# Distributional Semantics

## Lecture 7. Multi-sense Word Embeddings

Daniil Vodolazsky

April 6th, 2019

## Lecture Plan

## Lecture Plan

1. Ambiguity, Polysemy and Vagueness
2. Word Sense Induction and Word Sense Disambiguation
3. Automatic Learning the Number of Senses
4. AdaGram
5. Defenition Generation
6. Learning Word Sense Embeddings
7. Interpretable Word Sense Disambiguation
8. Seminar

## Introduction

## Introduction: What do you think?

**What is sense?**

## Ambiguity, Polysemy and Vagueness

## Ambiguity

**Ambiguity** is a term used to characterize phenomena that have more than only one meaning. **Lexical ambiguity** is concerned with multiple interpretations of lexemes. A word is ambiguous if it involves two lexical items that have identical forms, but have distinct, i.e. unrelated meanings.

**bank** (financial organization)

river **bank**

## Vagueness

**Vagueness** (Tuggy, 1993) is a linguistic phenomenon, where ”two or more meanings associated with a given phonological form are ... united as non-distinguished subcases of a single, more general meaning”. That means that vagueness involves ”a lexeme with a single but nonspecific meaning”.

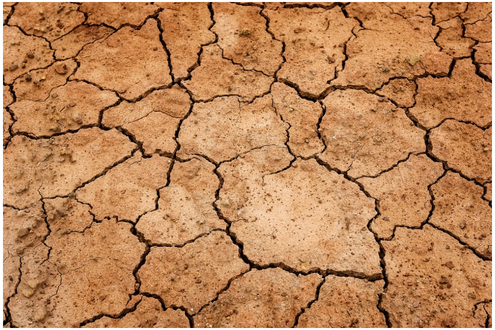
Shaquille O’Neal is a **tall** man.

This is a **tall** building.

## Polysemy

On a scale of meaning variance ambiguity and vagueness are the two extremes, whereas **polysemy** is in between the other two. It shares features with both and is a common phenomenon in everyday language use. Polysemy involves lexemes that are clearly united (share a common schema) as well as clearly seperable at the same time.

**Earth** Observation System



... and the **earth** cracked under it.

## Word Sense Induction and Word Sense Disambiguation

## Word Sense Induction and Word Sense Disambiguation

We address the problem of unsupervised learning of multiple representations that correspond to different meanings of a word, i.e. building multi-prototype word representations. This may be considered as specific case of **word sense induction (WSI)** problem which consists in automatic identification of the meanings of a word. In our case different meanings are distinguished by separate representations. Here we define **meaning** or **sense** as distinguishable interpretation of the spelled word which may be caused by any kind of ambiguity.

Word-sense induction is closely related to the **word sense disambiguation (WSD)** task where the goal is to choose which meaning of a word among provided in the *sense inventory* was used in the context. The sense inventory may be obtained by a WSI system or provided as external information.

## Automatic Learning the Number of Senses

## Automatic Learning the Number of Senses

Such an algorithm should have the property that a word should be associated with a new sense vector just when evidence in the context (e.g., neighboring words, document-level co-occurrence statistics) suggests that it is sufficiently different from its early senses.

Such a line of thinking naturally points to **Chinese Restaurant Processes (CRP)** which have been applied in the related field of word sense induction. In the analogy of CRP, the current word could either sit at one of the existing tables (belonging to one of the existing senses) or choose a new table (a new sense).

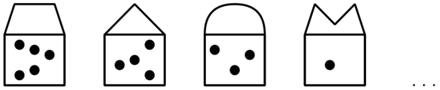
The decision is made by measuring semantic relatedness (based on local context information and global document information) and the number of customers already sitting at that table (the popularity of word senses).

## Chinese Restaurant Processes

In the analogy, each data point is compared to a customer in a restaurant. The restaurant has a series of tables $t$, each of which serves a dish $d_t$. This dish can be viewed as the index of a cluster or a topic. The next customer $w$ to enter would either choose an existing table, sharing the dish (cluster) already served or choosing a new cluster based on the following probability distribution:

$$P(t_w = t) \propto \begin{cases} N_t P(w \mid d_t) & \text{if } t \text{ already exists}, \\ \gamma P(w \mid d_{new}) & \text{if } t \text{ is new}, \end{cases}$$

where $N_t$ is the number of customers already sitting at table $t$ and $P(w \mid d_t)$ is the probability of assigning the current data point to cluster $d_t$. $\gamma$ is the hyper parameter controlling the preference for sitting at a new table.



## Incorporating CRP into Distributed Language Models (2015)

Each token $w$ is associated with a $d$-dimensional global embedding $\boldsymbol{u}_w$. Additionally, it is associated with a set of senses $\mathcal{Z}_w = \{1, 2, \ldots, K(w)\}$. Each sense $z$ is associated with a distinct sense-specific embedding $\boldsymbol{u}_{w,z}$.
We would use CRP to decide which sense the current occurrence $w_t$ corresponds to, or construct a new sense if it is a new meaning. Based on CRP, the probability that assigns the current occurrence to each of the discovered senses or a new sense is given by:

$$P(z_t = z \mid \mathcal{C}_t) \propto \begin{cases} N_z^w P(w_t, z_t = z \mid \mathcal{C}_t) & \text{if } z \text{ already exists,} \\ \gamma P(w_t, z_t = z \mid \mathcal{C}_t) & \text{if } z \text{ is new,} \end{cases}$$

where $N_z^w$ denotes the number of times already assigned to sense $z$ for token $w$.
The vector representation $\boldsymbol{u}_{w_t,z}$ for the newly detected sense would be obtained by maximizing the function $P(w_t, \mid z_t = z, \mathcal{C}_t)$. Otherwise, we use a standard update procedure.

## Obtaining Word Representations for NLU tasks

Next we describe how we decide sense labels for tokens in context.
Given a document or a sentence, we have an objective function with respect to sense labels. Computing the global optimum sense labeling—in which every word gets an optimal sense label—requires searching over the space of all senses for all words, which can be expensive. We therefore chose two simplified heuristic approaches:

- **Greedy Search**: Assign each token the locally optimum sense label and represent the current token with the embedding associated with that sense.
- **Expectation**: Compute the probability of each possible sense for the current word, and represent the word with the expectation vector:

$$\boldsymbol{u}_{w_t} = \sum_{z \in \mathcal{Z}_{w_t}} P(w_t \mid z_t = z, \mathcal{C}_t) \cdot \boldsymbol{u}_{w_t, z}.$$

## AdaGram

## AdaGram

**Adaptive Skip-gram (AdaGram)** (2016) model extends the original Skip-gram and may automatically learn the required number of prototypes for each word using Bayesian nonparametric approach.
For simplicity, assume that each word $w$ has $K(w)$ (a fixed number) meanings. That means that we have to modify an equation for hierarchical softmax to account for particular choice of the meaning. For this reason we introduce latent variable $z$ that encodes the index of active meaning and extend hierarchical softmax formula to

$$P(c \mid z_t = z, w_t) = \prod_{j=1}^{L(c)-1} \sigma\left( [\![ n(c, j+1) = \mathrm{ch}(n(c,j)) ]\!] \boldsymbol{u}_{w_t,z}^{\top} \boldsymbol{v}_{n(c,j)} \right).$$

In case of absence of any knowledge we can initialize prior probabilities with uniform distribution:

$$P(z_t = z \mid w_t) = \frac{1}{K(w_t)}.$$

## AdaGram: Naive EM-algorithm

Let $\mathcal{C}_t$ be a set of context words surrounding the word $w_t$ in the training corpora, and let $\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_T)$. For a single observation at position $t$ we can write

$$P(\mathcal{C}_t, z_t \mid w_t, \boldsymbol{\Theta}) = P(\mathcal{C}_t \mid z_t, w_t, \boldsymbol{\Theta}) P(z_t \mid w_t).$$

Since we do not know $z_t$, we can now use EM-algorithm that will both estimate our parameters $\boldsymbol{\Theta} = \{\boldsymbol{u}_{w,k}, \boldsymbol{v}_n\}$ and the probabilities of meanings of $w_t$ given its context $P(z_t \mid w_t, \mathcal{C}_t, \boldsymbol{\Theta})$.
A naive version of EM-algorithm can be written as follows.

- **E-step**. For each training object estimate the distribution on latent variable

$$P(z_t = z \mid w_t, \mathcal{C}_t, \boldsymbol{\Theta}) = \frac{P(\mathcal{C}_t \mid z, w_t, \boldsymbol{\Theta}) P(z_t = z \mid w_t)}{\sum_{z'=1}^{K(w_t)} P(\mathcal{C}_t \mid z', w_t, \boldsymbol{\Theta}) P(z_t = z' \mid w_t)}.$$

  We can do this in explicit manner assuming the number of meanings is reasonably small.
- **M-step**. Optimize

$$\mathbb{E}_{\mathcal{Z}} \log P(\mathcal{C} \mid \mathcal{Z}, \mathcal{D}, \mathcal{W}, \boldsymbol{\Theta}) P(\mathcal{Z} \mid \mathcal{D}, \mathcal{W}) \longrightarrow \max_{\boldsymbol{\Theta}}$$

  which is equivalent to train standard skip-gram with increased number of context words.

## AdaGram: Optimized EM-algorithm

Consider the gradient of $\mathbb{E}_{\mathcal{Z}} \log P(\mathcal{C} \mid \mathcal{Z}, \mathcal{D}, \mathcal{W}, \mathbf{\Theta}) P(\mathcal{Z} \mid \mathcal{D}, \mathcal{W})$ in detail:

$$\nabla_{\mathbf{\Theta}} \mathbb{E}_{\mathcal{Z}} \log P(\mathcal{C} \mid \mathcal{Z}, \mathcal{D}, \mathcal{W}, \mathbf{\Theta}) P(\mathcal{Z} \mid \mathcal{D}, \mathcal{W}) =$$
$$\nabla_{\mathbf{\Theta}} \mathbb{E}_{\mathcal{Z}} \sum_{t=1}^{T} (\log P(\mathcal{C}_t \mid z_t, w_t, \mathbf{\Theta}) + \log P(z_t \mid w_t)) =$$
$$\sum_{t=1}^{T} \mathbb{E}_{z_t} (\nabla_{\mathbf{\Theta}} \log P(\mathcal{C}_t \mid z_t, w_t, \mathbf{\Theta}) + \nabla_{\mathbf{\Theta}} \log P(z_t \mid w_t)) =$$
$$\sum_{t=1}^{T} \mathbb{E}_{z_t} \nabla_{\mathbf{\Theta}} \log P(\mathcal{C}_t \mid z_t, w_t, \mathbf{\Theta}).$$

Its unbiased estimate is

$$\mathbb{E}_{z_t} \nabla_{\mathbf{\Theta}} \log P(\mathcal{C}_t \mid z_t, w_t, \mathbf{\Theta}) = \sum_{z=1}^{K(w_t)} \underbrace{P(z_t = z \mid w_t, \mathcal{C}_t, \mathbf{\Theta})}_{\text{computed on E-step}} \nabla_{\mathbf{\Theta}} \log P(\mathcal{C}_t \mid z, w_t, \mathbf{\Theta}).$$

## AdaGram: Optimized EM-algorithm

Finally, the modified algorithm looks like this. Do one pass through training data. For each $t \in \{1, \dots, T\}$

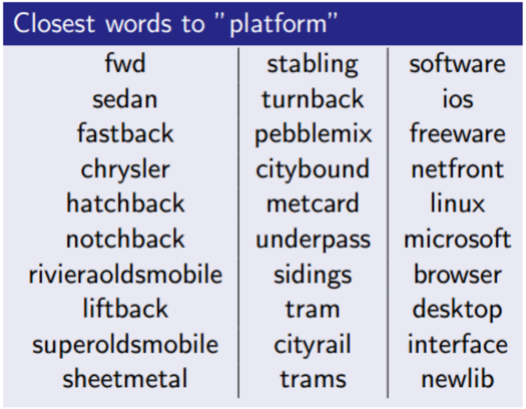
- **E-step**. Compute the probabilities of meanings for $w_t$:

$$P(z_t = z \mid w_t, \mathcal{C}_t, \boldsymbol{\Theta}) = \frac{P(\mathcal{C}_t \mid z, w_t, \boldsymbol{\Theta}) P(z_t = z \mid w_t)}{\sum_{z'=1}^{K(w_t)} P(\mathcal{C}_t \mid z', w_t, \boldsymbol{\Theta}) P(z_t = z' \mid w_t)}.$$

- **M-step**. Make one step towards stochastic gradient:

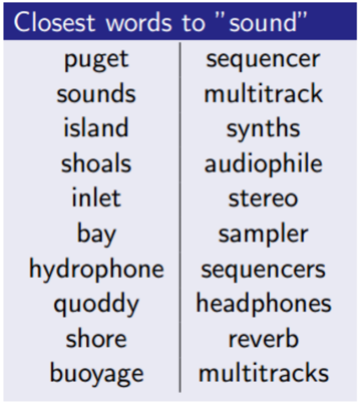
$$\boldsymbol{\Theta}_{new} = \boldsymbol{\Theta}_{old} + \epsilon \sum_{z=1}^{K(w_t)} P(z_t = z \mid w_t, \mathcal{C}_t, \boldsymbol{\Theta}) \nabla_{\boldsymbol{\Theta}} \log P(\mathcal{C}_t \mid z, w_t, \boldsymbol{\Theta}).$$

## AdaGram: Results



Closest words to "platform"

| | | |
|---|---|---|
| fwd | stabling | software |
| sedan | turnback | ios |
| fastback | pebblemix | freeware |
| chrysler | citybound | netfront |
| hatchback | metcard | linux |
| notchback | underpass | microsoft |
| rivieraoldsmobile | sidings | browser |
| liftback | tram | desktop |
| superoldsmobile | cityrail | interface |
| sheetmetal | trams | newlib |



Closest words to "sound"

| | |
|---|---|
| puget | sequencer |
| sounds | multitrack |
| island | synths |
| shoals | audiophile |
| inlet | stereo |
| bay | sampler |
| hydrophone | sequencers |
| quoddy | headphones |
| shore | reverb |
| buoyage | multitracks |

## AdaGram: Results

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation

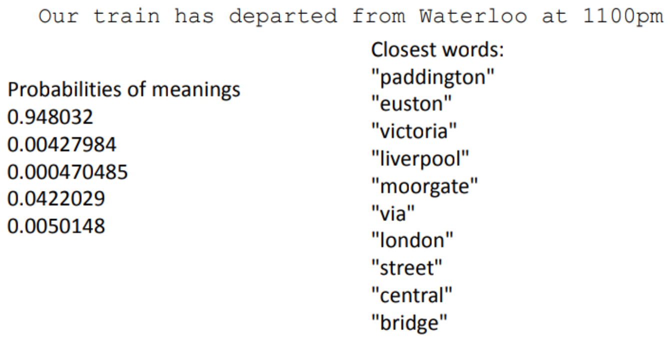
```
Who won the Battle of Waterloo?
```

Probabilities of meanings
0.0000098
0.997716
0.0000309
0.00207717
0.00016605

Closest words:
"sheriffmuir"
"agincourt"
"austerlitz"
"jena-auerstedt"
"malplaquet"
"königgrätz"
"mollwitz"
"albuera"
"toba-fushimi"
"hastenbeck"

# AdaGram: Results

- We run AdaGram with $\alpha = 0.2$
- 5 meanings for 'Waterloo' were found
- Let us try to make disambiguation



```
Our train has departed from Waterloo at 1100pm
```

Probabilities of meanings
0.948032
0.00427984
0.000470485
0.0422029
0.0050148

Closest words:
"paddington"
"euston"
"victoria"
"liverpool"
"moorgate"
"via"
"london"
"street"
"central"
"bridge"

## Definition Generation

## Definition Generation (2018)

| Word | Context | Definition |
|---|---|---|
| star | she got star treatment | a person who is very important |
| sentence | sentence in prison | an act of restraining someone or something |
| sentence | write up the sentence | a piece of text written to be printed |
| head | the head of a man | the upper part of a human body |
| head | he will be the head of the office | the chief part of an organization, institution, etc |
| rape | the woman was raped on her way home at night | the act of killing |
| invisible | he pushed the string through an inconspicuous hole | not able to be seen |
| shake | my faith has been shaken | cause to be unable to think clearly |
| nickname | the nickname for the u.s. constitution is ‘old ironsides ’ | a name for a person or thing that is not genuine |

## Learning Word Sense Embeddings

## Learning Word Sense Embeddings (2016)

A method consists of the four main stages:

1. learning word embeddings;
2. building a graph of nearest neighbours based on vector similarities;
3. induction of word senses using ego-network clustering;
4. aggregation of word vectors with respect to the induced senses.

This method can use existing word embeddings, sense inventories and word similarity graphs.

## Calculating Word Similarity Graph

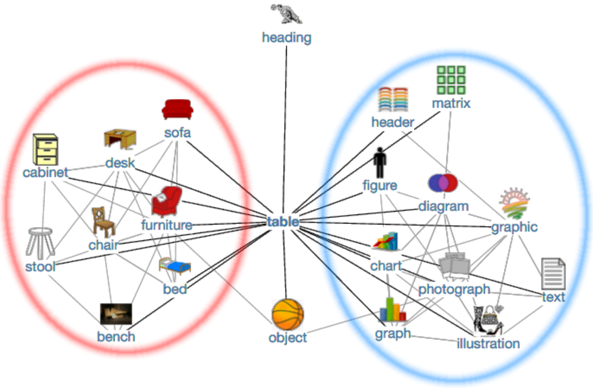
At this step, we build a graph of word similarities, such as (table, desk, 0.78). For each word we retrieve its 200 nearest neighbours. The similarity graph can be computed at least in two ways.

- **Similarities using Word2Vec** Nearest neighbours of a term are terms with the highest cosine similarity of their respective vectors.
- **Similarities using JoBimText** In this unsupervised approach, every word is represented as a bag of sparse dependency-based features extracted using the Malt parser and collapsed. Features are normalized and further pruned down according to the recommended defaults. Similarity of two words is equal to the number of common features.

# Word Sense Induction

To induce senses, first we construct an ego-network of a word and then perform graph clustering of this network. The identified clusters are interpreted as senses.

| Vector | Nearest Neighbours |
|---|---|
| table | tray, bottom, diagram, bucket, brackets, stack, basket, list, parenthesis, cup, trays, pile, playfield, bracket, pot, drop-down, cue, plate |
| table0 | leftmost0, column1, randomly0, tableau1, top-left0, indent1, bracket3, pointer0, footer1, cursor1, diagram0, grid0 |
| table1 | pile1, stool1, tray0, basket0, bowl1, bucket0, box0, cage0, saucer3, mirror1, birdcage0, hole0, pan1, lid0 |



## Word Sense Induction

**Input** $\mathcal{T}$ — word similarity graph, $N$ — ego-network size, $n$ — ego-network connectivity, $k$ — minimum cluster size
**Output** for each term $t \in \mathcal{T}$, a clustering $\mathcal{S}_t$ of its $N$ most similar terms

```
foreach $t \in \mathcal{T}$ do
    $V \leftarrow N$ most similar terms of $t$ from $\mathcal{T}$
    $\mathcal{G} \leftarrow$ graph with $V$ as nodes and no edges $E$
    foreach $v \in V$ do
        $V' \leftarrow n$ most similar terms of $v$ from $\mathcal{T}$
        foreach $v' \in V$ do
            if $v' \in V$ then
                add edge $(v, v')$ to $E$
    $\mathcal{S}_t \leftarrow$ ChineseWhispers($\mathcal{G}$)
    $\mathcal{S}_t \leftarrow \{s \in \mathcal{S}_t : |s| \geq k\}$
```

## Pooling of Word Vectors

We define a sense vector as a function of word vectors representing cluster items. Let $\mathcal{S}_i \subseteq \mathcal{W}$ be a sense cluster obtained during the previous step. Consider a function $\gamma_i : \mathcal{W} \to \mathbb{R}_+$ that maps cluster words to their weight in the cluster $\mathcal{S}_i$.
Two ways to calculate sense vectors: unweighted average of word vectors:

$$\boldsymbol{s}_i = \frac{\sum_{w \in \mathcal{S}_i} \boldsymbol{u}_w}{|\mathcal{S}_i|};$$

and weighted average of word vectors:

$$\boldsymbol{s}_i = \frac{\sum_{w \in \mathcal{S}_i} \gamma_i(w) \boldsymbol{u}_w}{\sum_{w \in \mathcal{S}_i} \gamma_i(w)}.$$

## Word Sense Disambiguation

Given a target word $w_t$ and its context words $\mathcal{C}_t$, we first map $w_t$ to a set of its sense vectors according to the inventory: $\{\boldsymbol{s}_1, \dots, \boldsymbol{s}_n\}$.

The first strategy for searching sense vector is based on sense probability in given context:

$$\boldsymbol{s}^* = \arg\max_i P(\mathcal{C}_t \mid \boldsymbol{s}_i) = \arg\max_i \frac{1}{1 + \exp\left(-\bar{\boldsymbol{v}}_c \cdot \boldsymbol{s}_i\right)},$$

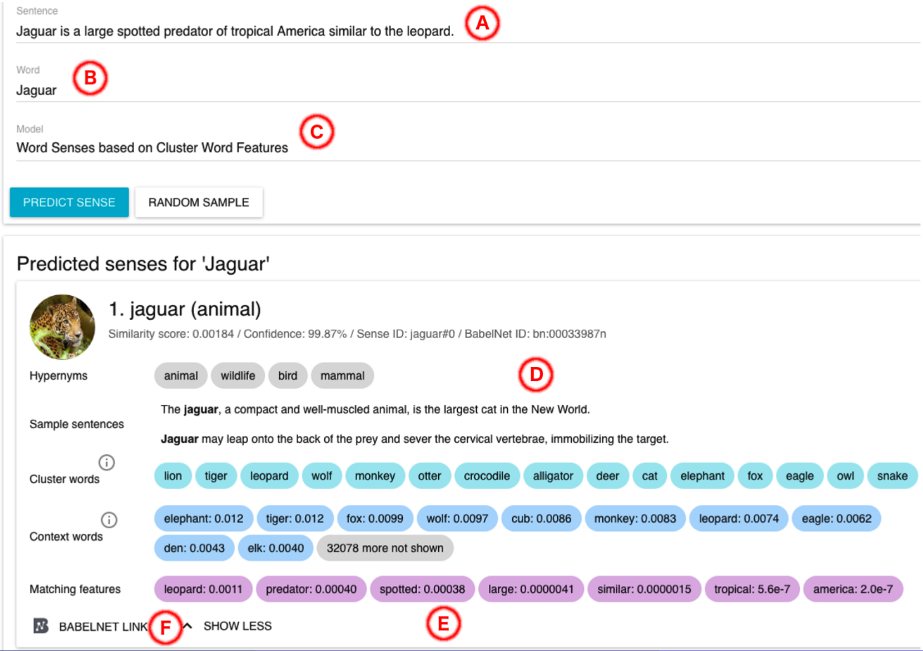
where $\bar{\boldsymbol{v}}_c$ is the mean of context embeddings.

The second disambiguation strategy is based on similarity between sense and context:

$$\boldsymbol{s}^* = \arg\max_i \mathrm{sim}(\boldsymbol{s}_i, \mathcal{C}_t) = \arg\max_i \cos(\boldsymbol{s}_i, \bar{\boldsymbol{u}}_c)$$

where $\bar{\boldsymbol{u}}_c$ is the mean of word embeddings for context words.

## Interpretable Word Sense Disambiguation



Sentence
Jaguar is a large spotted predator of tropical America similar to the leopard.

Word
Jaguar

Model
Word Senses based on Cluster Word Features

PREDICT SENSE    RANDOM SAMPLE

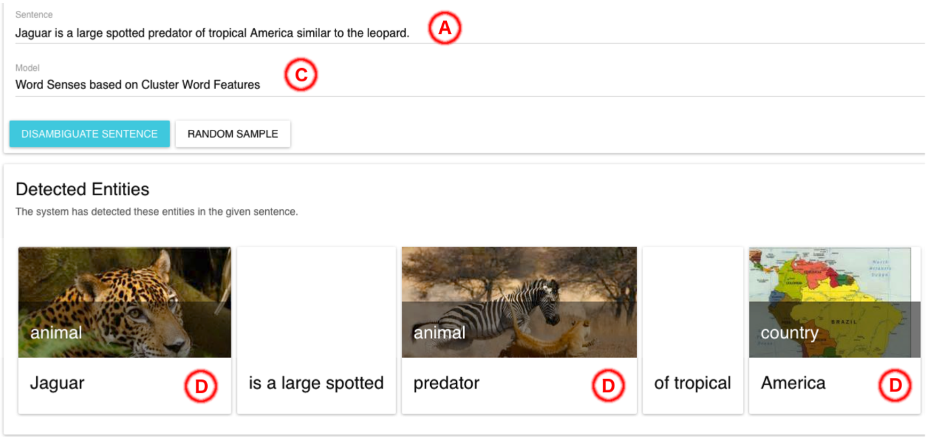
Predicted senses for 'Jaguar'

1. jaguar (animal)

Similarity score: 0.00184 / Confidence: 99.87% / Sense ID: jaguar#0 / BabelNet ID: bn:00033987n

| | |
|---|---|
| Hypernyms | animal, wildlife, bird, mammal |
| Sample sentences | The **jaguar**, a compact and well-muscled animal, is the largest cat in the New World. **Jaguar** may leap onto the back of the prey and sever the cervical vertebrae, immobilizing the target. |
| Cluster words | lion, tiger, leopard, wolf, monkey, otter, crocodile, alligator, deer, cat, elephant, fox, eagle, owl, snake |
| Context words | elephant: 0.012, tiger: 0.012, fox: 0.0099, wolf: 0.0097, cub: 0.0086, monkey: 0.0083, leopard: 0.0074, eagle: 0.0062, den: 0.0043, elk: 0.0040, 32078 more not shown |
| Matching features | leopard: 0.0011, predator: 0.00040, spotted: 0.00038, large: 0.0000041, similar: 0.0000015, tropical: 5.6e-7, america: 2.0e-7 |

BABELNET LINK    SHOW LESS



Sentence

Jaguar is a large spotted predator of tropical America similar to the leopard.

Model

Word Senses based on Cluster Word Features

Detected Entities

The system has detected these entities in the given sentence.

## Questions?

## Seminar

## RDT

1. Russian Distributional Thesaurus tutorial

https://nlpub.ru/Russian_Distributional_Thesaurus

Link