System for extracting data (facts) from large amount of unstructured documents (INTERNET)

Luka Bradeško

Jožef Stefan Institute

KnowItAll

Modelled on paper by Oren Etzioni et al. : Web-Scale Information Extraction in KnowItAll

What are we solving

- **Problem:** to manually search for factual information on large amount of data, is a tedious, error-prone process
- **Solution:** user describes wanted class using simple search query and gets a list of instances as a result.
- Examples:
  - **Search query:** Nobel prize winner/winners
    Instances: Albert Einstein, Kris Cardenas,Yasunori Kawabata,...
  - **Search query:** City/Cities
    Instances: Ljubljana, New York, Berlin,...

Extracting Nobel prize winners

- Input:
  - Search query: “nobel prize winnner;laureate”, “nobel prize winners;laureates”
  - Extraction rules:
    - ”<class2> such as <NPList>”
    - ”<NP> is a <class>“
    - “<class2> including <NPList>”
- Generate *extraction queries* for a search engine:
  - ”nobel prize winners such as *“
  - “* is a nobel prize winner“
  - “nobel prize winners including *”

Extracting Nobel prize winners

- Send *extraction queries to the* search engine and collect *query results:*

Utrecht travel guide - Wikitravel
... Van Unnik and Freudenthal. The list also includes **Nobel Prize Winners such as Gerard** 't Hooft (1999) from the Faculty of Physics and Astronomy. ...
wikitravel.org/en/Utrecht - 43k - Cached - Similar pages - Note this

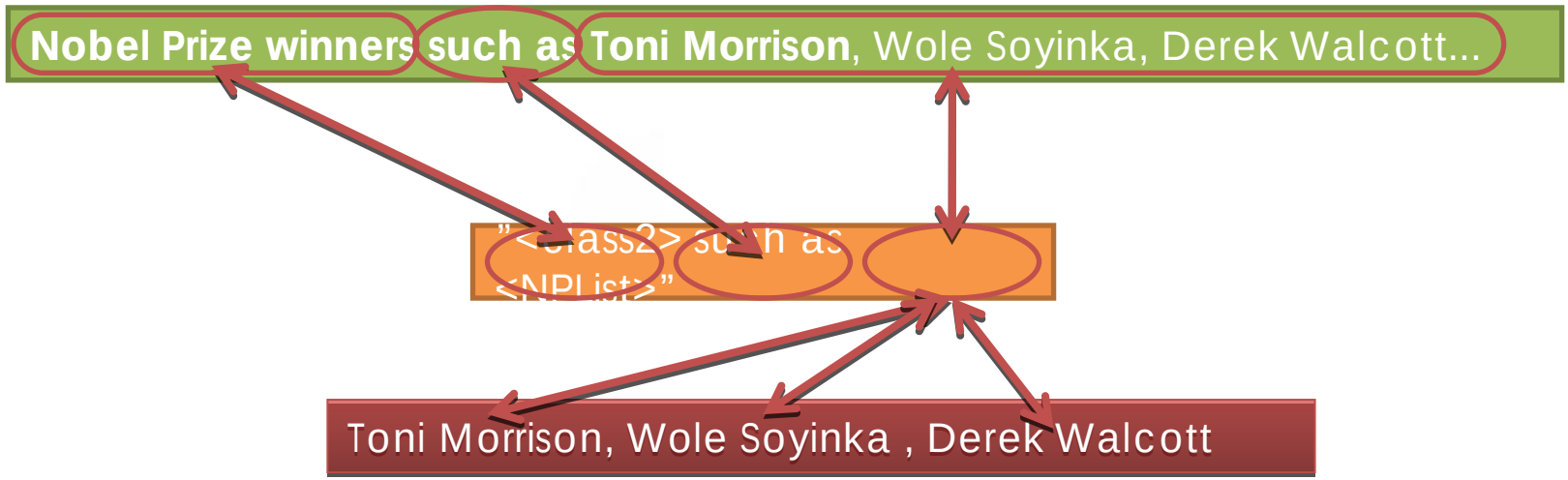
Universiteit Utrecht
The list also includes **Nobel Prize Winners such as Gerard** 't Hooft from the Faculty of Physics and Astronomy. Visit the Universiteit Utrecht website ...
www.wun.ac.uk/view.php?id=12 - 7k - Cached - Similar pages - Note this

University of Glasgow :: SMLC - Slavonic Studies :: Information ...
Great classical Russian writers such as Pushkin, Tolstoi, Dostoevskii and Chekhov loom large, as do **Nobel Prize winners such as Pasternak**, Solzhenitsyn, ...
www.arts.gla.ac.uk/Slavonic/prospective2006.htm - 21k -
Cached - Similar pages - Note this

[PDF] Research in Baden-Württemberg
File Format: PDF/Adobe Acrobat - View as HTML
produced more generations of **Nobel prize winners such as Albert**. Einstein, who was born in Ulm, Christiane Nüsslein-Volhard from ...
www.study-guide-bw.com/download.php/r4yiR/a/474.pdf - Similar pages - Note this

Extracting Nobel prize winners

- Extract candidates from snippets, using *extraction rules:*



Assessing the Candidates

- Generate *discriminators* (from rules and user input):
  - ”nobel prize winners such as <Candidate>”
  - “<Candidate> is a nobel prize winner“
  - “<Candidate> is a nobel prize laureate”
  - “nobel prize winners, including <Candidate>”

- Generate *discriminator queries* (from discriminators and candidates):
  - ”cities such as Toni Morrison”
  - “Toni Morrison is a nobel prize winner“
  - “Toni Morrison is a nobel prize laureate”
  - “nobel prize winners, including Toni Morrison”

- Evaluate candidates
  - Calculate PMI for each discriminator query

$$PMI = \frac{|HITS(D+C)|}{|HITS(C)|} \rightarrow \frac{HITS(\text{"}\mathbf{nobel\ prize\ winner\ Toni\ Morrison}\text{"})}{HITS(\text{"}\mathbf{Toni\ Morrison}\text{"})} = \frac{1,760}{1,640,000} = 0.001073$$

- Use PMI numbers as features for learning algorithms

Bootstraping and feature selection

- We need a training set of positive and negative instances of the target class
- Instead of evaluating all candidates, which is time consuming, we select **n** seed candidates
- Use average PMI as evaluation for positive and negative seeds
  - **m** candidates with highest average PMI are positive
  - **m** candidates with lowest average PMI are negative
- Select **k** beset discriminators, tested on **m** positive and negative seeds (this two steps can be iterated)
- Evaluate all candidates on **k** discriminators and use PMI as features.

Result: Tony Morrison (0.001073, 0.000, 0.3221, ...)

Pseudo code

```
KNOWITALL (information focus I, rules R)
{
   Q = make queries for each I from R
   Candidates C
   for each Q
   {
      w = send Q to search engine(s)
      for each w
      {
         C = Extract from w, using R
      }
   }
   D=make discriminators from R.
   Asses random n Candidates
   take m Candidates with best PMI as seeds
   take m Candidates with 0 PMI as negative seeds
   select k best discriminators
   Asses all Candidates
}
```

Hit limit in Search Engines

- **Problem:** Search engines have limit of top 1000 results per query

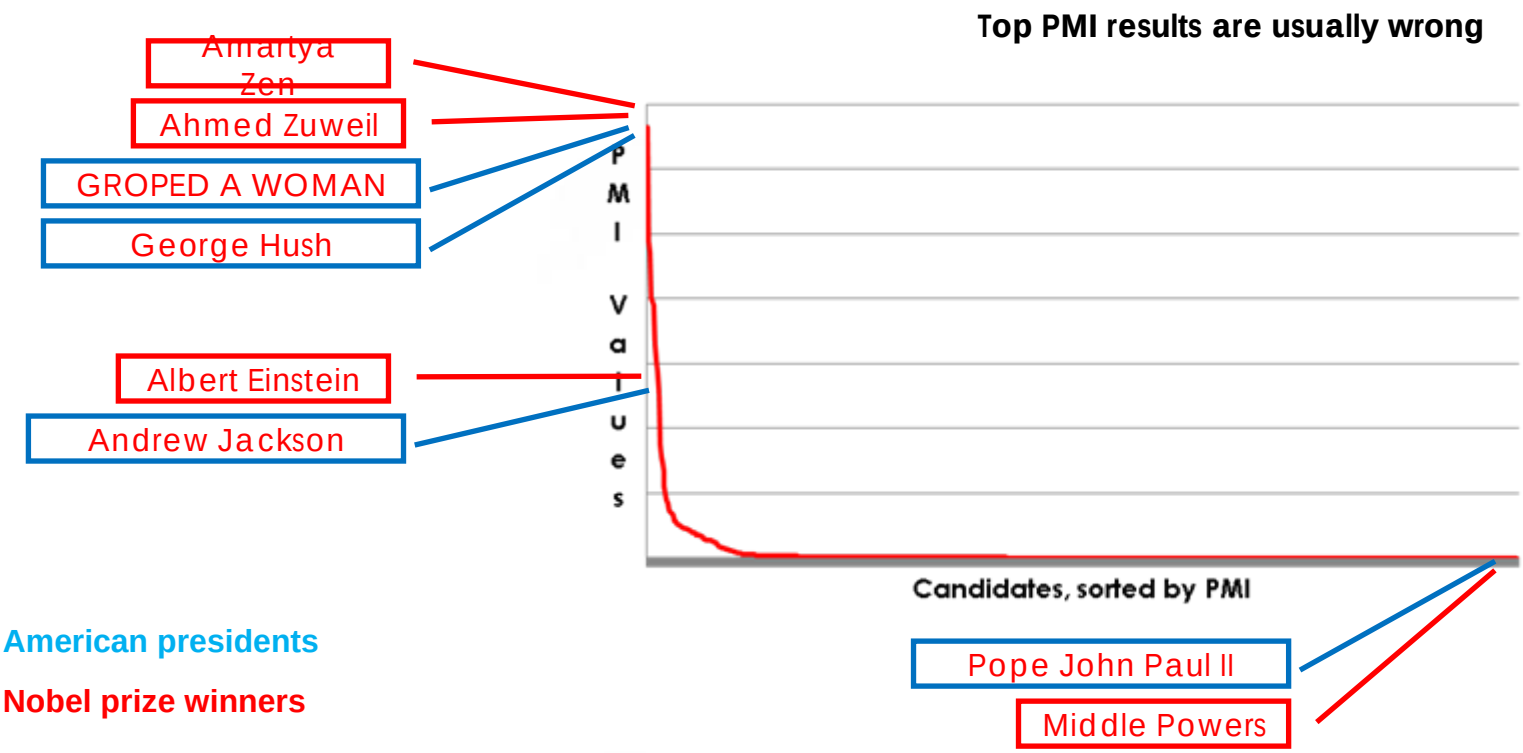
- **Solution:** Recursive Query Expansion
  - **Q= original query** (i.e.1900 results)
    Q1= Q + w (w= pre-specified list of words. Usually most frequent words) (1200 results)
    Q2 = Q – w (700 results)
  - **Example:**
    Q= “nobel prize winners such as *”
    Q1=“nobel prize winners such as *” + “what“
    Q2=“nobel prize winners such as *” - “what“

Difficulties

- Features
  - PMI does not always give needed information
  - Best 5 PMI results are usually not correct. (hard to choose positive and negative seeds)
  - **Solution:** use more , or other features:
    - PMI, hits, page rank, **redundancy number, number of independent positive PMIs**
- Precision/Recall
  - Hard to calculate, or even approximate on some classes
- Time complexity
  - **Problem:** Fetching web pages and assesing candidates is time consuming.
  - **Solution:** Extracting just from snippets is as good as from web pages. For fast results use redundancy numbers instead of PMI

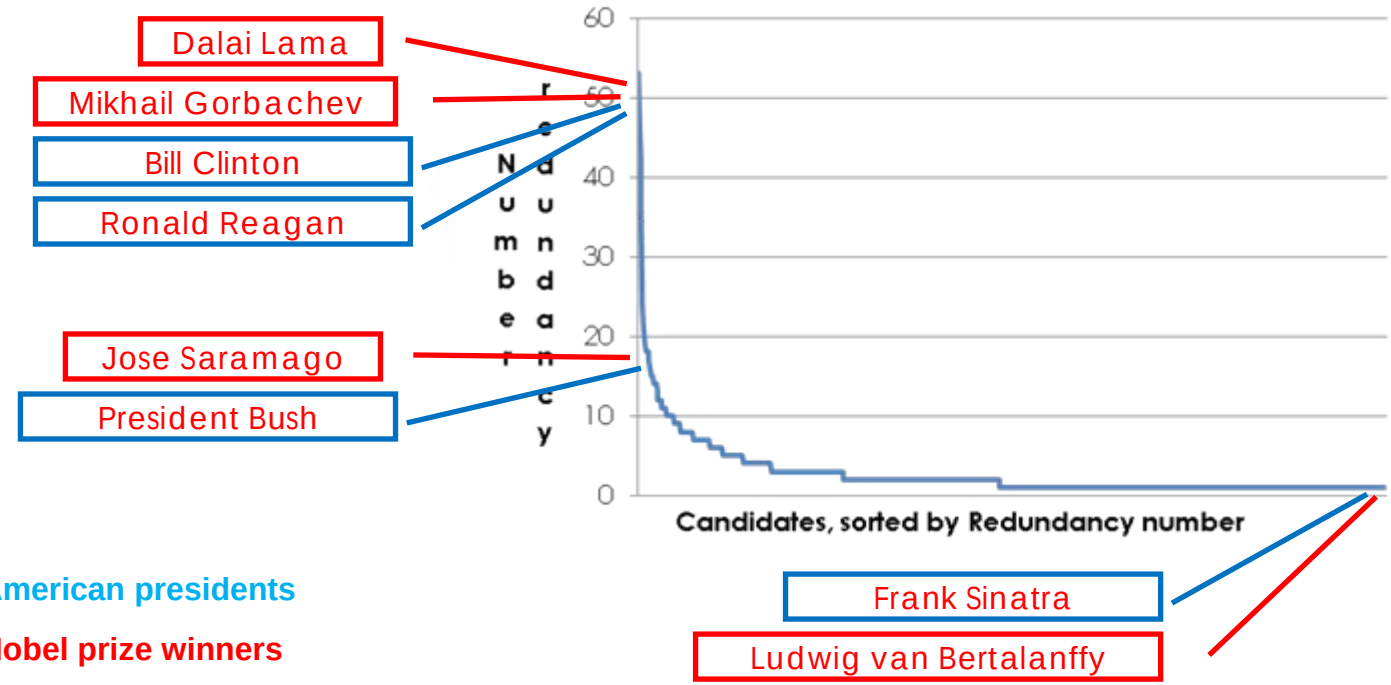
Features problem

- PMI features, used in original KnowItAll were not always useful.
  The problem is, when there is a few examples of candidate written in wrong way and they all come from one extractor.
  In this case PMI measure is very high. It can come close to 1/m, where m is number of assessors. Normal PMI should be in range 1/100m.



Features solution

- For bootstrapping and even if we just use treshold for dividing positive classes from negative, redundancy of candidates seems better evaluation than PMI.
  Top most hits are in most cases correct.
- Good solution would be also weighted redundancy, PMI and combination
- Positive PMI number gives similar results than redundancy number, but is not that fast.



Precision/recall

- Tested on searching for Nobel prize winners/ american presidents
- Results only from first 1000 hits
- Results edited (A. Einstein –> Albert Einstein)

PMI based		
	Precision	Recall
Nobel w.	83,7	53,4
presidents	66,0	81,4

Redundancy based (first 100,35)		
	Precision	Recall
Nobel w.	100	12,7
presidents	90	65

Conclusion

- Redundancy number is better evaluation method than PMI.
  - From some treshold point on, we can be almost 100% sure that candidates are positive (this is not true for PMI)
  - When used on all hits, recall increases without affecting precision
  - For automatic ontology populating, precision is more important than recall
- If you don't need probabilistic evaluation, redundancy method is really good for getting fast, precise results
- Best way to evaluate candidates is to use all of possible **weighted** features (PMI, redundancy, independence, hits, ...)

Further work

- What is working
  - Backbone of knowItAll system (structures, main algorithms, principles)
  - Working knowItAll prototype
  - Some new features for machine learning algorithms
  - Playground for testing different learning methods
- What is in development stage
  - Recursive Query Expansion
  - Multithreading (faster fetching of web results)
  - Learning methods (Authors of original knowItAll found that simple treshold is better than Naive Bayes they used)
  - Multiple search engines
  - Database (caching, storing previous searches, tracking of errors)