

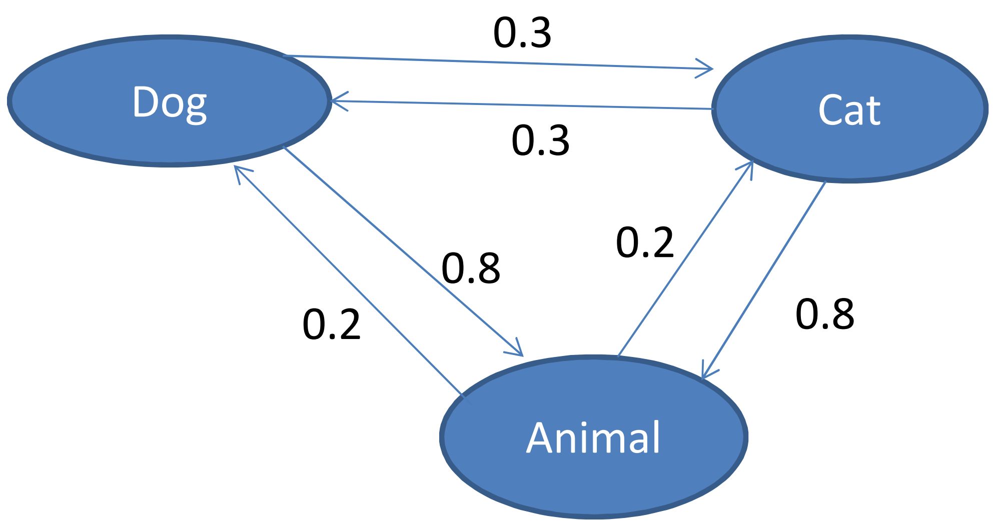
Keywords-based Search

- Terms Frequency - Inverse Document Frequency (TF-IDF)
- The query is parsed into terms and the weight of each term is evaluated.

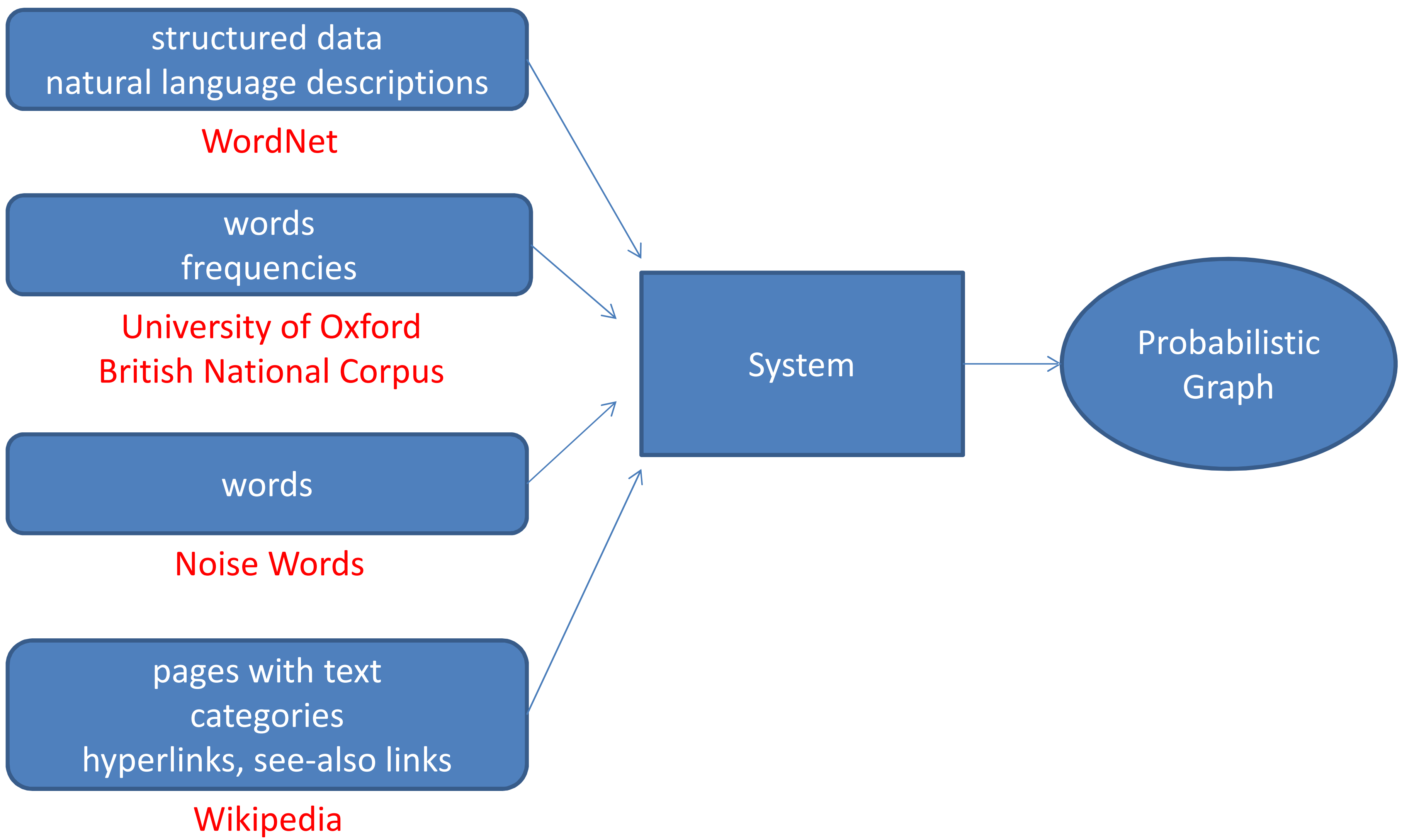
$$weight = \sqrt{tf} * (1 + log_2(\frac{numDocs}{docFreq+1}))^2$$

- `tf` is the number of times the terms appears in the document
- `numDocs` is the total number of documents.
- `docFreq` is the number of documents in which the term appears.
- Idea: if the query contains rarely used words, then the documents that contain them are given more weight.
- Advantage: good ranking of documents based on keywords matching.
- Problem: doesn't work on example from previous slides.

Probabilistic Graph Approach



Our System



structured data
natural language descriptions
WordNet
words
frequencies
University of Oxford
British National Corpus
words
Noise Words
pages with text
categories
hyperlinks, see-also links
Wikipedia
System
Probabilistic
Graph

About WordNet

- WordNet gives us information about the words in the English language.
- In our study, we use WordNet 3.0, which contains approximately 150,000 different words.
- WordNet also contains phrases (or word forms), such as sports utility vehicle.
- The meaning of a word form is not precise. For example, spring can mean *the season after winter*, *a metal elastic device*, or *natural flow of ground water*, among others.
- WordNet uses the concept of a sense. For example, spring has the three senses.
- Every word form has one or more senses and every sense is represented by one or more word forms. A human can usually determine which of the many senses a word form represents by the context in which the word form is used.

About WordNet (cont'd)

- WordNet contains the definition and example use of each sense. It also contains information about the relationship between senses.
- The senses in WordNet are divided into four categories: nouns, verbs, adjectives, and adverbs.
- For example, WordNet stores information about the hyponym and meronym relationship for nouns. The hyponym relationship corresponds to the ``kind-of" relationship (for example, dog is a hyponym of canine).
- The meronym relationship corresponds to the part-of relationship (for example, window is a meronym of building). Similar relationships are also defined for verbs, adjectives, and adverbs.

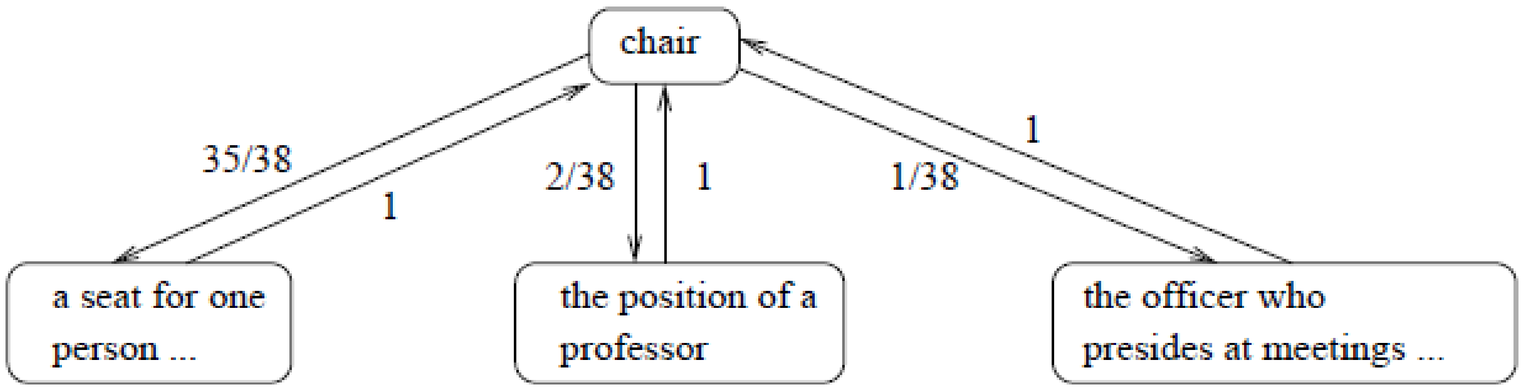
About Wikipedia

- Pages have titles and subtitles.
- Pages have see-also links.
- Pages contain text.
- Pages contain hyperlinks.
- Pages belong to categories.
- Categories can have super-categories and sub-categories.
- There are page redirects. One page redirects to another page.
- There are disambiguation pages.

Initial Probabilistic Graph

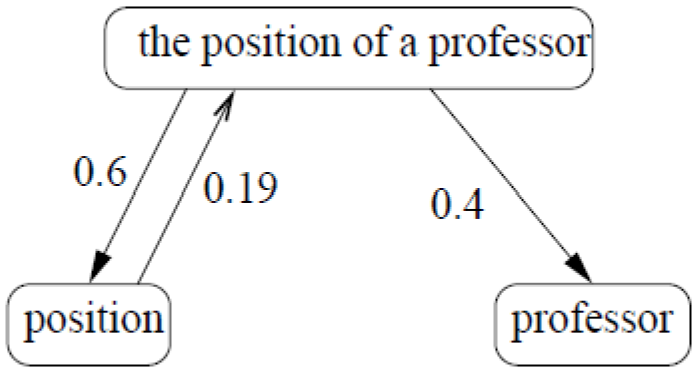
- Create a node for every word form from WordNet.
- Create a node for every sense.
- Create a node fore every Wikipedia page.
- Create a node for every Wikipedia category.

Processing the Senses (WordNet)



Frequency of use of each sense is given in WordNet.

Adding Definition Edges (WordNet)

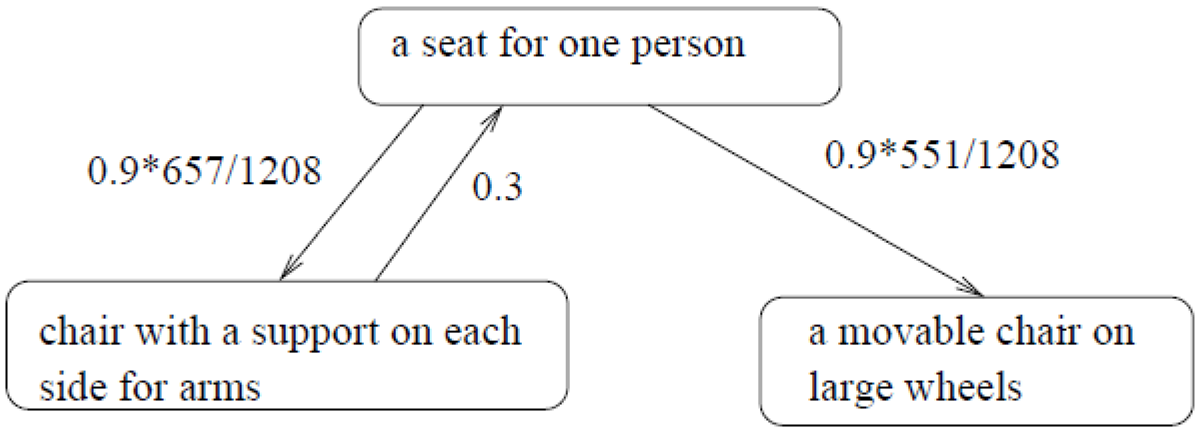


$$computeMinMax(minValue, maxValue, ratio) = minValue + (maxValue - minValue) * \frac{-1}{log_2(ratio)}$$

- Position is first word, so we give it greater importance.
- Forward edge: computeMinMax(0,0.6,ratio).
- If position appears in only three word form definitions,

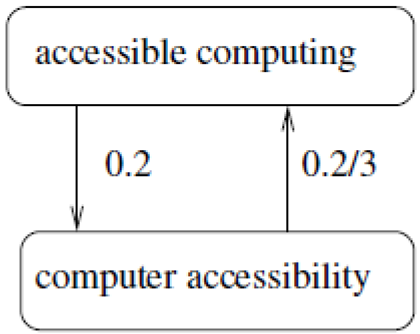
then we compute backward edge as computeMinMax(0,0.3,1/3).

Processing Hyponyms (WordNet)



In the British National Corpus, the frequency of armchair is 657 and the frequency of wheelchair is 551.

Page Redirections (Wikipedia)



We assume that there are three redirects that are pointng to computer accessibility.

Connecting Wikipedia and WordNet

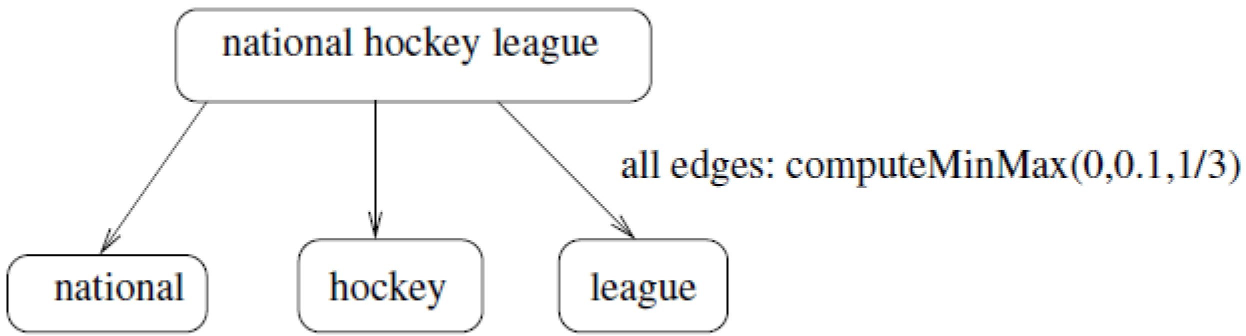
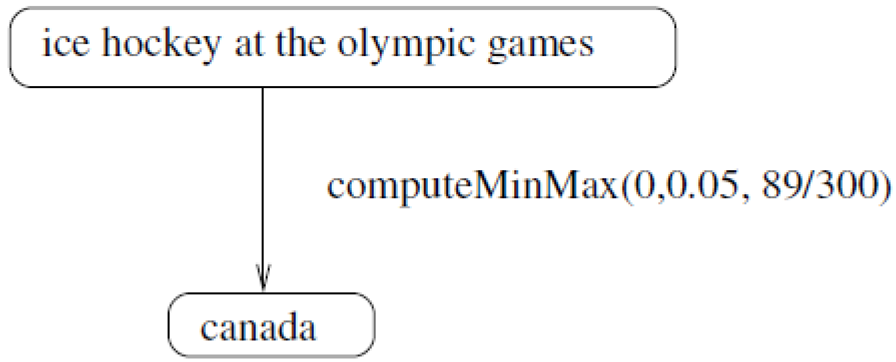


Fig. 2. Wikipedia pages to word form edges.

We perform this algorithm for Wikipedia titles, subtitles, and categories.

Processing Wikipedia Text



- We only consider words or phrases that appear five times or more. We consider single words, pair of words, and triplets of words.
- In our example, Canada appears 89 times in the document and the document has a total of 300 words that are part of frequently occurring phrases.

See-also links and Hyperlinks

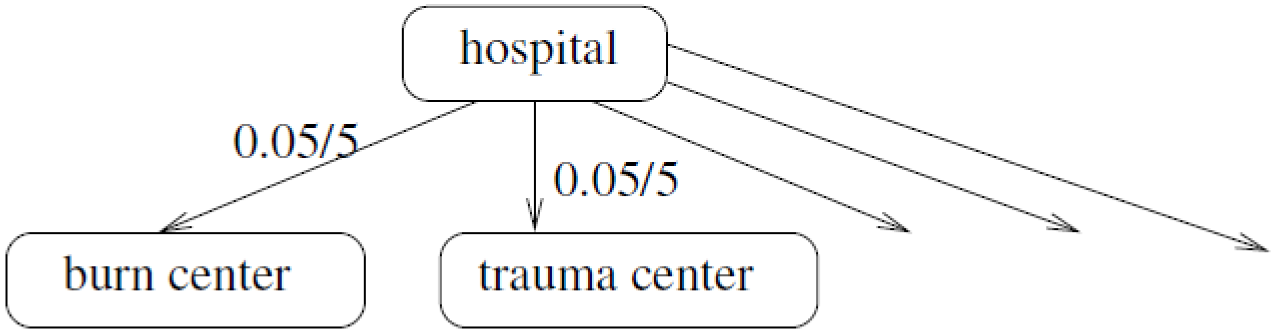


Fig. 5. Edges for see-also links.

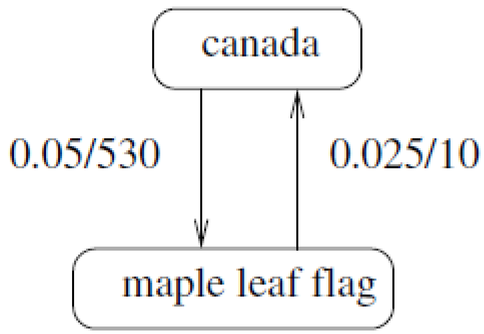


Fig. 6. Edges for hyperlinks.

The Wikipedia page contains a total of 530 distinct hyperlinks. At the same time, there are 10 links that point to maple leaf flag.

Pages and Categories

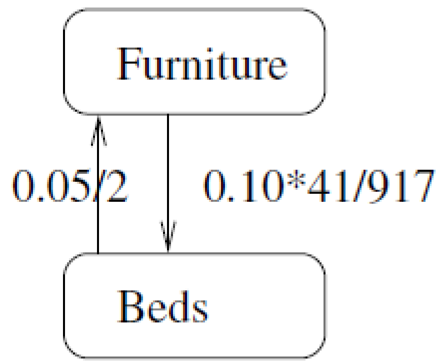


Fig. 7. Edges for subcategories.

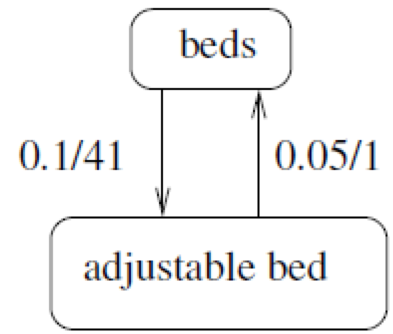
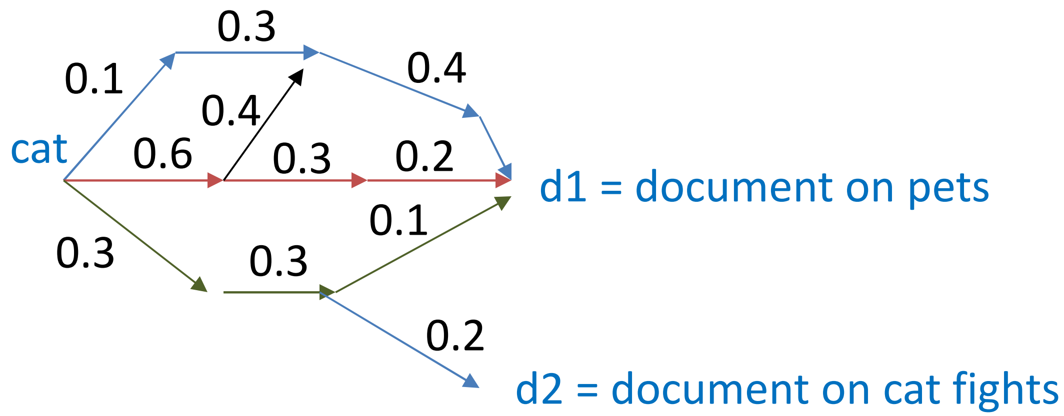


Fig. 8. Page-category edges.

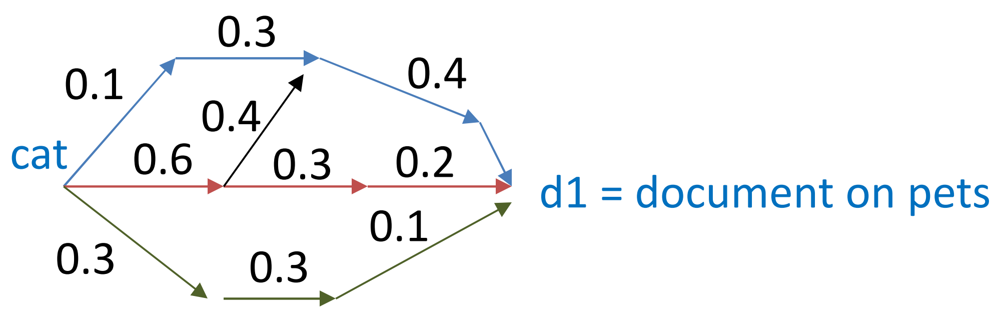
- The "size" of furniture is 917 and the size of beds is 41
- There are two super-categories of beds: furniture and sleep.

The Probabilistic Graph



- Normalize graph:
  - "Add" probabilities of identical edges using Markov Logic Network Model.
  - Sum of the weights of all edges going out of a node = 1
- Big Money Question:
  - P(relevant(d1)|relevant(cat)) = ?
  - P(relevant(d2)|relevant(cat)) = ?

Multiply probabilities



0.1*0.3*0.4+0.6*.03*0.2+0.3*0.3*0.1

Problem: can't count all edges

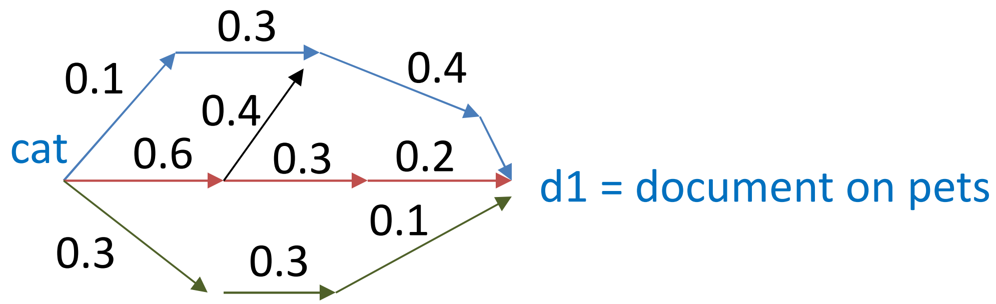
Markov Logic Network

$$A_n \xrightarrow{w_{n-1}} A_{n-1} \quad \cdots \quad A_1 \xrightarrow{w_0} A_0$$

$$\frac{P(rel(A_0) \wedge rel(A_n))}{P(rel(A_n))} = \frac{f11(n-1)}{f10(n-1) + f11(n-1)}$$

$$\begin{aligned} f00(0) &= e^{w_0} \\ f01(0) &= e^{w_0} \\ f10(0) &= 1 \\ f11(0) &= e^{w_0} \end{aligned} \qquad \begin{aligned} f00(i) &= f00(i-1) * e^{w_i} + f10(i-1) * e^{w_i} \\ f10(i) &= f00(i-1) * 1 + f10(i-1) * e^{w_i} \\ f01(i) &= f01(i-1) * e^{w_i} + f11(i-1) * e^{w_i} \\ f11(i) &= f01(i-1) * 1 + f11(i-1) * e^{w_i} \end{aligned}$$

Random Walk



Start with cat and follow a random path. Roll a dice to determine where to go at each node. Go for 5 edges. Do 10000 runs and see in how many d1 will be reached.
Problem: What if we never reach some documents?

Conclusion

- Regular search: find the requrested information.
- Problem: how do you take the meaning of the data into account. How do you rank the answers?
- Semantic Search: Return documents that are semantically relevant.
- Probability can be used to rank the documents in the answer.