# Semantic Analysis

Mariana Romanyshyn
Grammarly, Inc.

# Contents

1. Word sense disambiguation
2. Semantic role labeling
3. Textual entailment
4. Semantic parsing
5. Question answering

# 1.

# Word sense disambiguation

# Words have meanings

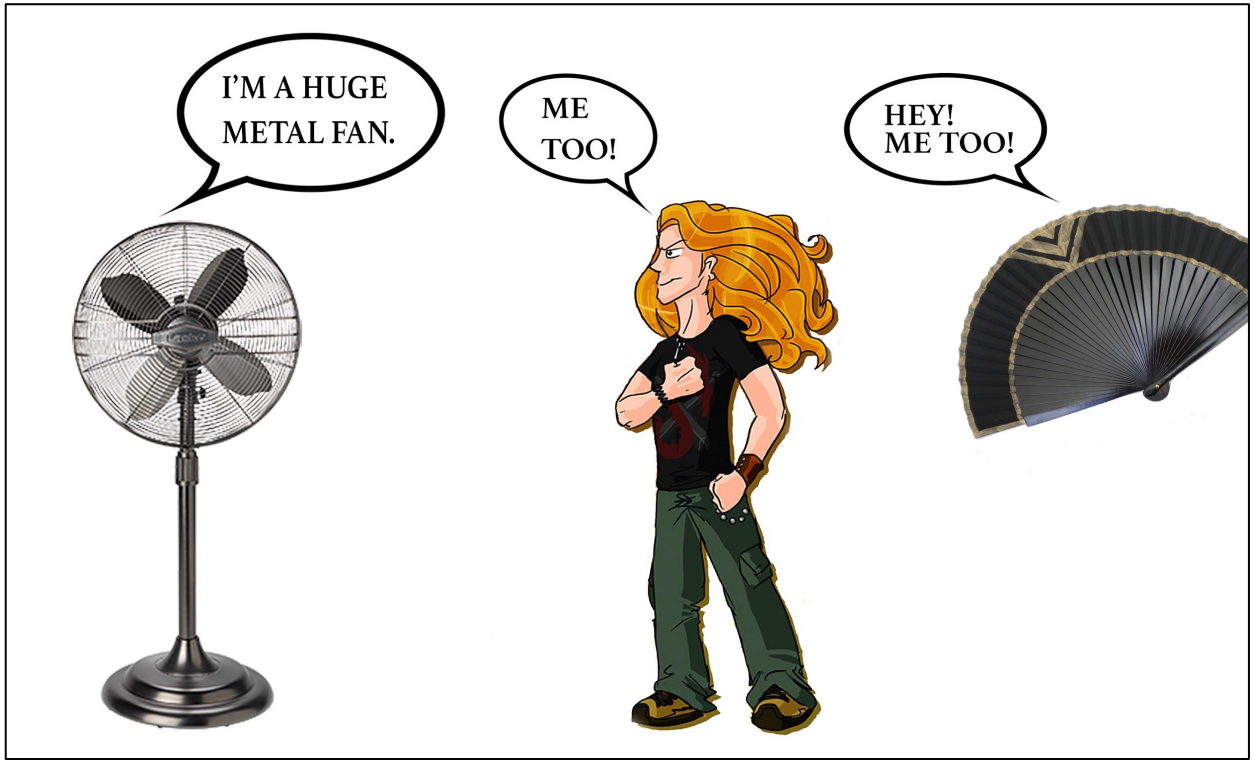



Image by Tetiana Turchyn

# Words have meanings

# Is it serious?

- ~40% of English words are polysemous
- most polysemous - verbs (~55% in WordNet)
- resources disagree
  - *“head”*, noun:
    - 11 meanings - Macmillan Dictionary
    - 16 meanings - Longman Dictionary
    - 33 meanings - WordNet
    - 34 meanings - Oxford Dictionary
- meanings overlap
  - *John works for the **newspaper** that you are reading.*

# Is it just English?

*... зробити так, щоби впала **стіна**?*

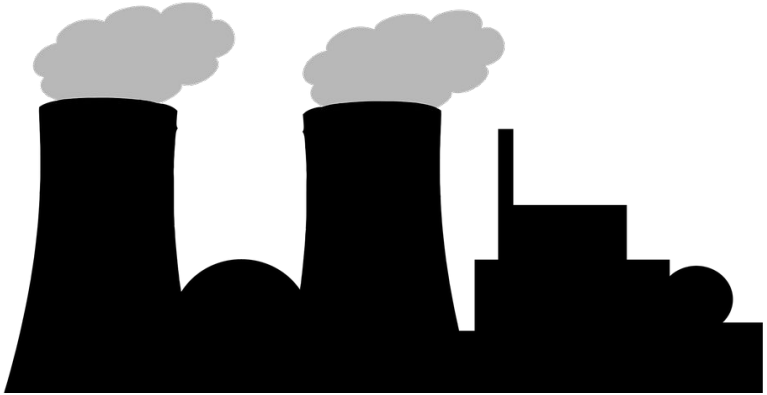
1. стіна будинку
2. стіна урвища
3. мур
4. те, що відокремлює, роз'єднує

# Personal assistants

**You:** I need to buy a big **plant** for my mom. She likes gardening!

**Siri:** Hmm...

?



# Information retrieval

Where is **Paris** now and what is she doing?

?

# Sentiment analysis

Paris Hilton is very **rich**.

This area is **rich** in natural resources.

These comments are a bit **rich** coming from someone with no money worries.

# Error correction

**Animate or inanimate?**

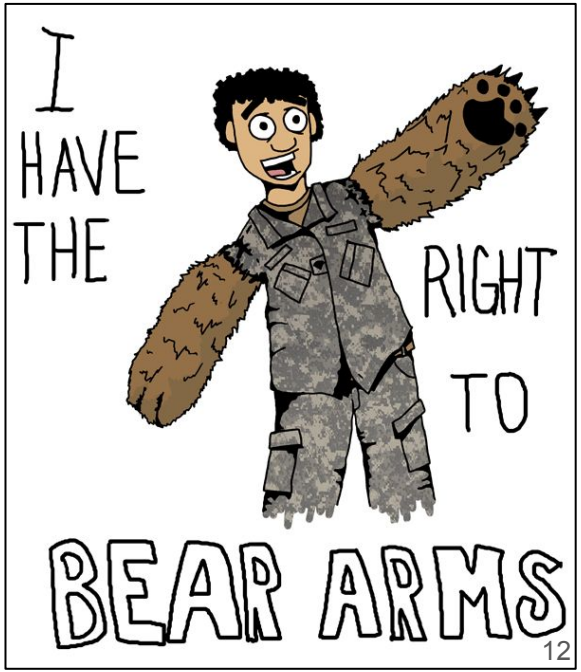
Then we have 2 laptop stands & 2 **mouses** left.

My cat catches **{mouses=>mice}** all the time.

# Text mining

US **sells arms** to countries well-known for violating human rights.

Using recycled prosthesis, a hospital in Tanzania **sells arms** for around $500 each. There is also high demand for legs.



# What is “sense”?

- senses = domains?
- senses = sentiments?
- senses = animate/inanimate?
- senses = jargon/standard?
- senses = countable/uncountable?
- senses = entities?
- senses = senses?

# Dictionaries

**bank** (*plural* **banks**)

1. (*hydrology*) An edge of river, lake, or other watercourse. [quotations ▼]
2. (*nautical*, *hydrology*) An elevation, or rising ground, under the sea; a shallow area of shifting sand, gravel, mud, and so forth (for example, a sandbank or mudbank).

   *the **banks** of Newfoundland*

3. (*geography*) A slope of earth, sand, etc.; an embankment.
4. (*aviation*) The incline of an aircraft, especially during a turn.
5. (*rail transport*) An incline, a hill.

Example from en.wiktionary.org

# Dictionaries

**man**[1] /mæn/ ●●● S1 W1 **noun** (*plural* **men** /men/)

1 MALE PERSON [countable] an adult male human → **woman**

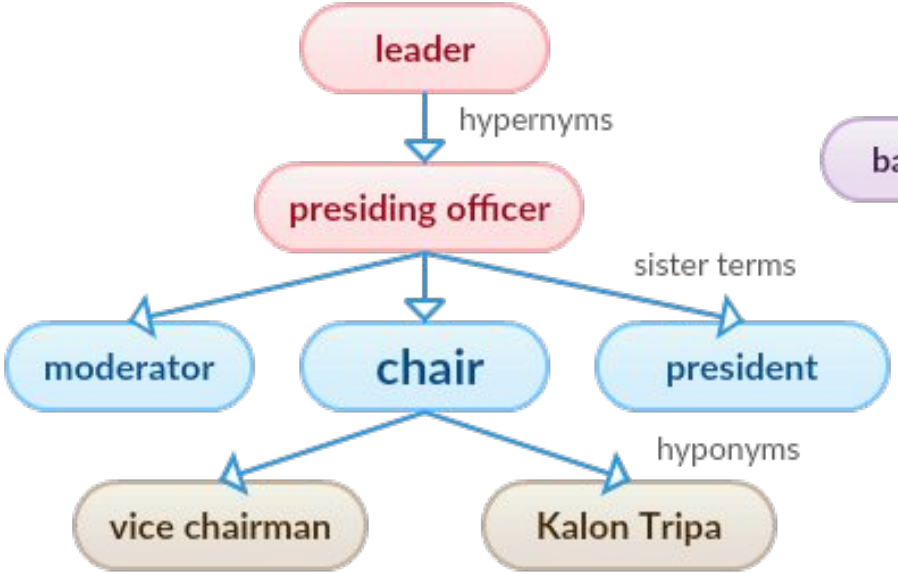
2 STRONG/BRAVE [countable usually singular] a man who has the qualities that people think a man should have, such as being brave, strong etc

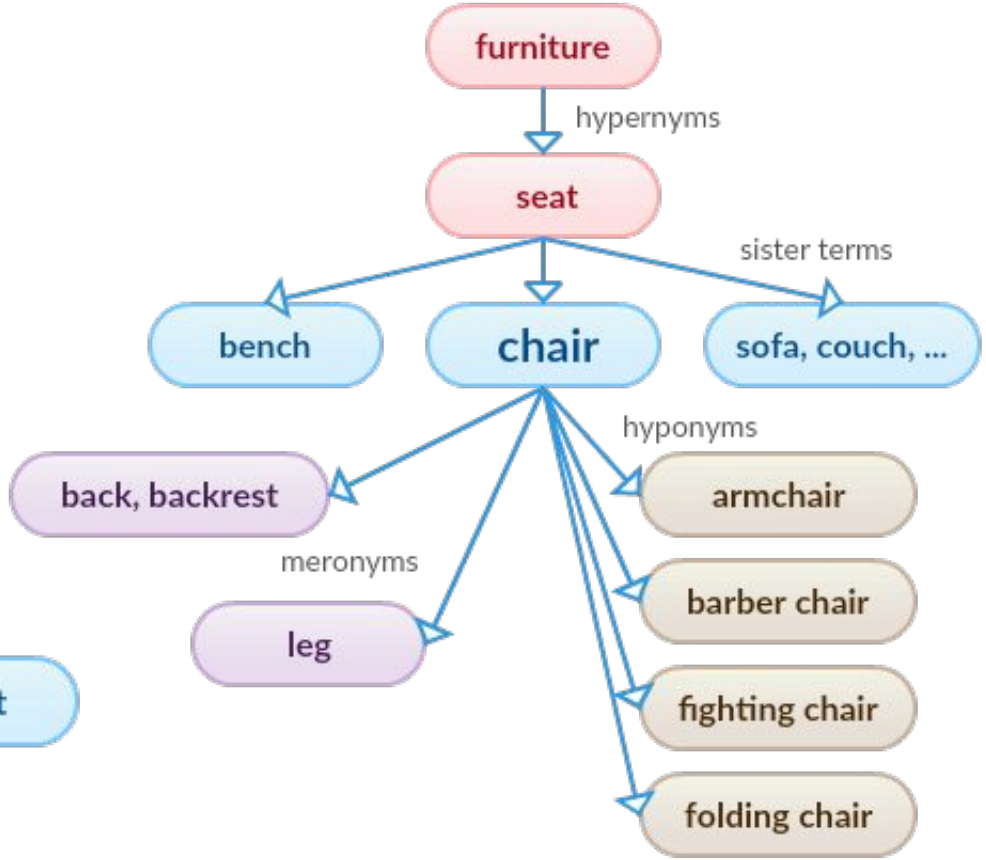
3 PERSON [countable] a person, either male or female – used especially in formal situations or in the past

4 PEOPLE [uncountable] people as a group

Example from www.ldoceonline.com

# Ontologies




Example of relations in WordNet

# Wikipedia, Wikidata, DBpedia

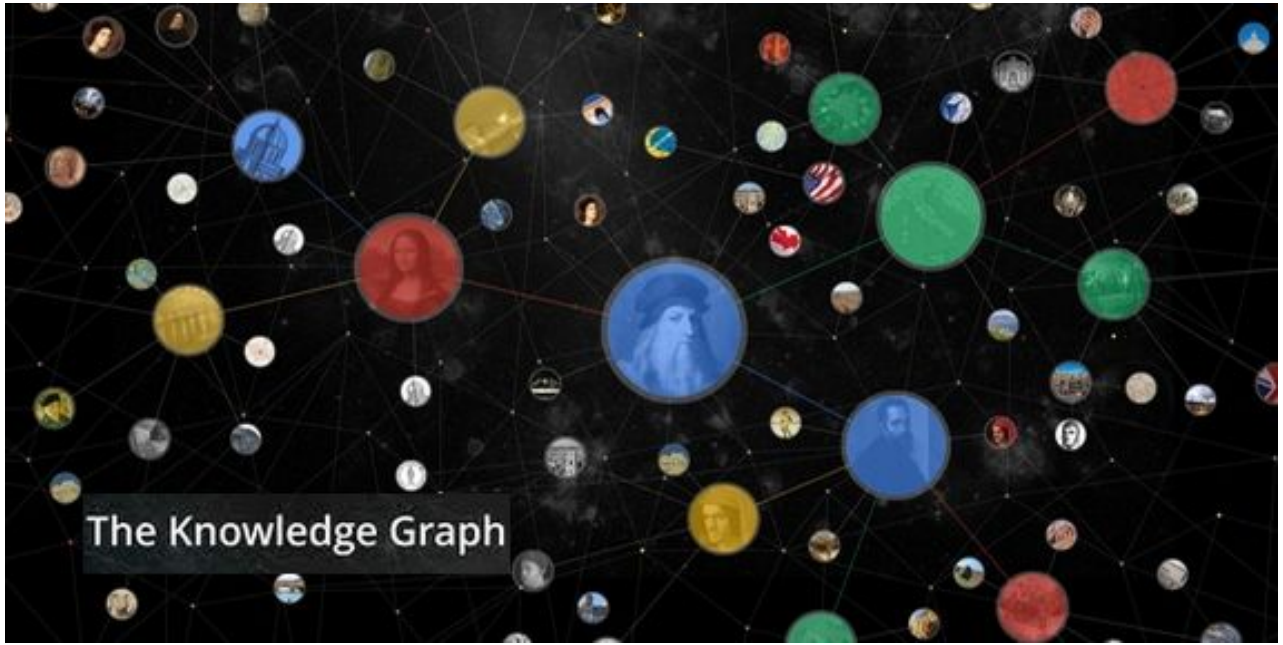
## Finance [ edit ]

- Central bank
- Mutual savings bank
- Savings bank

## Natural geography [ edit ]

- Bank (geography), a raised portion of seabed or sloping ground along the edge of a stream, river, or lake
- Ocean bank (topography)
- Ocean bank, a shallow area in a body of water
- Stream bank or riverbank, a terrain alongside the bed of a river, creek, or stream

# Knowledge Graph

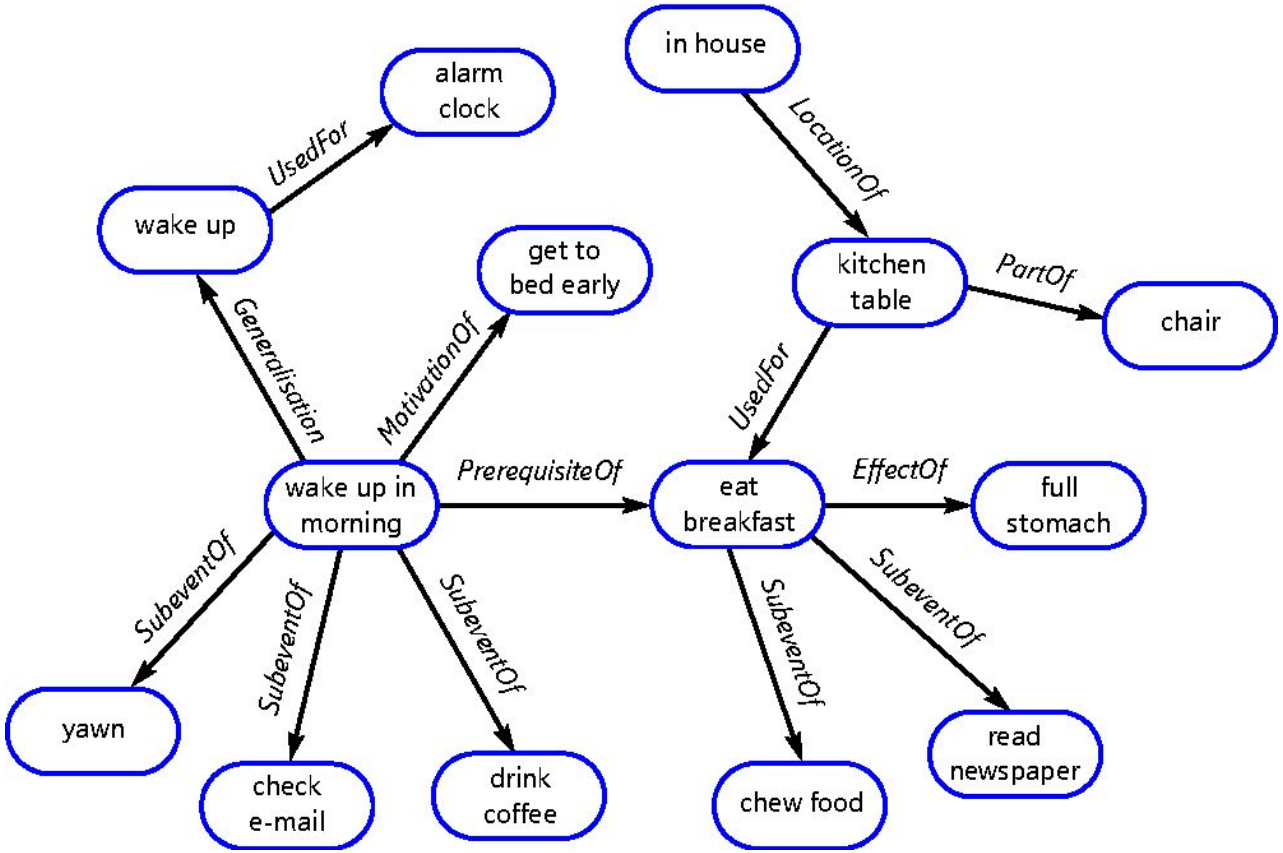



570 million entities

18 billion facts

Used in Google infoboxes, Google Assistant, and Google Home

# ConceptNet




8 mln nodes

21 mln edges

83 languages

36 relation types

Represents human knowledge about the world.

Example from conceptnet.io

# BabelNet

Noun


fan, mechanical fan, ventilator

A device for creating a current of air by movement of a surface or surfaces

ID: 00033599n | Concept


fan, rooter, sports fan

An enthusiastic devotee of sports

ID: 00033600n | Concept

15.8 mln nodes

380 mln edges

284 languages

Example from babelnet.org

# Corpora: SemCor

<wf>The</wf>

**<wf lemma="model" wnsn="3">model</wf>**

<wf lemma="quite" wnsn="1">quite</wf>

<wf lemma="plainly" wnsn="1">plainly</wf>

<wf lemma="think" wnsn="1">thought</wf>

<wf lemma="person" wnsn="1">Michelangelo</wf>

<wf lemma="crazy" wnsn="1">crazy</wf>

<wf>;</wf>

http://web.eecs.umich.edu/~mihalcea/downloads.html

# Corpora: Wikipedia

Beverly Johnson (born October 13, 1952) is an **[American|"United States"]** **[model|"Model (person)"]**, **[actress|"Actress"]**, **[singer|"Singer"]**, and **[businesswoman|"Businesswoman"]**.

https://en.wikipedia.org/wiki/Beverly_Johnson

# How to determine the sense?

*You shall know a word by the company it keeps.*

John Rupert Firth, 1957

# 1. Lesk

With which sense **signature** does your **context** overlap the most?

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

  best-sense ← most frequent sense for word
  max-overlap ← 0
  context ← set of words in sentence
  for each sense in senses of word do
    signature ← set of words in the gloss and examples of sense
    overlap ← COMPUTEOVERLAP(signature, context)
    if overlap > max-overlap then
        max-overlap ← overlap
        best-sense ← sense
  end
  return(best-sense)
```

# Lesk

*Simon works at an industrial **plant** as an engineer.*

- S: (n) **plant**, works, industrial plant (buildings for carrying on industrial labor) *"they built a large plant to manufacture automobiles"*
- S: (n) **plant**, flora, plant life ((botany) a living organism lacking the power of locomotion)
- S: (n) **plant** (an actor situated in the audience whose acting is rehearsed but seems spontaneous to the audience)
- S: (n) **plant** (something planted secretly for discovery by another) *"the police used a plant to trick the thieves"; "he claimed that the evidence against him was a plant"*

Example from WordNet

# Lesk

- for context, use lemmas and filter stopwords
- for *signature* of each sense, use
  - examples
  - definitions
  - related terms
  - synonyms, hyponyms, hypernyms, holonyms, meronyms...
  - sentences from corpora, etc.

# Lesk

How to compute overlap?

- number of overlapping words
- weighed by the number of occurrences
- weighed by –log(P(w))
- weighed by IDF score: log(C(docs) / C(docs with word *i* ))
- …

# Important linguistic hypothesis

One sense per discourse!

*I bought a **plant** yesterday and put it in my small tank with some inch long baby cichlids.Lost 3 fish over night i never lose fish. i dont see any nibbles on the **plant** though.. any advice?*

# Results

Pros:

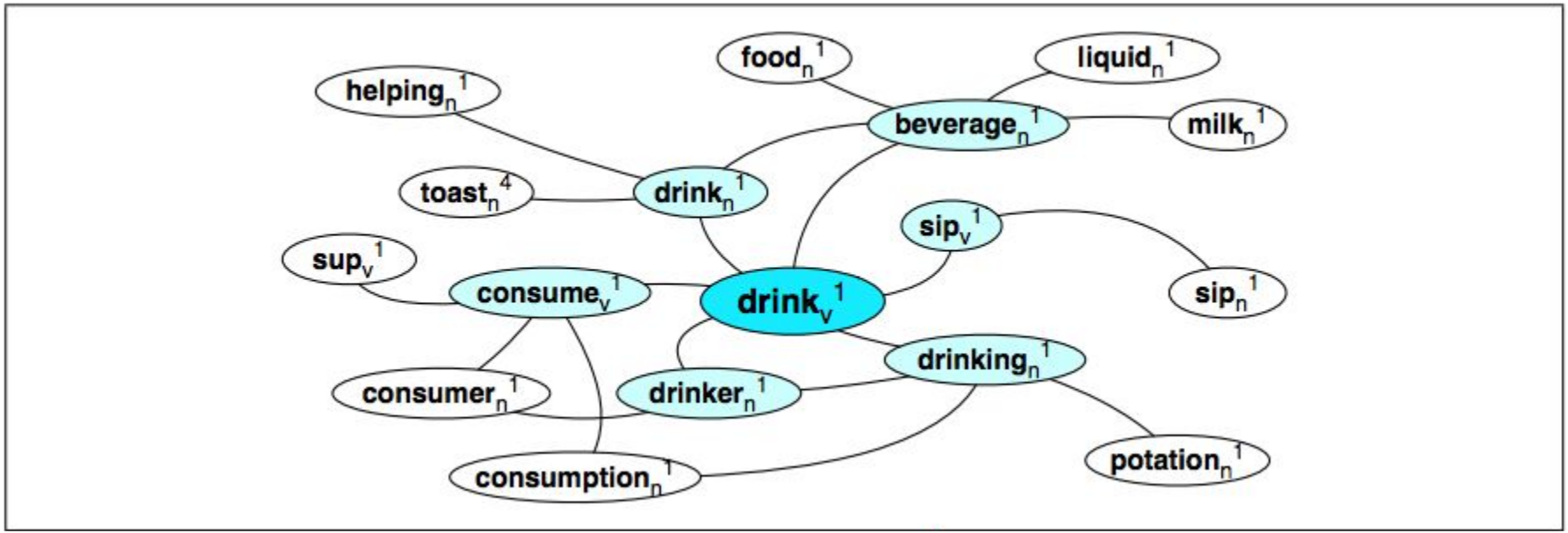
- good for partially annotating corpora
  - can be continued in a semi-supervised fashion
- links to the KB can be preserved
- unreasonably effective: The Unreasonable Effectiveness of Counting Words Near Other Words (2017)

Cons:

- some senses will be poorly covered

# 2. Graph-Based

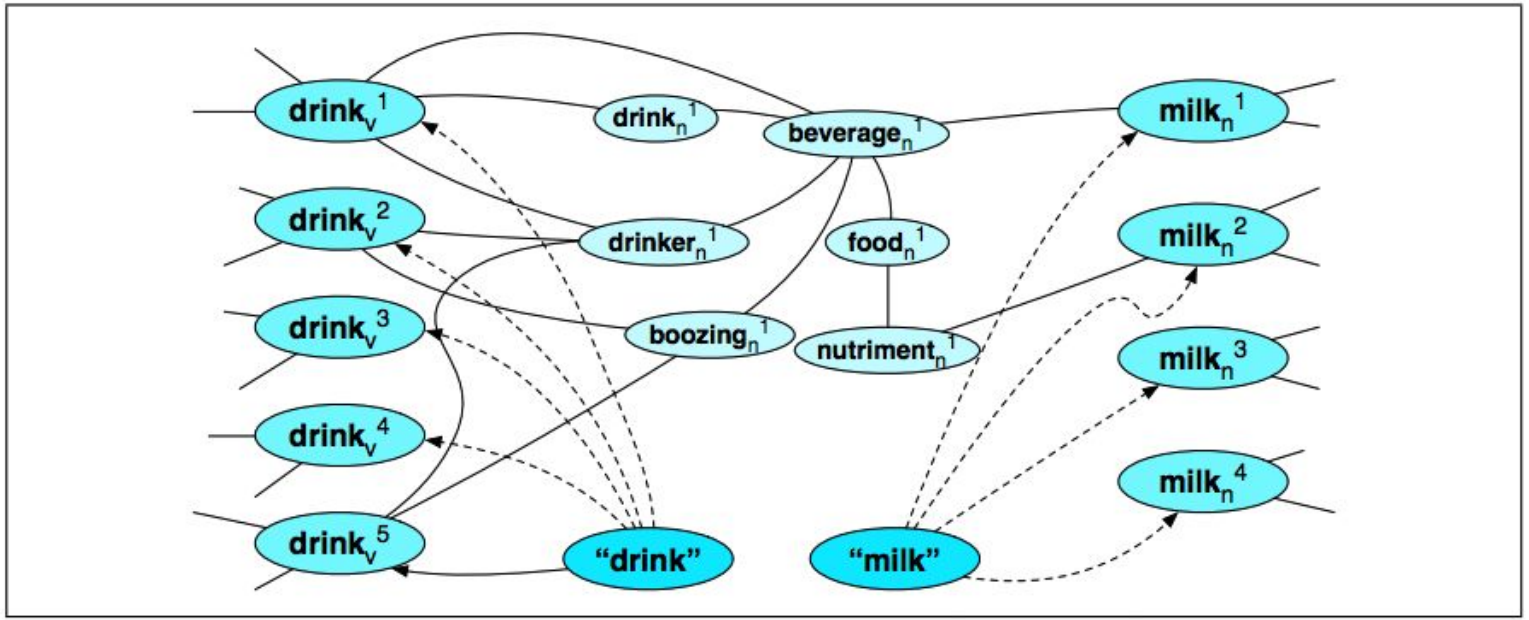
Which sense is the closest to context words?




Example from Navigli and Lapata (2010)

# Graph-Based

Which sense of the context word to choose?




Example from Navigli and Lapata (2010)

# Path similarity

*Simon works at an industrial **plant** as an engineer.*

```
>>> plant_1 = wn.synset('plant.n.01')
>>> plant_1.definition()
u'buildings for carrying on industrial labor'

>>> plant_2 = wn.synset('plant.n.02')
>>> plant_2.definition()
u'(botany) a living organism lacking the power of locomotion'

>>> engineer = wn.synset('engineer.n.01')
```

# Path similarity

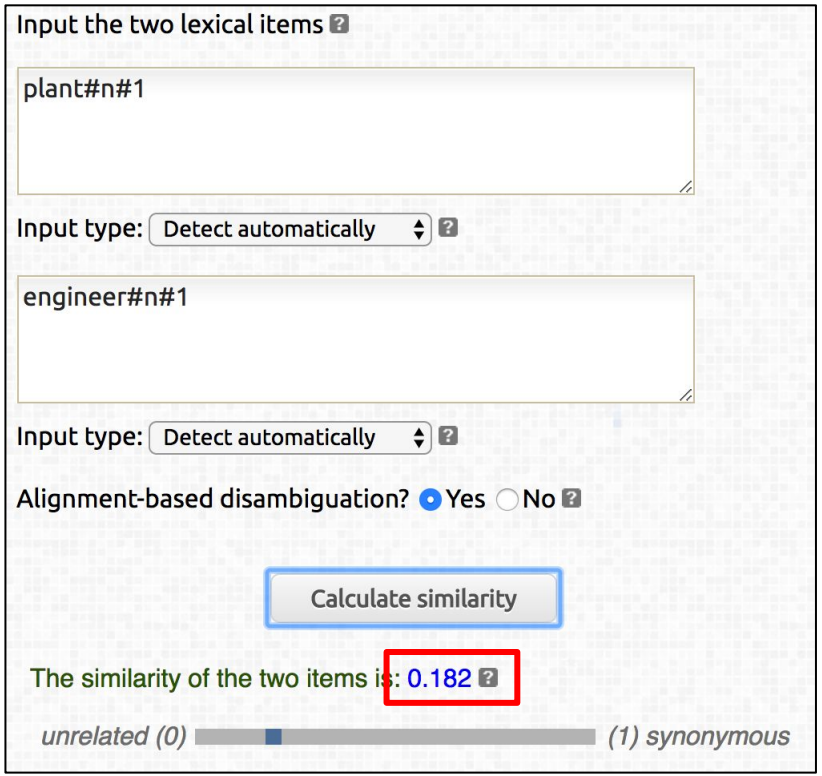
*Simon works at an industrial **plant** as an engineer.*

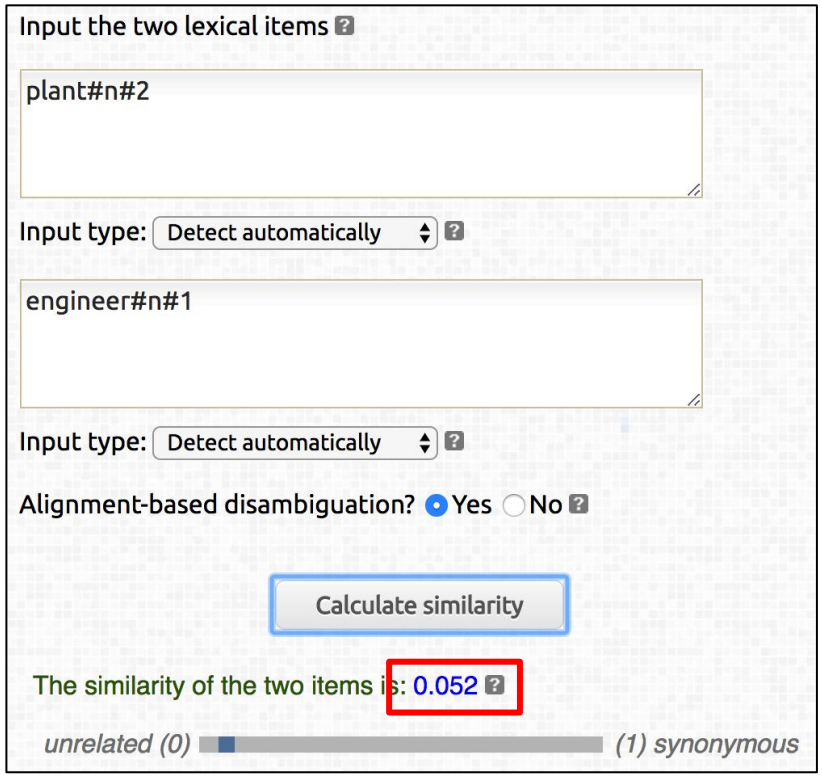
```
>>> plant_1 = wn.synset('plant.n.01')
>>> plant_1.definition()
u'buildings for carrying on industrial labor'

>>> plant_2 = wn.synset('plant.n.02')
>>> plant_2.definition()
u'(botany) a living organism lacking the power of locomotion'

>>> engineer = wn.synset('engineer.n.01')
>>> plant_1.path_similarity(engineer)
0.1111111111111111
>>> plant_2.path_similarity(engineer)
0.25
```

# Align, Disambiguate, and Walk




Demo: http://lcl.uniroma1.it/adw/

# Babelfy

I need to buy a big plant for my mom .

**need**
Have need of

**buy**
Obtain by purchase; acquire by means of a financial transaction

**flora**
(botany) a living organism lacking the power of locomotion

**mommy**
Informal terms for a mother



Example from babelfy.org

# Babelfy algorithm

- *[done once]* create semantic signatures for BabelNet concepts
  - assign weights to edges; higher weights in more densely connected areas

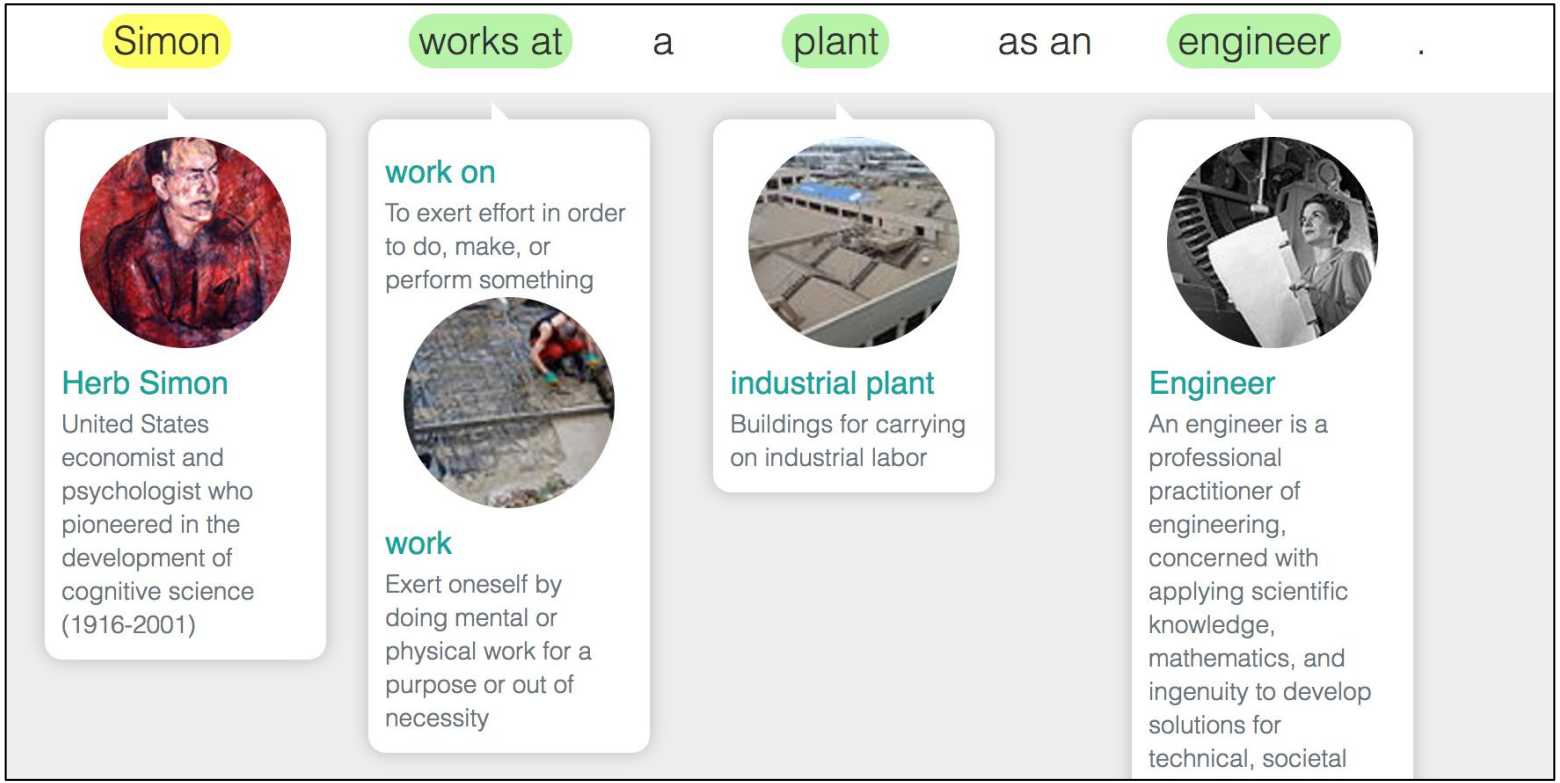
# Babelfy algorithm

- *[done once]* create semantic signatures for BabelNet concepts
  - assign weights to edges; higher weights in more densely connected areas
- extract all linkable fragments from the text
- list possible meanings for the fragments
- link the possible meanings of the fragments

# Babelfy algorithm

- *[done once]* create semantic signatures for BabelNet concepts
  - assign weights to edges; higher weights in more densely connected areas
- extract all linkable fragments from the text
- list possible meanings for the fragments
- link the possible meanings of the fragments
- extract a dense subgraph of this representation
  - connect meanings if they are in each other's signature
- select the best candidate meaning for each fragment

# Babelfy






Example from babelfy.org

# Babelfy

The teacher and the pupils entered the classroom .

**teacher**
A person whose occupation is teaching

**pupil**
The contractile aperture in the center of the iris of the eye; resembles a large black dot

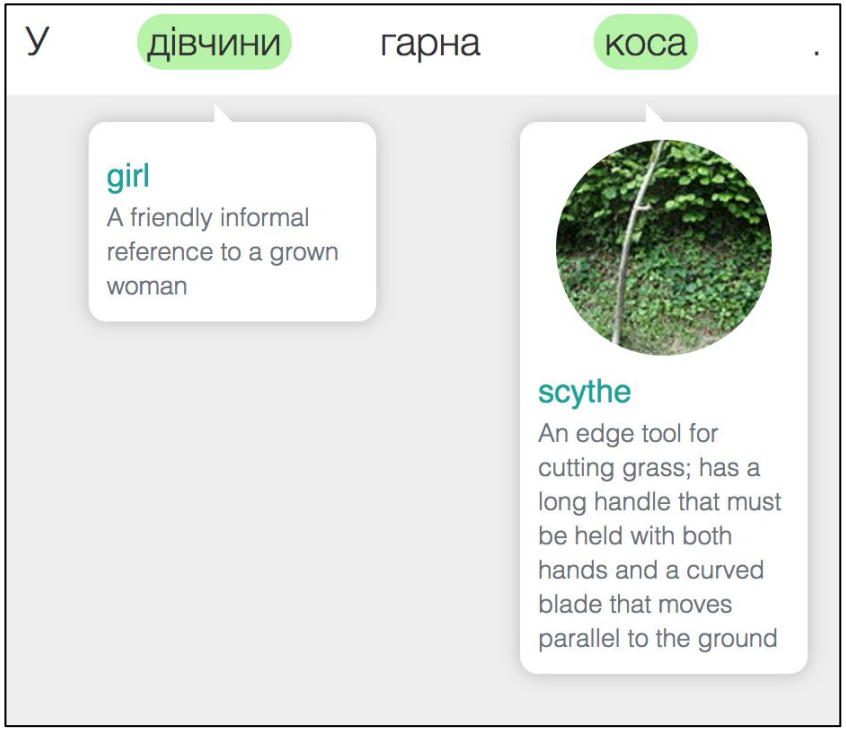
**enroll**
Register formally as a participant or member

**classroom**
A room in a school where lessons take place



Example from babelfy.org

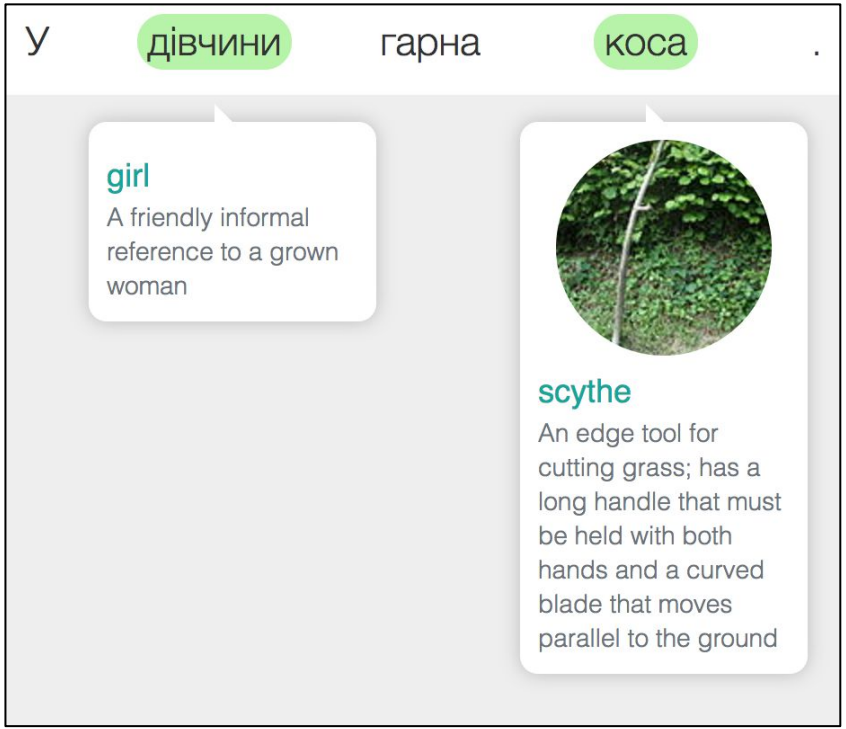
# Babelfy

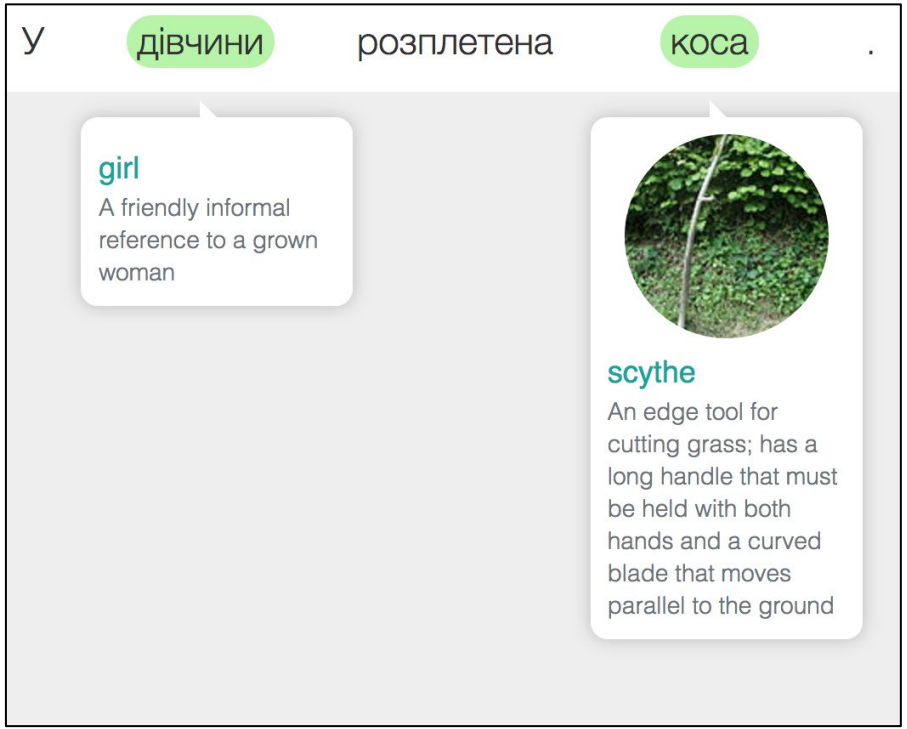





Example from babelfy.org

# Babelfy

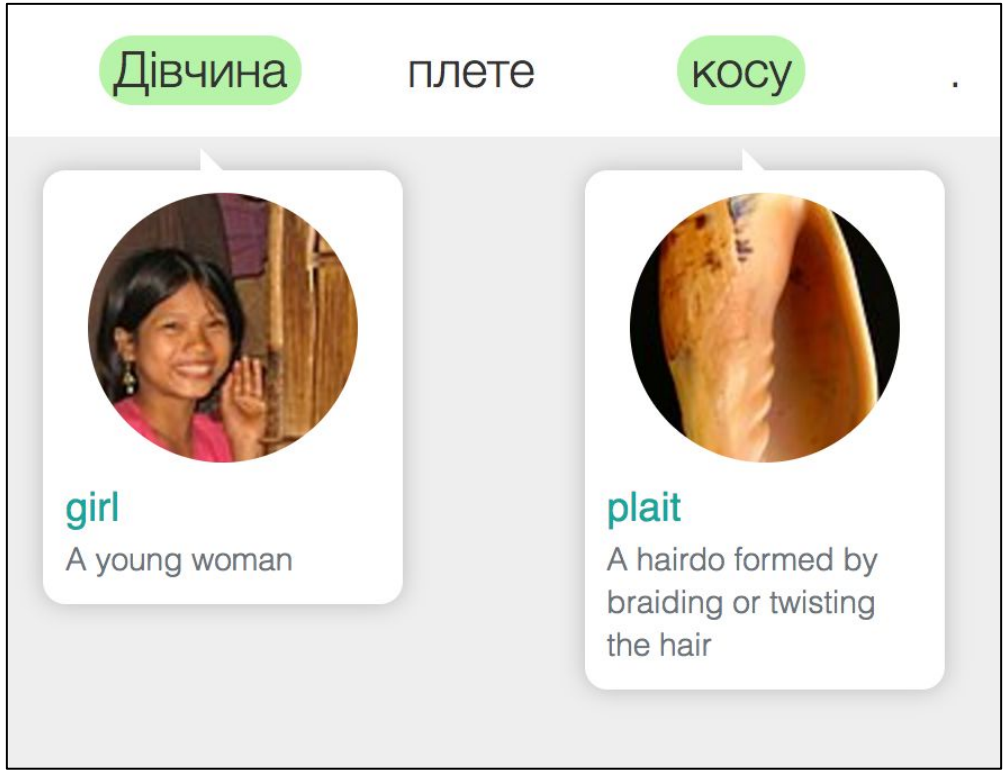





Example from babelfy.org

# Babelfy





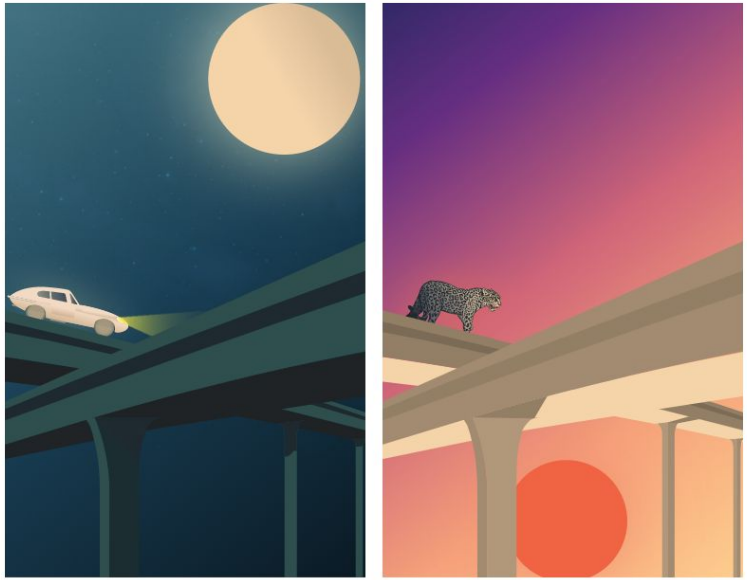

Example from babelfy.org

# 3. Classification

- For a specific word
  - *ngrams, syngrams, morphological features*
  - *overlap of current context with a predefined set of content words (extracted from a dictionary)*

- For a category

# OpenAI: Type-based Neural Entity Disambiguation



The man saw a **Jaguar** speed on the highway.

Jaguar Cars 0.70

jaguar 0.12

The prey saw the **jaguar** cross the jungle.

Jaguar Cars 0.03

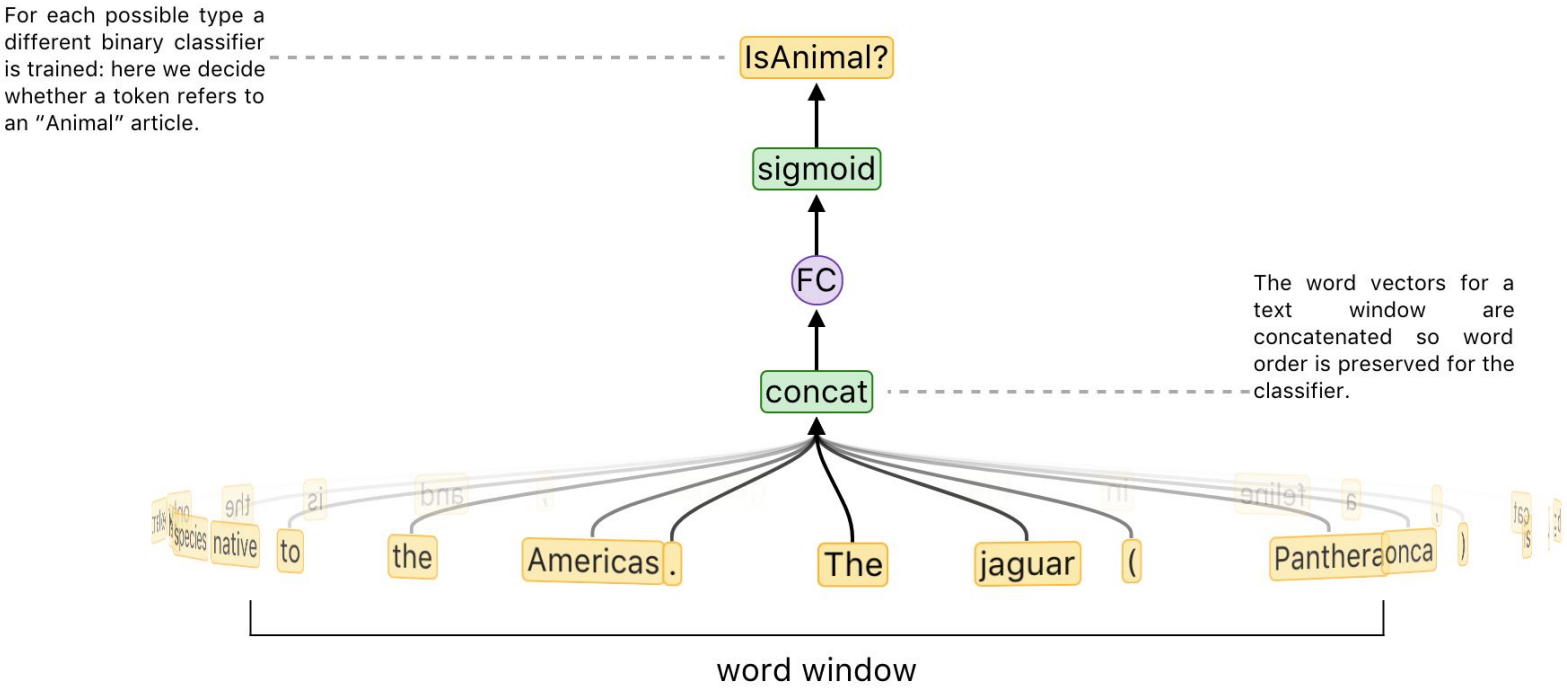
jaguar 0.89

Example from blog.openai.com

# OpenAI: Type-based Neural Entity Disambiguation

- Extract entities from Wikipedia
- Extract the set of categories for each entity from Wikipedia
- Pick a list of ~100 categories to be your “type” system
- Generate train data: word contexts mapped to 100-dimensional vectors
- Train: 400M tokens, bidirectional LSTM, F1 - 0.91

# OpenAI: Type-based Neural Entity Disambiguation






Example from blog.openai.com

# 4. Word sense induction

Idea:

- for each word occurrence, compute a context vector
- cluster these context vectors
- map clusters to senses

Problem: how many clusters is enough?

# Evaluation

- Intrinsic:
  - word similarity
  - word relatedness
  - analogy relations
  - synonym selection
- External:
  - sentiment analysis
  - textual entailment
  - question answering

# 2.

# Semantic role labeling

# Who did what to whom

*The police officer* *detained* *the suspect* *at the crime scene*.

**detainer** **detainee** **location**

*The suspect* *was* *detained* *at the crime scene* *by* *the police officer*.

**detainee** **location** **detainer**

*This is* *the police officer* *who* *detained* *the suspect* *at the crime scene*.

**detainer** **detainee** **location**

# Who did what to whom

- Causer (agent/force)
- Instrument
- Result
- Patient
- Theme
- Source, path, goal/recipient, location
- Experiencer
- Stimulus
- Beneficiary
- Time, manner, reason...

# PropBank

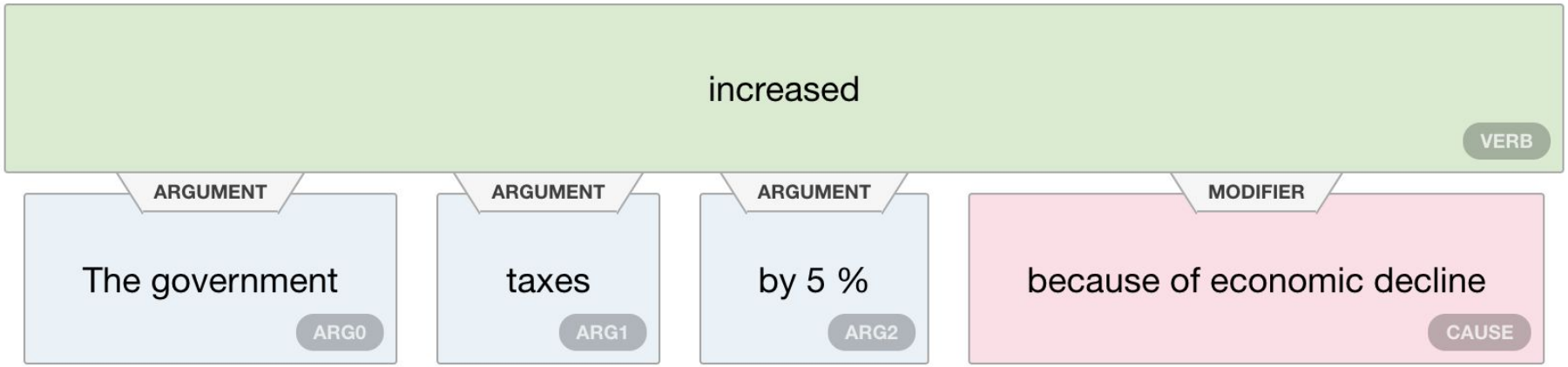
- *increase.01 “go up incrementally”*
  - *Arg0: causer of increase*
  - *Arg1: thing increasing*
  - *Arg2: amount increased by*
  - *Arg3: start point*
  - *Arg4: end point*

| | |
|---|---|
| **TMP** | when? |
| **LOC** | where? |
| **DIR** | where to/from? |
| **MNR** | how? |
| **PRP/CAU** | why? |
| **REC** | |
| **ADV** | miscellaneous |

- ***The government*** *increased* ***taxes*** *by* ***5%****.*
- ***Taxes*** *increased.*

# Semantic role labelling: AllenNLP

The government increased taxes by 5 % because of economic decline .




Parsed by https://demo.allennlp.org/

# FrameNet

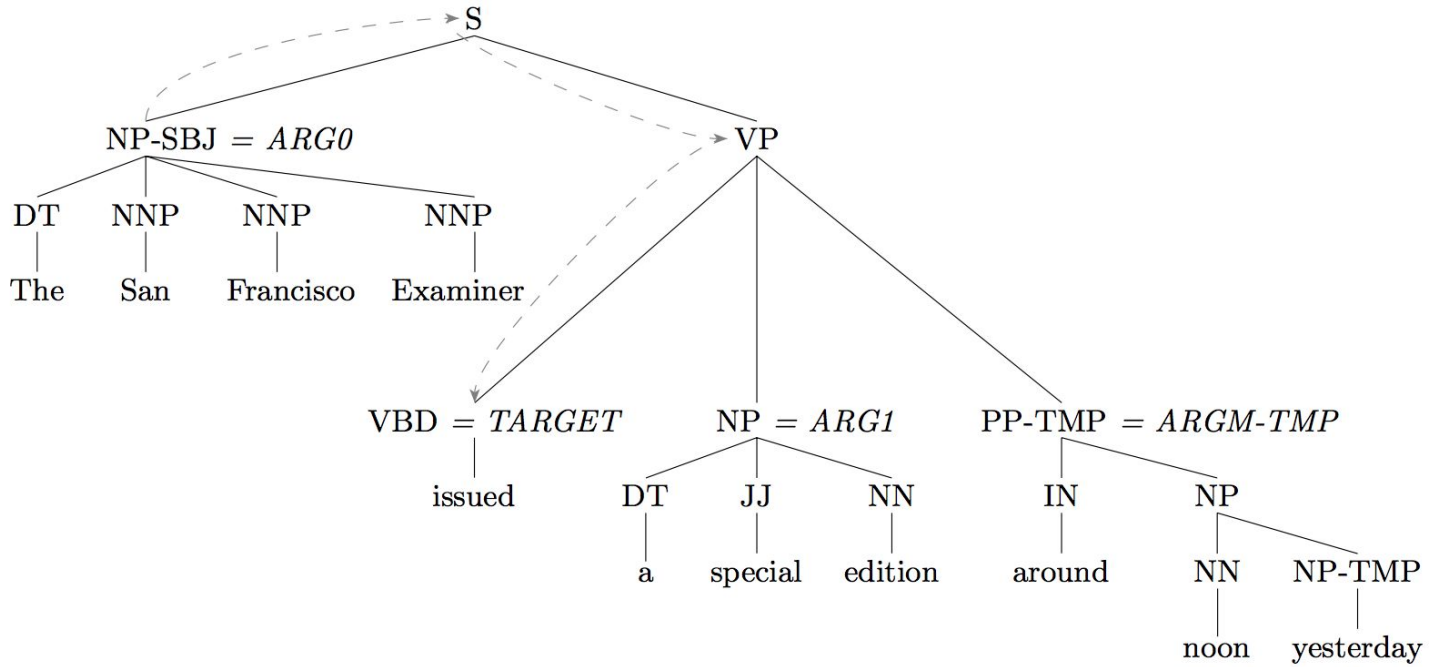
- Abandonment**:**
  - *abandon, abandoned, abandonment, leave, forget*
- An ***Agent*** leaves behind a ***Theme*** effectively rendering it no longer within their control or as one's property...
- examples:
  - ***Carolyn*** *abandoned* ***her car*** *and jumped on a bus.*
  - *Abandonment of* ***a child*** *is considered to be a serious crime in many jurisdictions.*
  - *Perhaps* ***he*** *left* ***the key*** *in the ignition.*

# Basic SRL algorithm

- parse the sentence
- find all predicates *(mapped to PropBank or FrameNet)*
- for each predicate in the sentence
  - for each node in the parse tree
    - assign the semantic role (if any) for the predicate

*Logistic regression, SVM, Perceptron, CRF, etc.*

# Basic SRL example




What features would you use?



Image from Jurafsky, Martin (2017)

# Basic SRL features

- predicate
  - lemma, POS, active/passive voice, etc.
  - syntactic frame
    - e.g., VP → VBD NP PP
- node
  - label
  - headword lemma, headword POS, etc.
  - linear position, distance to predicate
- path from the constituent to the predicate
  - e.g., NP↑S↓VP↓VBD
- more: ngrams, dependencies, etc.

# Basic SRL notes

- Evaluation:
  - precision, recall, f-measure
- Also possible two steps:
  - identification
    - to prune unlikely constituents
    - to avoid nested constituents
  - classification
- To avoid two ARG0s:
  - return probability distribution of classes for nodes
  - do reranking

# 3.

# Textual entailment

# Textual entailment

We say that text *T* entails hypothesis *H* (i.e., *T* ⇒ *H*) if the meaning of *H* can be inferred from the meaning of *T*, *as would typically be interpreted by people*.

- **Positive TE** (text entails hypothesis):
  - *A girl swings high in the air.*
  - *A girl is on a swing.*
- **Negative TE** (text contradicts hypothesis):
  - *A girl swings high in the air.*
  - *A girl is laying in the pool.*
- **Non-TE** (text does not entail nor contradict):
  - *A girl swings high in the air.*
  - *A girl is looking for her mother.*

# Textual entailment

**Text:** The purchase of Houston-based LexCorp by BMI for $2Bn prompted widespread sell-offs by traders as they sought to minimize exposure. LexCorp had been an employee-owned concern since 2008.

**Hyp** 1**:** BMI acquired an American company.
**Hyp** 2**:** BMI bought employee-owned LexCorp for $3.4Bn.
**Hyp** 3**:** BMI is an employee-owned concern.

# Textual entailment

**Text:** The purchase of Houston-based LexCorp by BMI for $2Bn prompted widespread sell-offs by traders as they sought to minimize exposure. LexCorp had been an employee-owned concern since 2008.

**Hyp** 1**:** BMI acquired an American company.
**Hyp** 2**:** BMI bought employee-owned LexCorp for $3.4Bn.
**Hyp** 3**:** BMI is an employee-owned concern.

- NER: *LexCorp, BMI, $2Bn*
- WordNet: *purchase (n.) => purchase (v.), acquire, buy*
- Knowledge base: *Houston-based => American*
- Syntactic analysis + semantic role labeling / semantic parsing

# Textual entailment

**Text:** The purchase of Houston-based LexCorp by BMI for $2Bn prompted widespread sell-offs by traders as they sought to minimize exposure. LexCorp had been an employee-owned concern since 2008.

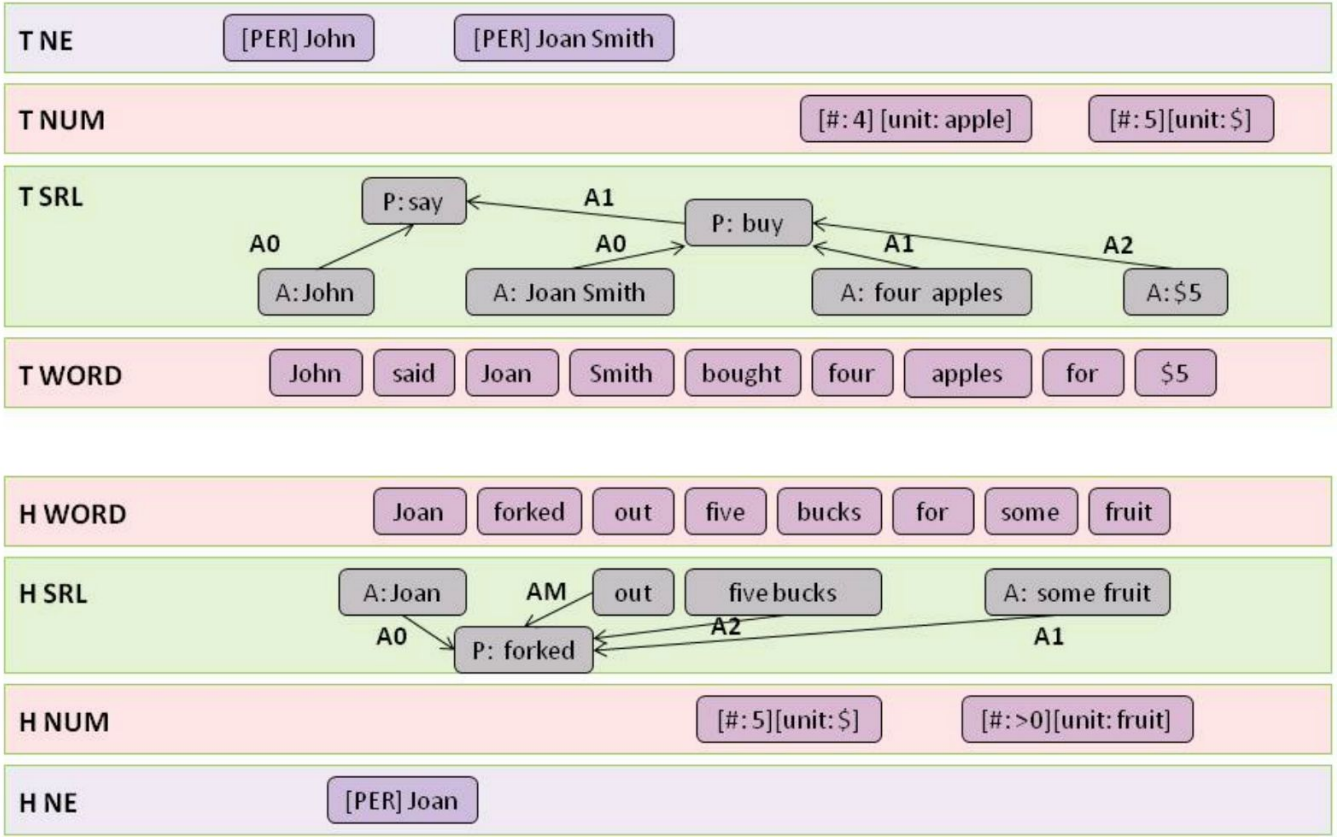
**Hyp 1:** BMI acquired an American company.
**Hyp 2:** BMI bought employee-owned LexCorp for $3.4Bn.
**Hyp 3:** BMI is an employee-owned concern.

- Normalization of quantities: *$2Bn => 2,000,000,000*
- Fuzzy match of NE: *BMI => Business Machine Inc.*
- Coreference resolution
- Scope of negation

# Textual entailment pipeline

- Text and hypothesis preprocessing
- Align syntactic/semantic representations of *T* and *H*
  - only a small portion of *T* is relevant to *H*
- Features: compare the constituents
  - match of NEs and numerical measures
  - lexico-semantic relations (synonyms, antonyms, hypernyms, meronyms, entailment)
  - word similarity
  - noun-verb relations (*“is a winner” => “won”*)
  - idioms (*“kick the bucket” => “die”*)
  - paraphrases (*“has been unable to” => “could not”*)
  - syntactic mapping (active => passive)

# Textual entailment pipeline



# Textual entailment: TEASE

- Extract entailment paraphrases from web using a dependency parser
  - both lexical and grammatical
- Apply iteratively to transform *T* to *H*

| | | |
|---|---|---|
| *X* write *Y* | *X* who write *Y* | *X* produce *Y* |
| | *X* publish *Y* | *X* pen *Y* |
| | *X* compose *Y* | *X* create *Y* |
| | read *Y* by *X* | *X*'s *Y* |
| | *Y* attributed to *X* | *X* complete *Y* |
| | perform *Y* by *X* | *X* book of *Y* |
| | *X* writer of *Y* | *X* say in *Y* |
| | selected *Y* of *X* | *X* work include *Y* |

# 4.

# Semantic parsing

# Meaning representation formalisms

- abstract meaning representation
- typed lambda-calculus expressions
- combinatory categorial grammar
- minimal recursion semantics
- universal conceptual cognitive annotation
- … (a hundred more formalisms)

# Meaning representation formalisms

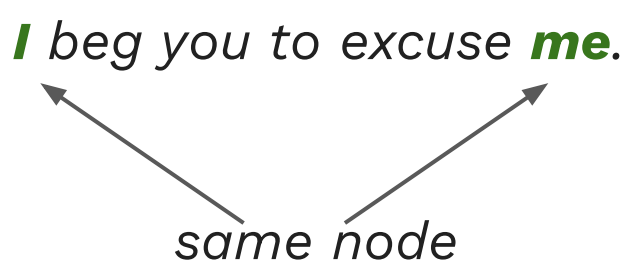
- abstract meaning representation
- typed lambda-calculus expressions
- combinatory categorial grammar
- minimal recursion semantics
- universal conceptual cognitive annotation
- … (a hundred more formalisms)

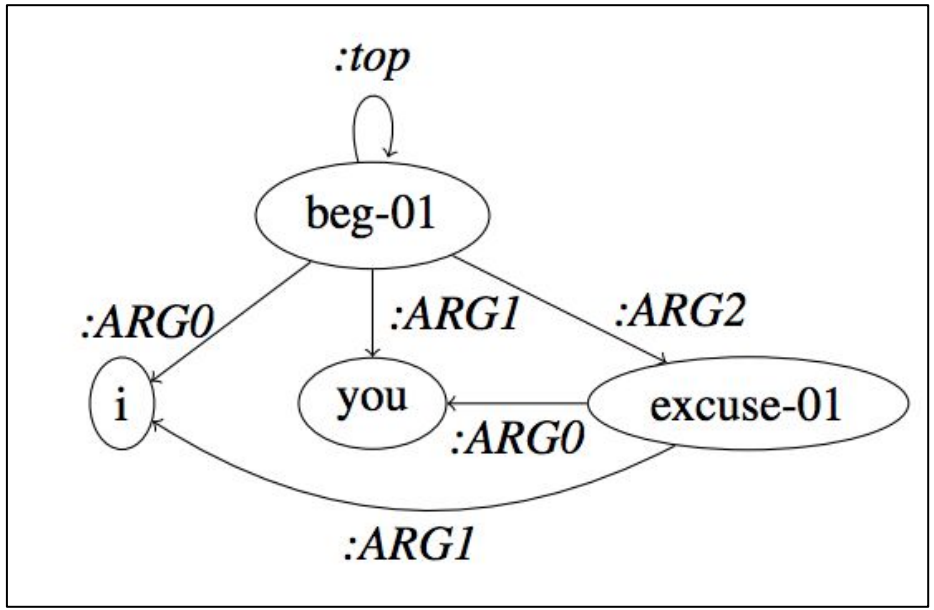
# Abstract meaning representation

# What is AMR?

Sentence => a *rooted, directed, acyclic* graph, where:

- nodes are concepts
- edges are semantic relations
- function words are omitted



# AMR nodes

AMR nodes are concepts that can be:

- a PropBank or FrameNet frame (“beg-01”)
- a word (“you”, “boy”)
- a special keyword (“person”, “organization”, “date-entity”, “volume-quantity”, “temporal-quantity”, etc.)

# AMR pros and cons

AMR handles:

- semantic roles
- entity types
- coreference
- modality
- polarity
- wikification

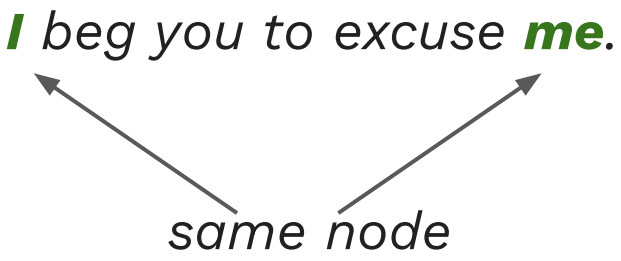
AMR doesn’t handle:

- tense
- definiteness
- plurality

# AMR notation

AMR uses PENMAN notation, where each node is assigned a variable:

```
(b / beg-01
    :ARG0 (i / i)
    :ARG1 (y / you)
    :ARG2 (e / excuse-01
        :ARG0 y
        :ARG1 i))
```



# AMR examples: frames

*The boy desires the girl to believe him.*
*The boy wants the girl to believe him.*
*The boy desires to be believed by the girl.*
*The boy has a desire to be believed by the girl.*
*The boy's desire is for the girl to believe him.*
*The boy is desirous of the girl believing him.*

```
(w / desire-01
   :ARG0 (b / boy)
   :ARG1 (b2 / believe-01
              :ARG0 (g / girl)
              :ARG1 b))
```

# AMR examples: polarity

*The soldier was not afraid of dying.*
*The soldier was not afraid to die.*
*The soldier did not fear death.*

```
(f / fear-01
   :polarity "-"
   :ARG0 (s / soldier)
   :ARG1 (d / die-01
            :ARG1 s))
```

# AMR examples: modality

*The boy must not go.*
*It is obligatory that the boy not go.*

```
(o / obligate-01
   :ARG2 (g / go-02
            :ARG0 (b / boy)
            :polarity -))
```

# AMR examples: unknown variables

*Which state borders with Kansas?*

```
(b / border-01
   :ARG0 (s / state
            :name (n / name :op1 (a / amr-unknown)))
   :ARG1 (s2 / state
            :name (n2 / name :op1 “Kansas”)))
```

# AMR examples: unknown variables

*Does Texas border with Kansas?*

```
(b / border-01
   :ARG0 (s / state
            :name (n / name :op1 “Texas”))
   :ARG1 (s2 / state
            :name (n2 / name :op1 “Kansas”))
   :polarity (a / amr-unknown))
```

*14,000 people fled their homes at the weekend after a tsunami warning was issued, the UN said on its web site.*

```
(s / say-01
   :ARG0 (o / organization
            :name (n / name :op1 "UN"))
   :ARG1 (f / flee-01
            :ARG0 (p / person :quant 14000)
            :ARG1 (h / home :poss p)
            :time (w / weekend)
            :time (a2 / after
                      :op1 (w2 / warn-01
                               :ARG1 (t / tsunami))))
   :medium (s2 / site
               :poss o
               :mod (w3 / web)))
```

# AMR data

AMR Banks in PENMAN format:

- 1.K sentences from *The Little Prince* :)
- 59K sentences of newswire, discussion forum and other web logs, television transcripts
- 7K sentences of Bio AMR (cancer-related PubMed articles)

# *The Little Prince* in AMR

*You become responsible , forever , for what you have tamed .*

```
(b / become-01
   :ARG1 (y / you)
   :ARG2 (r / responsible-03
            :ARG0 y
            :ARG1 (t2 / thing
                      :ARG1-of (t / tame-01
                                   :ARG0 y))
            :extent (f / forever)))
```

# AMR parsing

Parsing algorithms:

- graph-based
- transition-based
- ~~rule-based~~
- ~~seq2seq-based~~

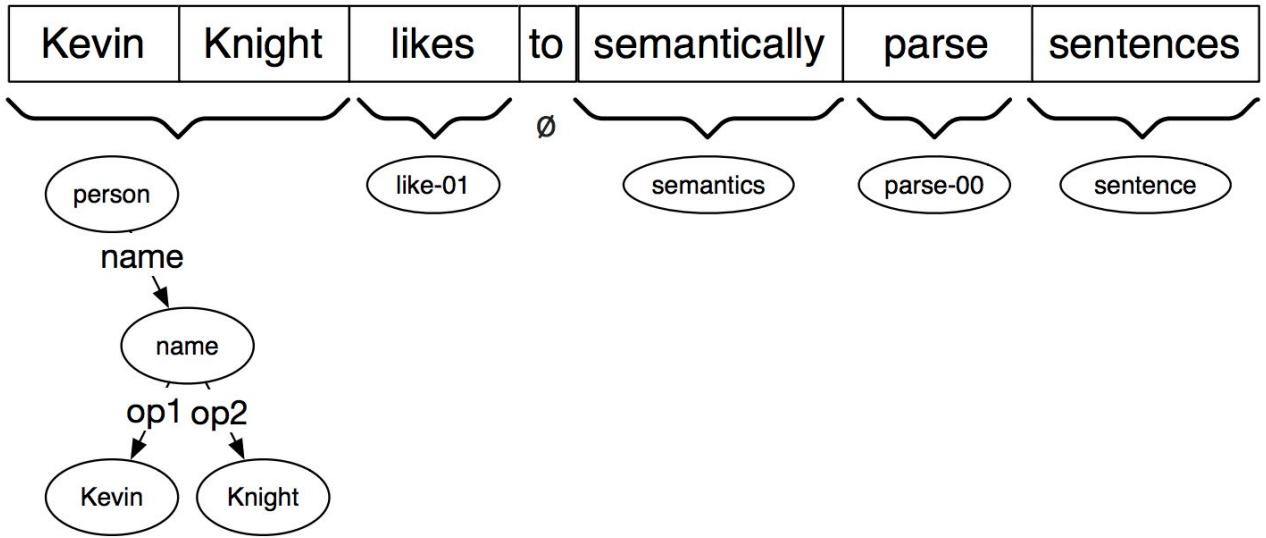
What is needed:

- POS tagging
- NER
- syntactic parsing
- coreference resolution

# Graph-based AMR parsing

JAMR parser:

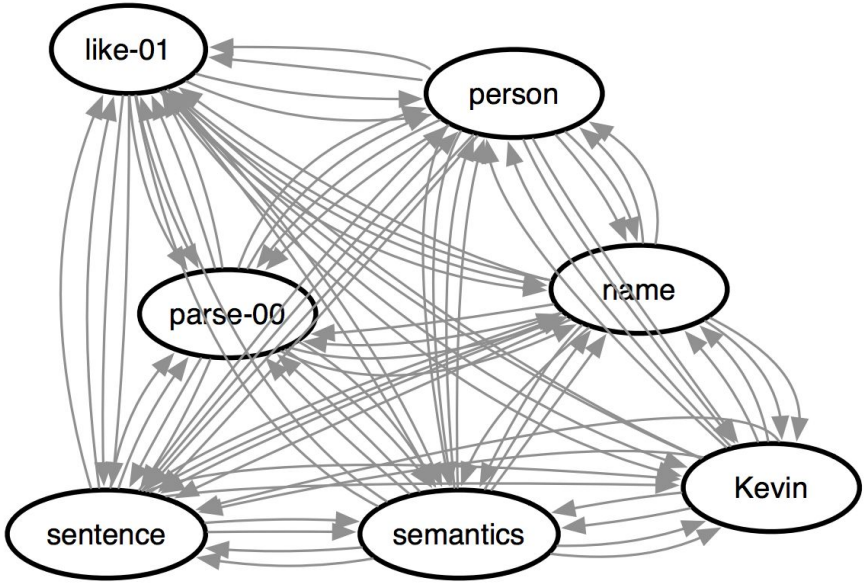
- identify concepts
- identify relations between concepts



# Graph-based AMR parsing

JAMR parser:

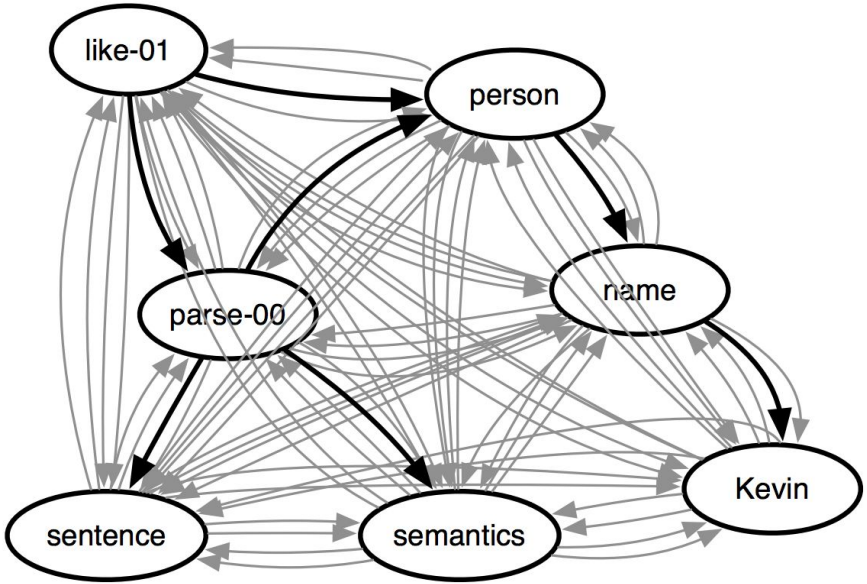
- identify concepts
- identify relations between concepts



# Graph-based AMR parsing

JAMR parser:

- identify concepts
- identify relations between concepts

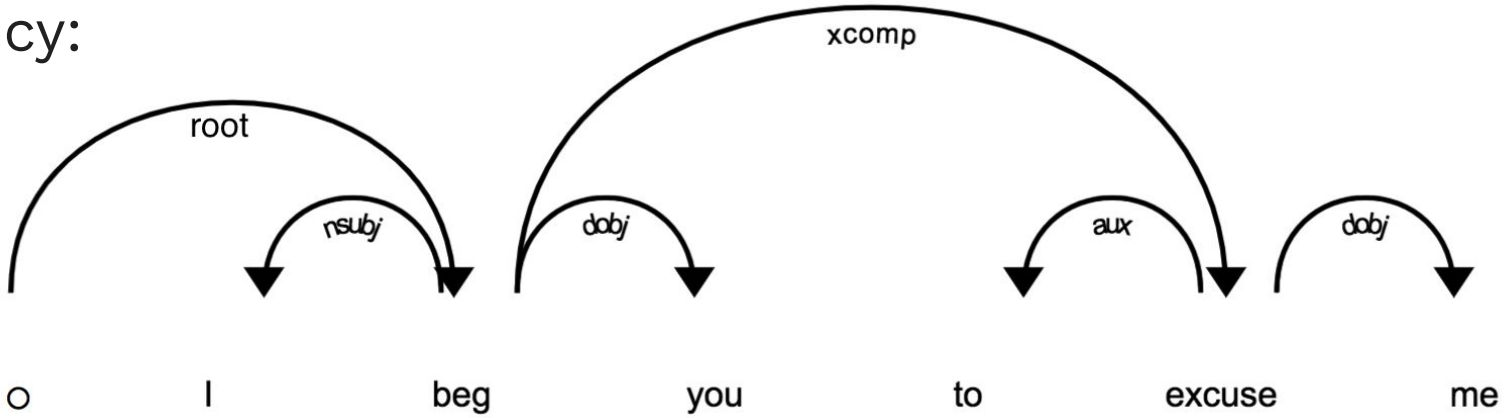


# Transition-based AMR parsing

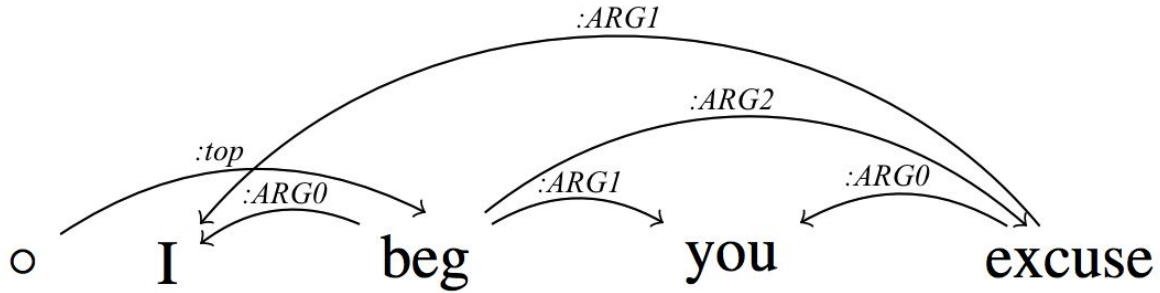
AMR graphs look similar to dependency parse trees, don’t they?

# Transition-based AMR parsing

Dependency:




AMR:



# Transition-based AMR parsing

CAMR parser - transform a dependency parse tree into AMR

- traverse the dependency tree
- at each node/edge, collect features and classify action
  - merge nodes
  - swap nodes
  - delete a node
  - replace node
  - re-enter a node
  - attach edge
  - delete edge and re-attach to a new node
  - label with concept

# Transition-based AMR parsing

CAMR parser setup:

- σ - a buffer with not yet processed nodes
  - $\sigma_0$ - top of σ
- β - a buffer with not yet processed edges attached to $\sigma_0$
- G - span graph that stores the partial parses
  - initialised as a dependency tree

# Transition-based AMR parsing

Actions:

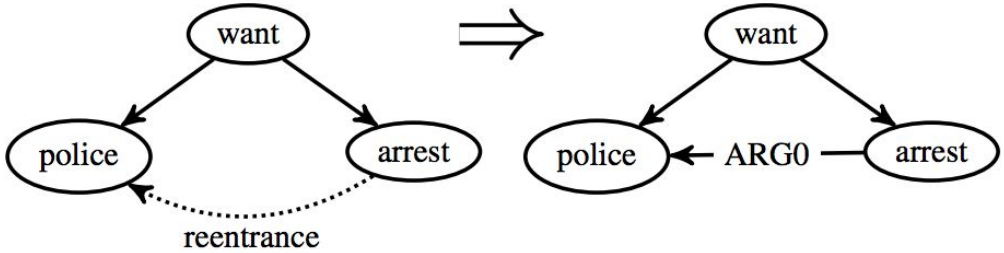
- NEXT-EDGE-$l_r$ - attach edge *(σ0, β0)* and move to next node
- SWAP-$l_r$ - swap nodes and attach with edge
- REATTACH$_k$-$l_r$ - delete edge and reattach to already processed node
- REPLACE-HEAD - replace node with another node
- REENTRANCE$_k$-$l_r$ - attach edge to already processed node
- MERGE - merge two nodes
- NEXT-NODE-$l_c$ - label with concept and move to next word
- DELETE-NODE - delete a word

# Transition-based AMR parsing

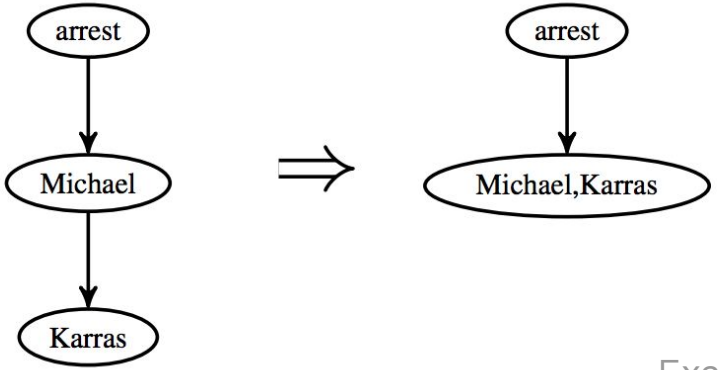
| Action | Current state ⇒ Result state | Assign labels | Precondition |
|---|---|---|---|
| NEXT EDGE-$l_r$ | $(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\sigma_0\|\sigma', \beta', G')$ | $\delta[(\sigma_0, \beta_0) \rightarrow l_r]$ | $\beta$ is not empty |
| SWAP-$l_r$ | $(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\sigma_0\|\beta_0\|\sigma', \beta', G')$ | $\delta[(\beta_0, \sigma_0) \rightarrow l_r]$ | |
| REATTACH$_k$-$l_r$ | $(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\sigma_0\|\sigma', \beta', G')$ | $\delta[(k, \beta_0) \rightarrow l_r]$ | |
| REPLACE HEAD | $(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\beta_0\|\sigma', \beta = CH(\beta_0, G'), G')$ | NONE | |
| REENTRANCE$_k$-$l_r$ | $(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\sigma_0\|\sigma', \beta_0\|\beta', G')$ | $\delta[(k, \beta_0) \rightarrow l_r]$ | |
| MERGE | $(\sigma_0\|\sigma', \beta_0\|\beta', G) \Rightarrow (\tilde{\sigma}\|\sigma', \beta', G')$ | NONE | |
| NEXT NODE-$l_c$ | $(\sigma_0\|\sigma_1\|\sigma', [], G) \Rightarrow (\sigma_1\|\sigma', \beta = CH(\sigma_1, G'), G')$ | $\gamma[\sigma_0 \rightarrow l_c]$ | $\beta$ is empty |
| DELETE NODE | $(\sigma_0\|\sigma_1\|\sigma', [], G) \Rightarrow (\sigma_1\|\sigma', \beta = CH(\sigma_1, G'), G')$ | NONE | |

# AMR: transition-based parsing

REENTRANCE$_k$-l$_r$




MERGE




Example from Wang et al (2015)

# AMR evaluation: smatch

```
(w / want-01
   :ARG0 (b / boy)
   :ARG1 (b2 / believe-01
             :ARG0 (g / girl)
             :ARG1 b))
```

```
instance(w, want-01)          /* w is an instance of wanting */
instance(b, boy)              /* b is an instance of boy */
instance(b2, believe-01)      /* b2 is an instance of believing */
instance(g, girl)             /* g is an instance of girl */
ARG0(w, b)                    /* b is the wanter in w */
ARG1(w, b2)                   /* b2 is the wantee in w */
ARG0(b2, g)                   /* g is the believer in b2 */
ARG1(b2, b)                   /* b is the believee in b2 */
```

# AMR evaluation: smatch

- F1 smatch (up to 70%)
  - graph-based > transition-based

- Speed
  - graph-based < transition-based

# Application of AMR

- natural language generation (SemEval shared task, 2017)
- information extraction (Rao et al., 2017)
- entity linking (Pan et al., 2015)
- text summarization (Liu et al., 2015; Takase et al., 2016, Dohare et al., 2017, Liao et al., 2018)
- question answering (Jurczyk and Choi, 2015)
- machine comprehension (Sachan and Xing, 2016)

# 5.

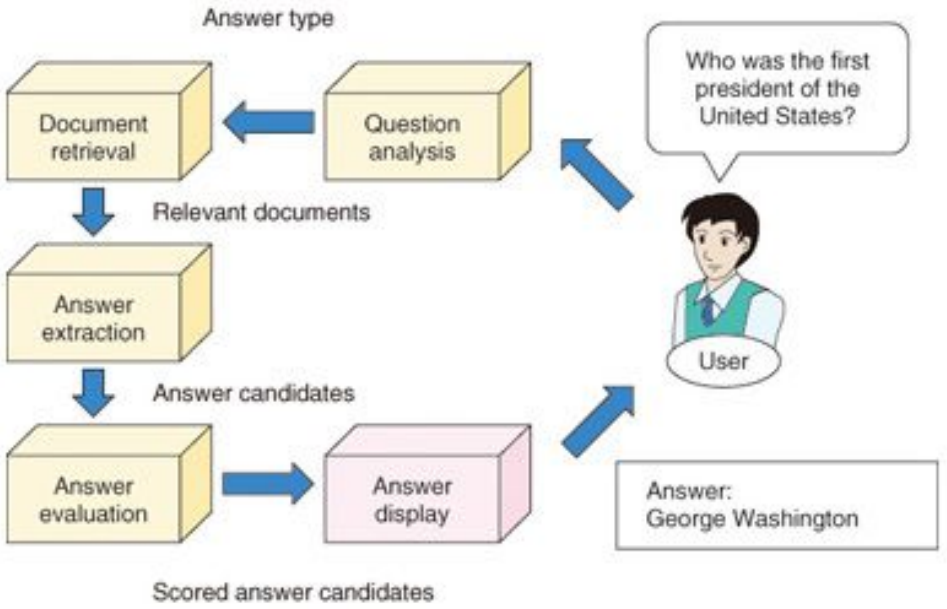
# Question Answering

# Question answering

- Open-domain
- Closed-domain

Question types:

- Factoid questions
- Meaning comprehension (+ story cloze)
- Multi-hop questions
- General reasoning



# Factoid question answering

Extension of search, but:

- naturally phrased question instead of keywords
- output is not a whole document, but just the snippet of information

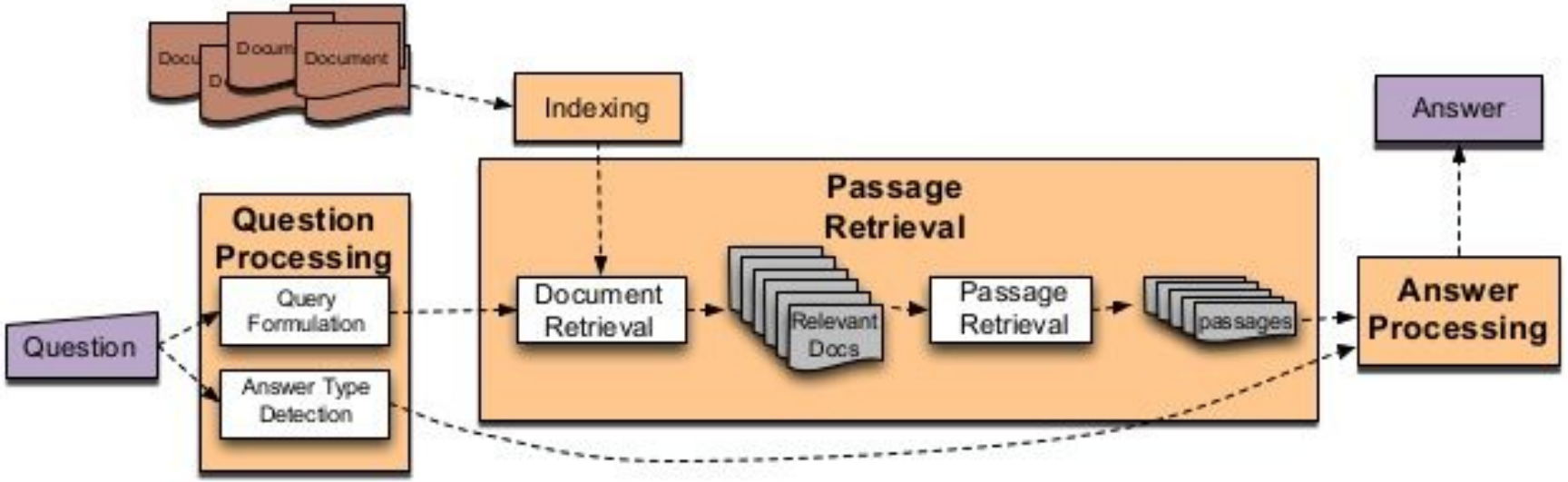
Ideal for voice:

- people ask in whole sentences
- we reply with just what the user needs to know

# Factoid QA approaches

- Information Retrieval QA
  Usage of search engines to retrieve answers and then apply filters and ranking on the recovered passage.
- Natural Language Processing QA
  Usage of linguistic intuitions and machine learning methods to extract answers from retrieved snippet.
- Knowledge Base QA
  Find answers from structured data source (a knowledge base) instead of unstructured text.
- Hybrid QA
  High performance QA systems make use of as many types of resources as possible. A hybrid approach is the combination of IR QA, NLP QA and KB QA. The main example of this paradigm is IBM Watson.

# IR-based Factoid QA



# IR-based question answering

- formulate the query
  - *e.g., process morphology, use synonyms or paraphrasing*
- find relevant documents
  - *e.g., indexed with tf-idf or using a search engine*
- rank text paragraphs from the database by relevance
  - *using the answer type*
  - *using NEs, keywords, keyword ngrams, similarity*
- do span labelling
  - *NER or patterns*
  - *ML: NEs, keywords, is appositive, novel, followed by comma…*
  - *DL: compute query vector and compare it with tokens' vectors to detect start and end of the span*

# Example questions (from TREC)

- Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"?
- What was the monetary value of the Nobel Peace Prize in 1989?
- What does the Peugeot company manufacture?
- How much did Mercury spend on advertising in 1993?
- What is the name of the managing director of Apricot Computer?
- Why did David Koresh ask the FBI for a word processor?
- What debts did Qintex group leave?
- What is the name of the rare neurological disease with symptoms such as: involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc.)?

# AskJeeves

- It largely did pattern matching to match your question to their own knowledge base of questions

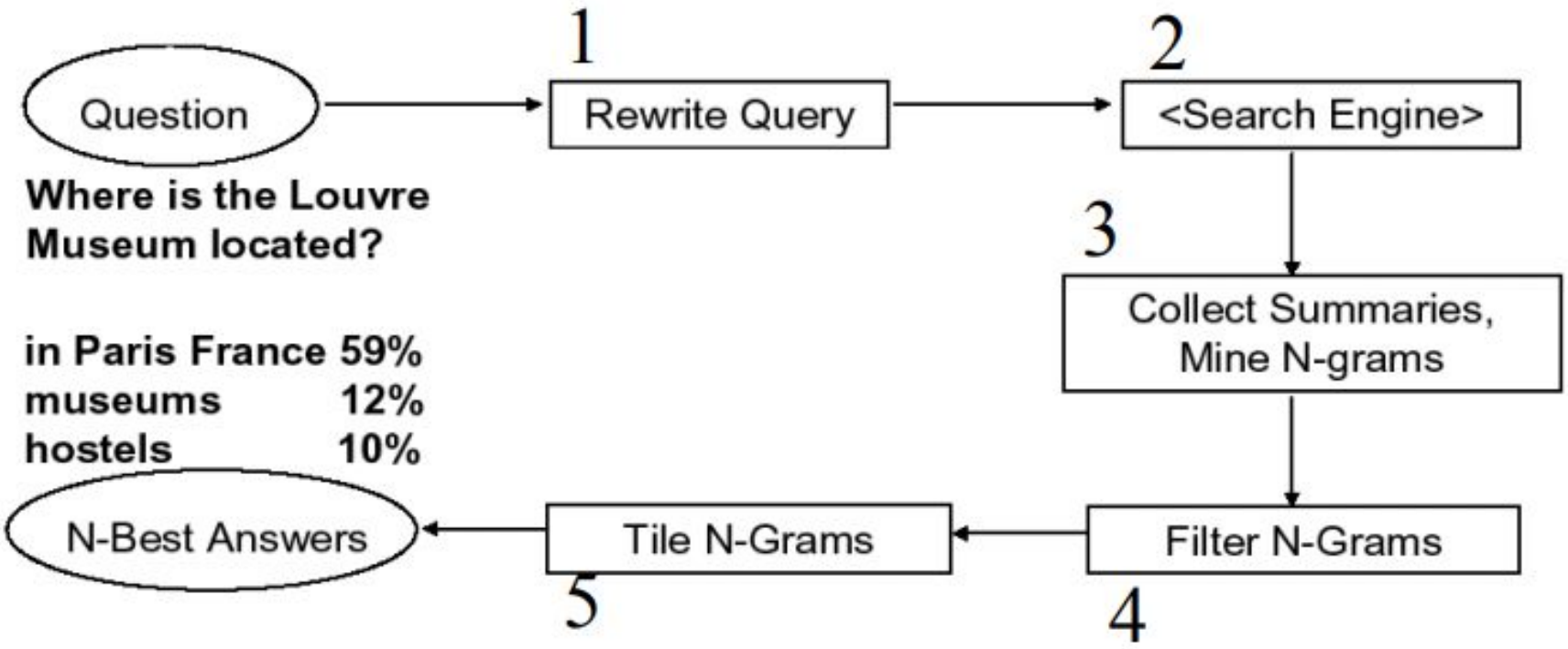
- If that works, you get the human-curated answers to that known question
- If that fails, it falls back to regular web search

# Using Pattern Matching for IR QA

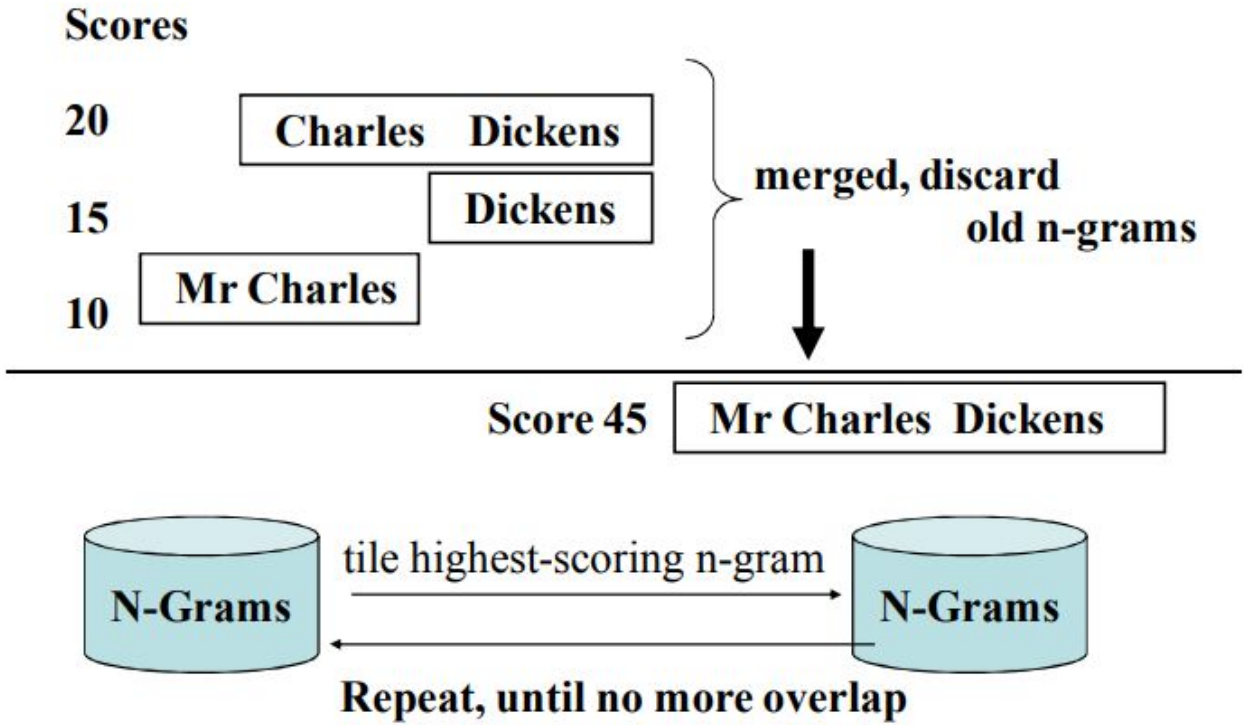
- Surface patterns:
  Question: "When was <person> born”
  Typical answers:
    - "Mozart was born in 1756.”
    - "Gandhi (1869-1948)...”
  Patterns:
    - "<NAME> was born in <BIRTHDATE>”
    - "<NAME> ( <BIRTHDATE>-”

- Use pattern learning

# Web QA: AskMSR



# AskMSR answer tiling

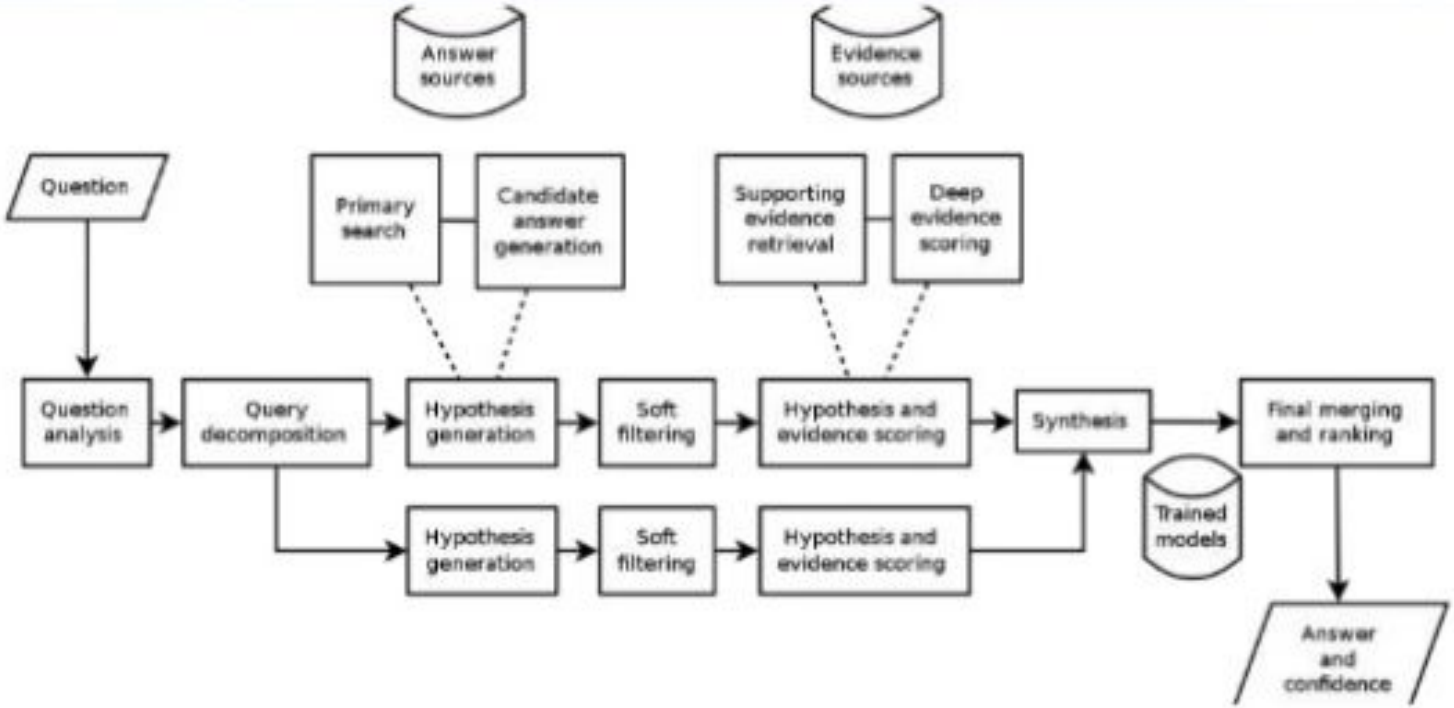


# IBM Watson

## Watson Description

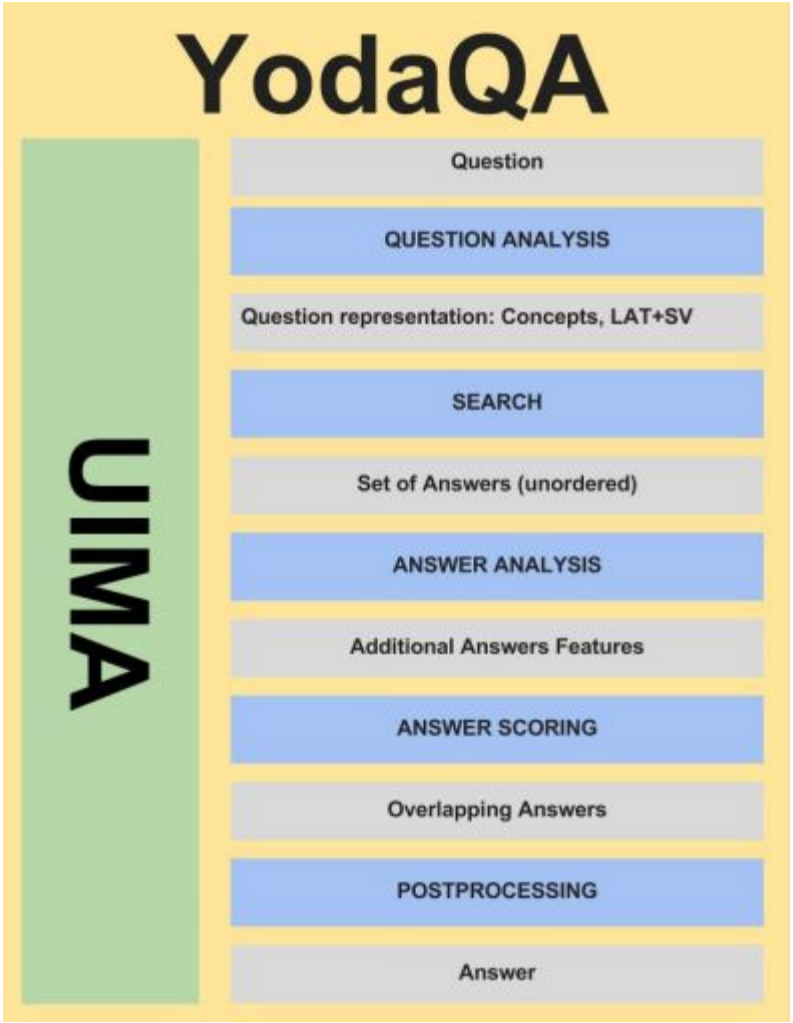
- **Hardware:** Watson system has 2,880 POWER processor threads and has 16 terabytes of RAM.
- **Data**: encyclopedias, dictionaries, thesauri, newswire articles, and literary works. Watson also used databases, taxonomies, and ontologies such as DBPedia, WordNet, and Yago were used.
- **Software**: DeepQA software and the Apache UIMA framework. The system was written in various languages, including Java, C++, and Prolog, and runs on the SUSE Linux Enterprise Server 11 operating system using Apache Hadoop framework to provide distributed computing.

# The high level architecture of DeepQA



# YodaQA

- Question Analysis
- Database Search
- Generate many candidate answers
- Answer Typing & Scoring



# YodaQA NLP Analysis

Output: linguistic representation of the question

- OpenNLP segmenter
- StanfordParser (pos + dependencies + constituents)
- LanguageTool lemmatizer
- OpenNLP NER off-the-shelf models: person, place, date, ...

# YodaQA Lexical Answer Type (LAT)

- Focus identification
- Focus name proxy
- LAT by focus
- LAT by Wordnet
- Selection verb (and LAT by SV)

Question focus identification is the point of the sentence where you should be able to place the answer:

- What was the first **book** written by Terry Pratchett?
- The **actor** starring in Moon?
- **Where** is Mt. Olympus?
- **When** was the U.S. capitol built?
- **Who** played Marge in The Simpsons?

# YodaQA Focus Identification

Blackboard rules based on specific dependencies and pos tags detected in the sentence (about 5 rules), e.g.:

- If there's ADVMOD (adverb modifier) dependency pointing at lemma "how", the other end is the focus (e.g. "how"-" old")
- As a fallback, the NSUBJ (noun subject) dependency endpoint is the focus (e.g. "who"-"did"-"X", "spouse"-" of"-"Madonna")

# YodaQA LAT

LAT By Focus - transform focus to LAT; mostly just copy it over, plus blackboard:

where → place, Who → person

LAT By Wordnet - expands existing LATs, adding more LATs, by hypernymy relations in Wordnet:

biologist → scientist → person → living being

# YodaQA SV identification

Find semantically meaningful verb in the sentence

- Where is Mt. Olympus (none)
- When was the U.S. capitol **built**?
- Who **invented** telephone?
- Who **played** Marge in The Simpsons?

Simple heuristic rule based on the parse tree root

# YodaQA Question Clues

Search keywords and keyphrases

- Noun phrases (constituents, parser output)
- Nouns
- Named entities
- LAT and SV

\+ Entity Linking

Clues to set of concepts (links to DBpedia)

- Candidate generation
- Entity lookup
- Disambiguation
- Deduplication

# YodaQA Candidate Generation

- Candidates from clues
- Candidates from 2,3,4-grams
- Lookup by fuzzy search in entity names+aliases
  (sorted by popularity - number of cross-links)
- Lookup in crosswikis: labels used as enwiki link texts (sorted by probability of linking the particular entity)

# YodaQA Candidates

Marge:

- crosswiki: Marge Simpson (p=0.858966)
- fuzzy lookup: Marge (d=0), Margay (d=2), Margaz (d=2)
  - List of supporting Harry Potter characters
  - Marge (cartoonist)
  - Marge Burns
- The Simpsons

# YodaQA Candidate Ranking

Classifier scoring candidate entities to be linked to the keyword/keyphrase (top 5 kept; logistic regression)
Features: p_crosswiki, log-popularity, edit distance, origin

- The Simpsons Movie: 0.784
- The Simpsons: 0.667
- Marge Simpson: 0.376
- Marge Champion: 0.112
- List of supporting Harry Potter characters: 0.079

# YodaQA Question Representation

- LAT, SV (“semantics”)
- Concepts (entity linking)
- Clues (search keywords; in database-only task, mostly superseded by concepts)

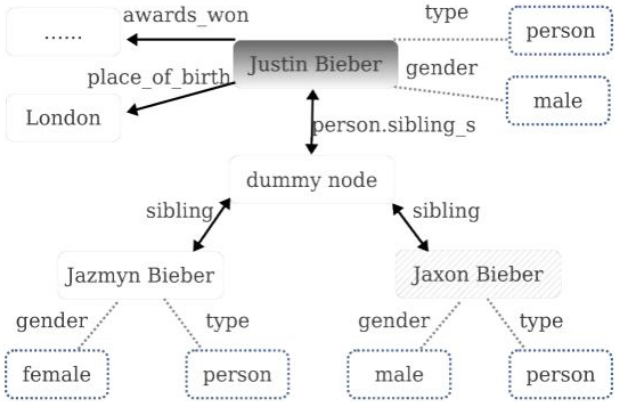
# YodaQA Database Queries

- Freebase (robust)
- DBpedia(noisy, gappy!)
- Wikidata (work in progress)

- Use RDF, SPARQL
- Use Freebase Property Paths

  LAT = director

  try property "/film/director/film"



# YodaQA Answer Scoring

- Match property label of the answer to question repr using word embeddings
- Match question LAT (e.g. "person") to the answer LAT (e.g. "director" or "date").
- Gradient-boosted Decision Forest (200 trees)
  Extra features:
  - Question Class (6-fold classification)
  - Entity Linking Score

# YodaQA Result

Who played Marge in The Simpsons?

1199 candidates, 308 unique

1. Julie Kavner (conf. 0.991)
2. Dan Castellaneta (conf. 0.092)
3. Hank Azaria (conf. 0.092)

Quality:

- KB: 62.9% on a collection of movie questions (Google: 59.9%)
- Fulltext: 35% on general Wikipedia-based questions

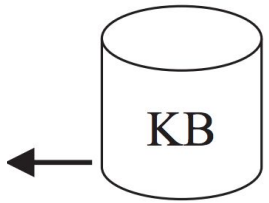
# YodaQA Limitations

- List queries (+ “first”, “how many”)
- Transformations: how old → birthdate (and back)
- Compound queries:
  When was the third wife of Tom Cruise born?
- Performance: Fast SPARQL is tricky

# Knowledge-based question answering

Question: *What is the capital of Germany?*
*Which city is the capital of Germany?*
*Which city in Germany is the capital?*

Logical form: λx.capital(x, Germany)




Result: *{Berlin}*

# Typed lambda-calculus expressions

**Idea:** text → a logical expression

*“People who lived in France”*

$\lambda x.\text{person}(x) \wedge \exists e.\text{PlacesLived}(x, e) \wedge \text{Location}(e, \text{France})$

# Lambda calculus notation

- Constants
  - *entity* Texas*, property* size(e, r)
- Logical connectors - ∧, ∨, ¬,
- Quantification - ∀, ∃
- Lambda expressions
  - λx.borders(x, Texas)
- Additional functions
  - count, argmax, argmin, etc.

# Lambda calculus examples

1. What states border Texas?

$$\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{Texas})$$

# Lambda calculus examples

1. What states border Texas?

   $\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{Texas})$

2. What is the largest state bordering Texas?

   $\text{argmax}(\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{Texas}), \lambda x.\text{size}(x))$

# Lambda calculus examples

1. What states border Texas?

   $\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{Texas})$

2. What is the largest state bordering Texas?

   $\text{argmax}(\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{Texas}), \lambda x.\text{size}(x))$

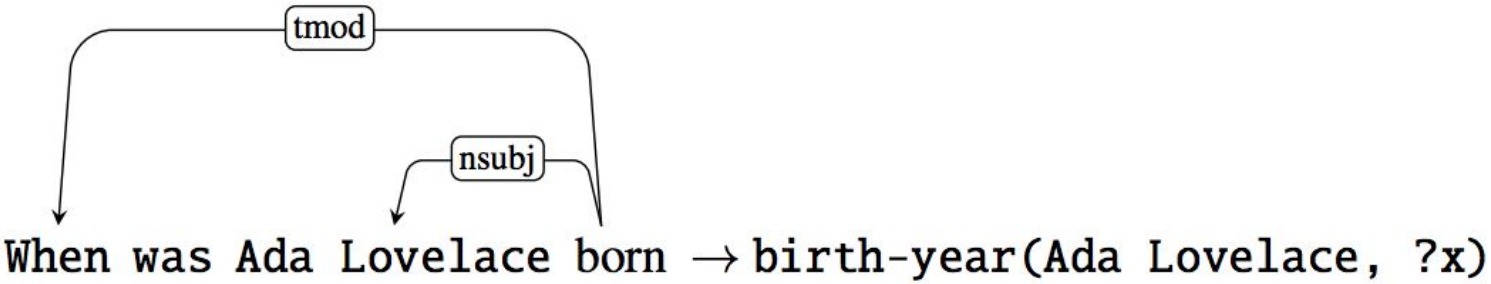
3. What states border the state that borders the most states?

   $\lambda x.\text{state}(x) \wedge \text{borders}(x, \text{argmax}(\lambda y.\text{state}(y), \lambda y.\text{count}(\lambda z.\text{state}(z) \wedge \text{borders}(y, z))))$

# QA using Semantic parsing

- parse a sentence
- split the parse tree into tuples by predicates
- classify each predicate to a KB relation
- map arguments

Data: QA datasets.



# Semantic parsing variation problem

Problems:

- each KB has its own set of relations
- the same relation can be expressed with different words

| | |
|---|---|
| $s_1$ | *What is the population of Berlin?* |
| $s_2$ | *How many people live in Berlin?* |
| $lf_1$ | $\lambda$x.population(Berlin,x) |
| $lf_2$ | count($\lambda$x.person(x)$\wedge$live(x,Berlin)) |

# Semantic parsing variation problem

WikiAnswers

| |
|---|
| *How many people live in chembakolli?*<br>*How many people is in chembakolli?*<br>*How many people live in chembakolli india?*<br>*How many people live there chembakolli?*<br>*How many people live there in chembakolli?*<br>*What is the population of Chembakolli india?* |
| *What currency is used on St Lucia?*<br>*What is st lucia money?*<br>*What is the money used in st lucia?*<br>*What kind of money did st lucia have?*<br>*What money do st Lucia use?*<br>*Which money is used in St Lucia?* |



Taken from Chen et al. (2016)

# Semantic parsing variation problem

Paraphrasing:

- with a dictionary
  - *daughter => female child*
- with a paraphrase database
  - *created from => made out of*
- with templates
  - *How many people live in => What is the population*

# QA datasets

http://nlpprogress.com/english/question_answering.html

- TREC QA Datasets (https://trec.nist.gov/data/qa.html)
- SQuAD (+ Ukrainian version)
- WikiQA
- and many many others like:
  - Quora Question Pairs (QQP)
  - RecipeQA
  - NarrativeQA

# QA evaluation

- Accuracy, F1, Prec, Rec (as usual)
- Confidence-weighted accuracy
- Mean Average Precision (MAP) (standard IR measure)

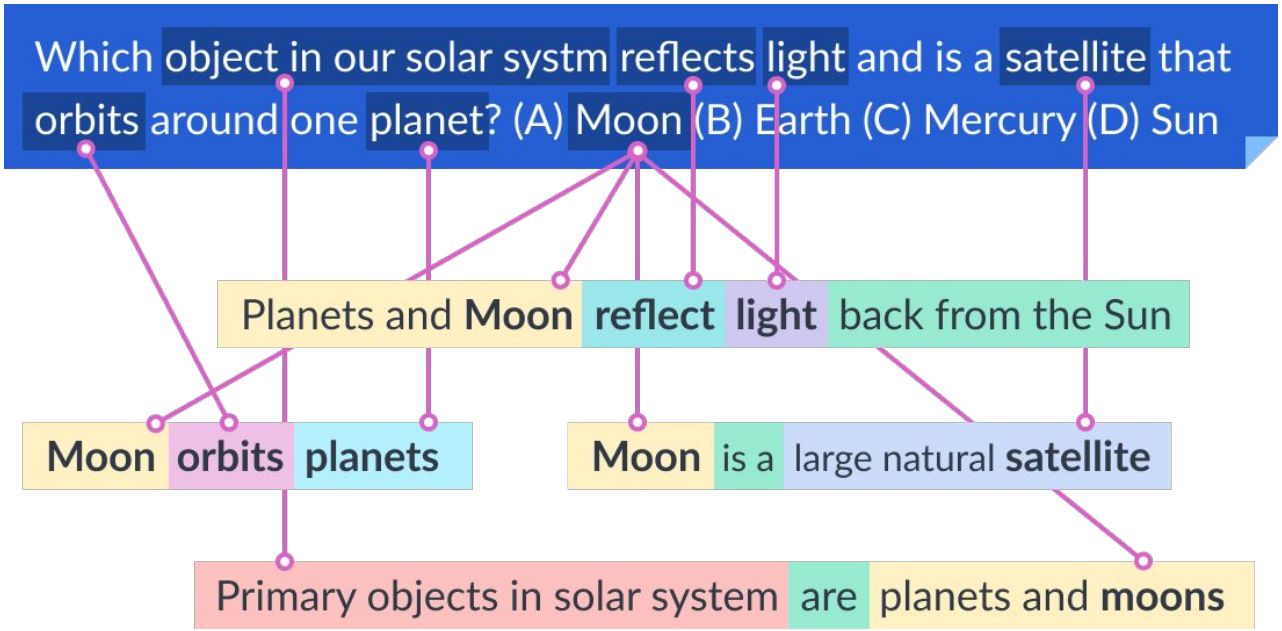
$$MAP = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{Q_j}\sum_{i=1}^{Q_j}P(rel=i)$$

- Mean Reciprocal Rank (MRR)

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}$$

# Towards general-purpose QA

AllenAI Aristo https://allenai.org/aristo



# References


- Nematzadeh, Meylan and Griffiths (2017), The Unreasonable Effectiveness of Counting Words Near Other Words
- Navigli and Lapata (2010), An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation
- Raiman and Raiman (2018), Discovering Types for Entity Disambiguation
- Moro, Raganato, and Navigli (2015), Babelfy: Entity Linking meets Word Sense Disambiguation
- Robert Speer (2018), ConceptNet 5.5: An Open Multilingual Graph of General Knowledge
- Athiwaratkun and Wilson (2017), Multimodal Word Distributions
- Wang, Berant and Liang (2015), Building a Semantic Parser Overnight
- Chen et al. (2016), Sentence Rewriting for Semantic Parsing
- Alicia Ageno (2014), Textual Entailment
- Mark Sammons (2012), Recognizing Textual Entailment
- Jurafsky and Martin (2017), Speech and Language Processing, Chapters 19, 20 and 25

# References


- AMR Banks: https://amr.isi.edu/download.html
- Banarescu et al. (2013), AMR for Sembanking
- Schneider et al. (2015), The Logic of AMR
- Damonte et al. (2017), An Incremental Parser for AMR
- Chen and Palmer (2017), Unsupervised AMR-Dependency Parse Alignment
- Wang et al. (2015), A Transition-based Algorithm for AMR Parsing
- Flanigan et al. (2014), A Discriminative Graph-Based Parser for AMR
- Parser for the PENMAN format: https://github.com/goodmami/penman
- JAMR parser and generator: https://github.com/jflanigan/jamr
- AMREager parser: http://cohort.inf.ed.ac.uk/amreager.html
- CAMR parser: https://github.com/c-amr/camr
- Tutorial on QA Systems: https://www.slideshare.net/shekarpour/tutorial-on-question-answering-systems
- Technology behind YodaQA: http://ailao.eu/yodaqa/technology.pdf