# Making Sense of Word Embeddings

By:Alexander Panchenko Et al.

Group 8 :

- Naman Agrawal
- Helen Zheng
- Sneha Thomas
- Daniel Sampreeth Reddy Eadara

## Why Language is difficult ..

polysemous

synonymous

Concept Layer

Lexical Layer

He sat on the river bank and counted his dough.

She went to the bank and took out some money.

# Word embeddings

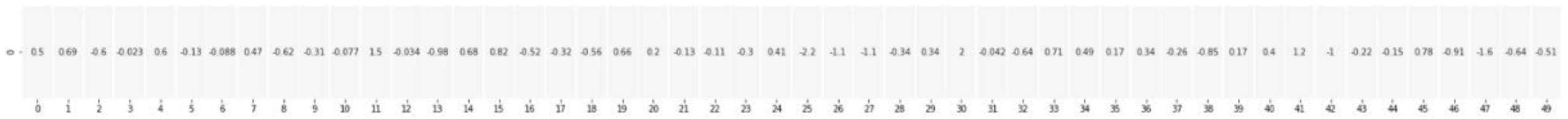
Representing text as numbers

- Machine learning models take vectors (arrays of numbers) as input. When working with text, the first thing we must do come up with a strategy to convert strings to numbers (or to "vectorize" the text) before feeding it to the model
- Vector are very useful as you can define vector spaces and calculate distance, closeness and similarity and plot them on graphs to perform actions such as clustering and other operations.

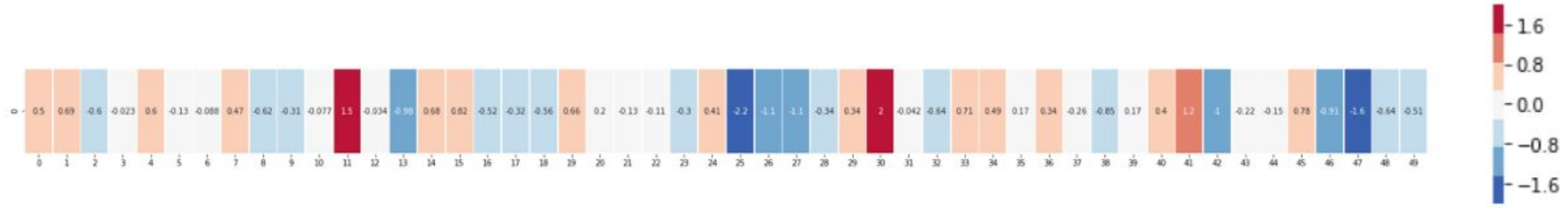
This is a word embedding for the word “king” (GloVe vector trained on Wikipedia):

```
[ 0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 ,
-0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961
, -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 ,
-0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 ,
-1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042 ]
```

It’s a list of 50 numbers. We can’t tell much by looking at the values. But let’s visualize it a bit so we can compare it other word vectors. Let’s put all these numbers in one row:

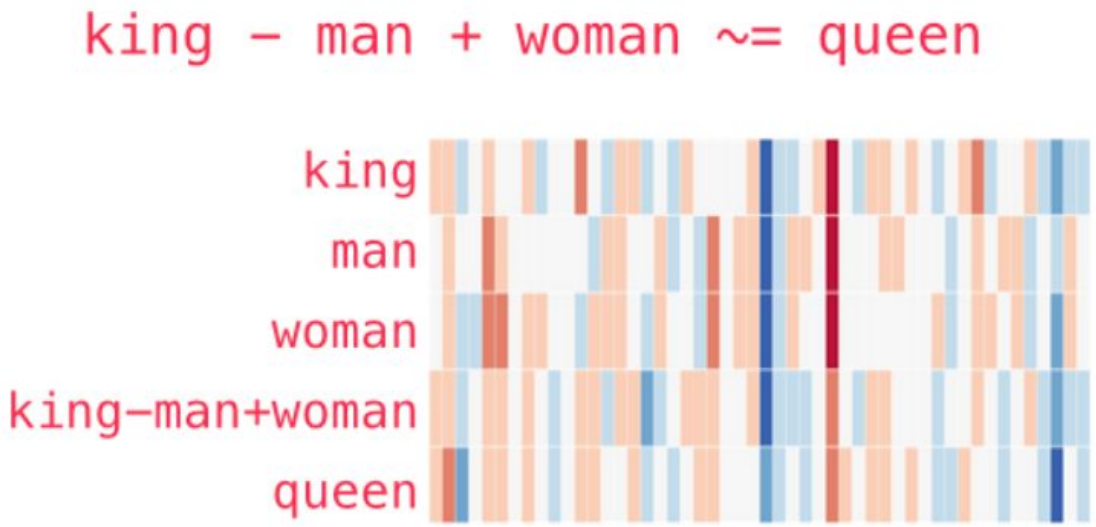


Let’s color code the cells based on their values (red if they’re close to 2, white if they’re close to 0, blue if they’re close to -2):



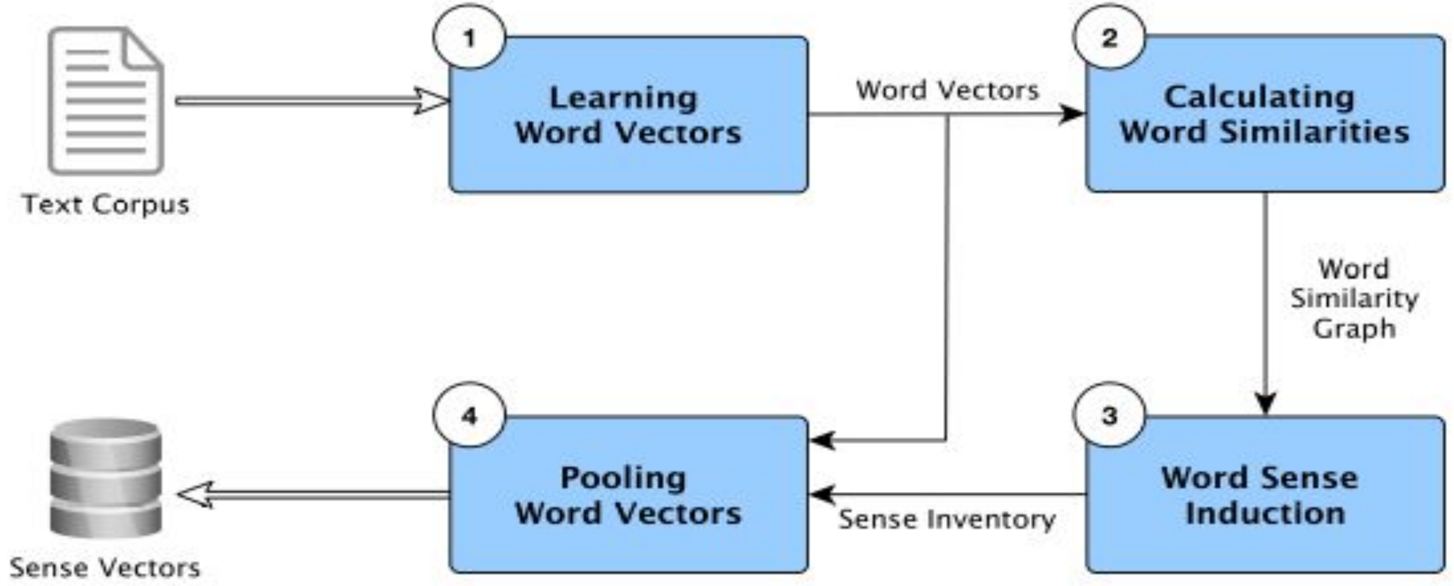
We’ll proceed by ignoring the numbers and only looking at the colors to indicate the values of the cells. Let’s now

We can visualize this analogy as we did previously:

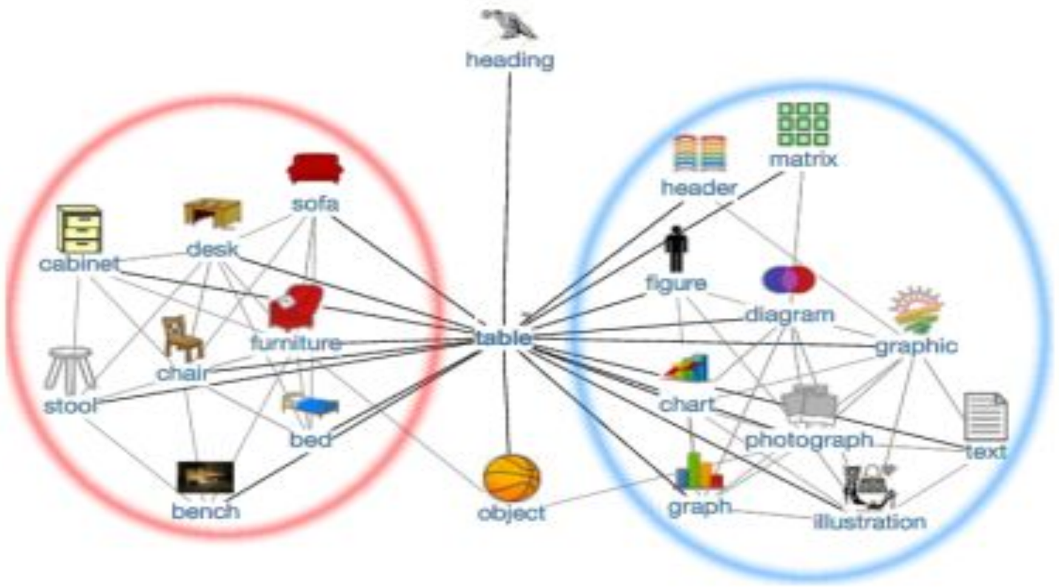


The resulting vector from "king-man+woman" doesn't exactly equal "queen", but "queen" is the closest word to it from the 400,000 word embeddings we have in this collection.

## Learning Word Sense Embeddings



## Word Sense Induction: Ego-Network Clustering



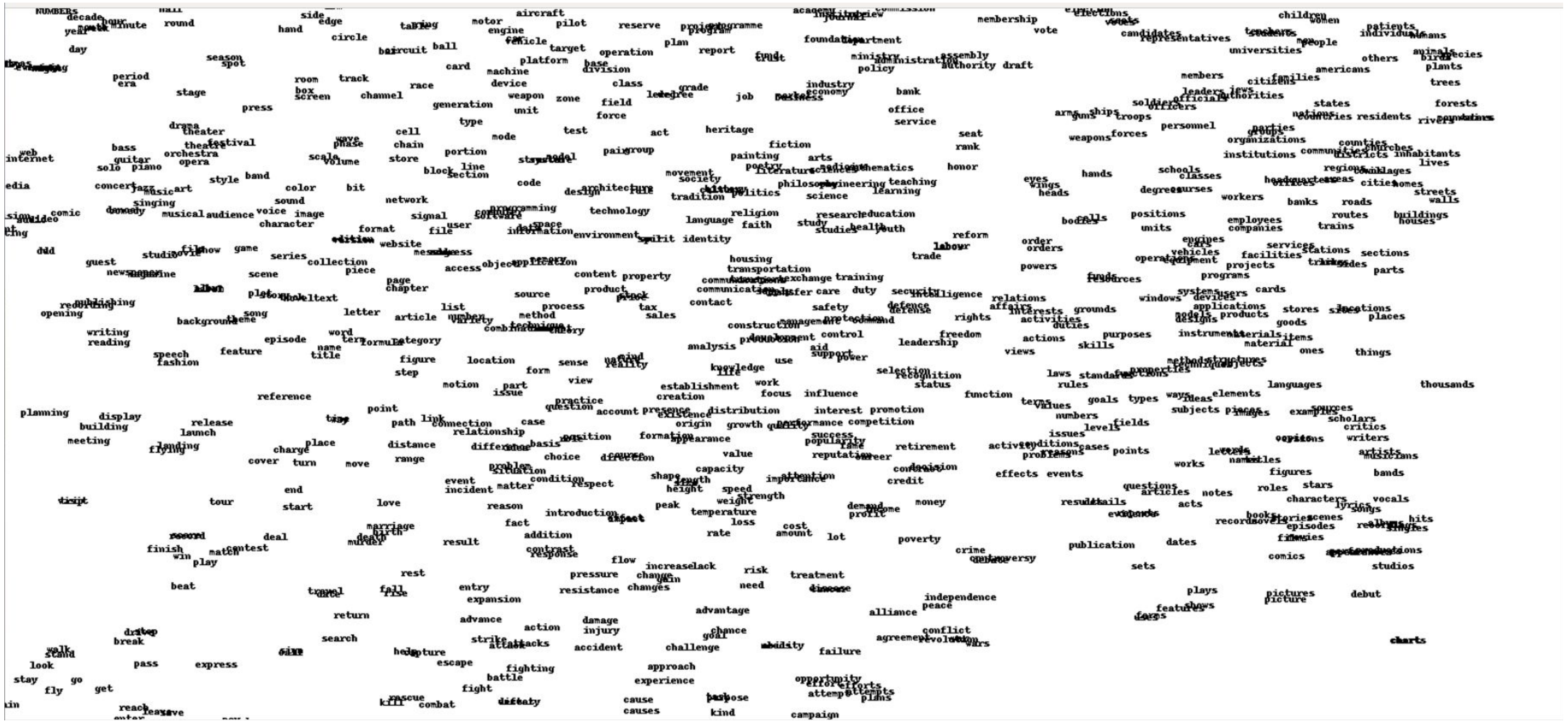
- Graph clustering using the **Chinese Whispers algorithm** (Biemann, 2006).

## Neighbours of Word and Sense Vectors

| Vector | Nearest Neighbours |
|---|---|
| table | tray, bottom, diagram, bucket, brackets, stack, bas-ket, list, parenthesis, cup, trays, pile, playfield, bracket, pot, drop-down, cue, plate |
| table#0 | leftmost#0, column#1, randomly#0, tableau#1, top-left0, indent#1, bracket#3, pointer#0, footer#1, cur-sor#1, diagram#0, grid#0 |
| table#1 | pile#1, stool#1, tray#0, basket#0, bowl#1, bucket#0, box#0, cage#0, saucer#3, mirror#1, birdcage#0, hole#0, pan#1, lid#0 |

- Neighbours of the word “table" and its senses produced by this method.
- The neighbours of the initial vector belong to **both senses**.
- The neighbours of the sense vectors are **sense-specific**.

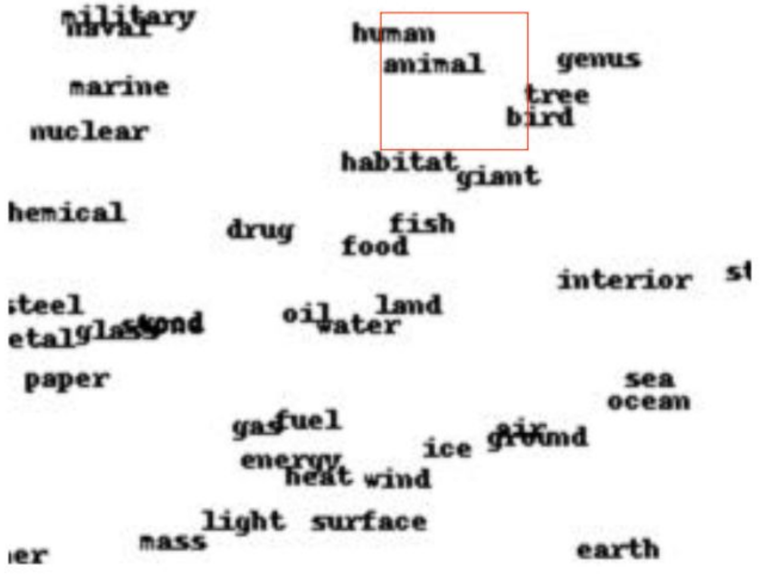
2D-projection of vector space : http://www.cs.toronto.edu/~hinton/turian.png



When we zoom in :

## Zoom in ...

http://www.cs.toronto.edu/~hinton/turian.png



22

## Previous Work

1. **Multi-Prototype Word Vector Spaces**
    a. Skip-gram model (Mikolov et al., 2013):
        - Uses parametric clustering algorithms that produce a fixed number of senses per word
    b. Neelakantan et al. (2014):
        - Multisense extension of the Skip-gram model.
        - During training, a new sense vector is allocated if the current context's similarity to existing senses is below some threshold.
    c. Adagram (Bartunov et al. (2016)):
        - Non-parametric Bayesian extension of Skip-gram capable to automatically learn the required number of representations for all words

## Previous Work

2. **Word Sense Disambiguation (WSD)**

   a. Supervised approaches:
      - Use an explicitly sense-labeled training corpus to construct a model, usually one per target word
      - Require considerable amounts of sense labeled examples

   b. Unsupervised approaches:
      - Automatically induce a sense inventory from raw corpora.
      - Categories:
        1. Context clustering
        2. Word clustering

## Limitations of Previous Work

- Does not take into account word ambiguity and conflate all senses of a word into a single Vector
- Require a known number of word meanings or learn them using greedy heuristic approaches
- Either directly learn sense representations from corpora or rely on sense inventories from lexical resources

## Author’s Proposal

- Proposes a simple and effective approach for learning word sense vectors.
- Relies on existing single-prototype word embeddings, transforming them to sense vectors via ego-network clustering.
- Fitted with a word sense disambiguation(WSD) mechanism wherein words in context can be mapped to these sense representations.
- Can use existing word embeddings and/or existing word sense inventories to build sense embeddings.
- Performance comparably to state-of-the-art unsupervised WSD systems

## Authors’ Approach - Overview

The authors’ approach consists of 4 steps:

1.) “Learning word embeddings.”
2.) “Building a graph of nearest neighbors based on vector similarities.”
3.) “Induction of word senses using ego-network clustering.”
4.) “Aggregation of word vectors with respect to the induced senses.”

-Pelevina et al.

## Authors’ Approach: Step 1

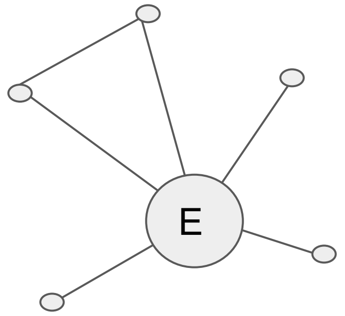
- “A word embedding is a learned representation for text where words that have the same meaning have a similar representation.”-- Jason Brownlee
- To create a word embedding, the authors approached using the CBOW model in the word2vec.
- The CBOW (Continuous Bag of words) model is a shallow neural network that learns to map a word (as a word vector) using the context windows.
    - A context window is the set of words in a sentence surrounding the word.
    - For example,
        - In “We are learning word embeddings.”
        - “are” and “word” are part of the context window of length 1 surrounding the word “learning.”
- From CBOW, the authors have word vectors.

## Authors’ Approach: Step 2

- Word similarities are computed.
- Because each word is represented by a word vector, word similarities can be computed by using cosine similarity.
  - Example: (table, desk, 0.78)—Pelevina et al.

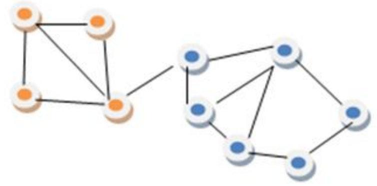
## Authors’ Approach: Step 3

- To create a word cluster for a word sense:
  - Step 1: Ego-networks of size N is created for each word.
  - Step 2: n similar word vectors are found for each N.
  - Step 3: An additional edge is then added between the N node and n node.
- Now the authors have an ego-network of word vectors.



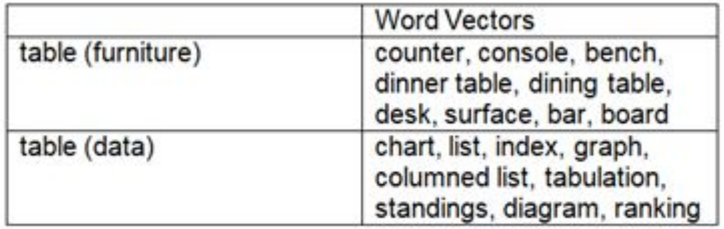
## Authors' Approach: Step 3

- Chinese Whispers clustering algorithm is used to group the nodes into classes.
  - Step 1: All nodes are a distinct class.
  - Step 2: A node is placed into a class that it has the most links with. (1 is randomly picked if there is more than 1 class as such.)
  - Repeat for all nodes.
  - Repeat until convergence.
- Now there are clusters of word vectors. These clusters are then sense vectors.



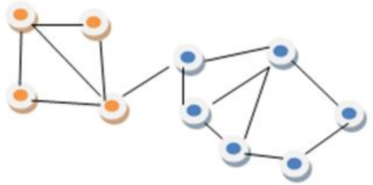
## Authors' Approach: Step 4

- Each word vector in a sense cluster is then weighted.
- And differentiation between table (as a furniture) and table (as data)
    - As clusters of word vectors or sense vectors.



| | Word Vectors |
|---|---|
| table (furniture) | counter, console, bench, dinner table, dining table, desk, surface, bar, board |
| table (data) | chart, list, index, graph, columned list, tabulation, standings, diagram, ranking |

Note: Info from table taken from paper by Pelevina et al.



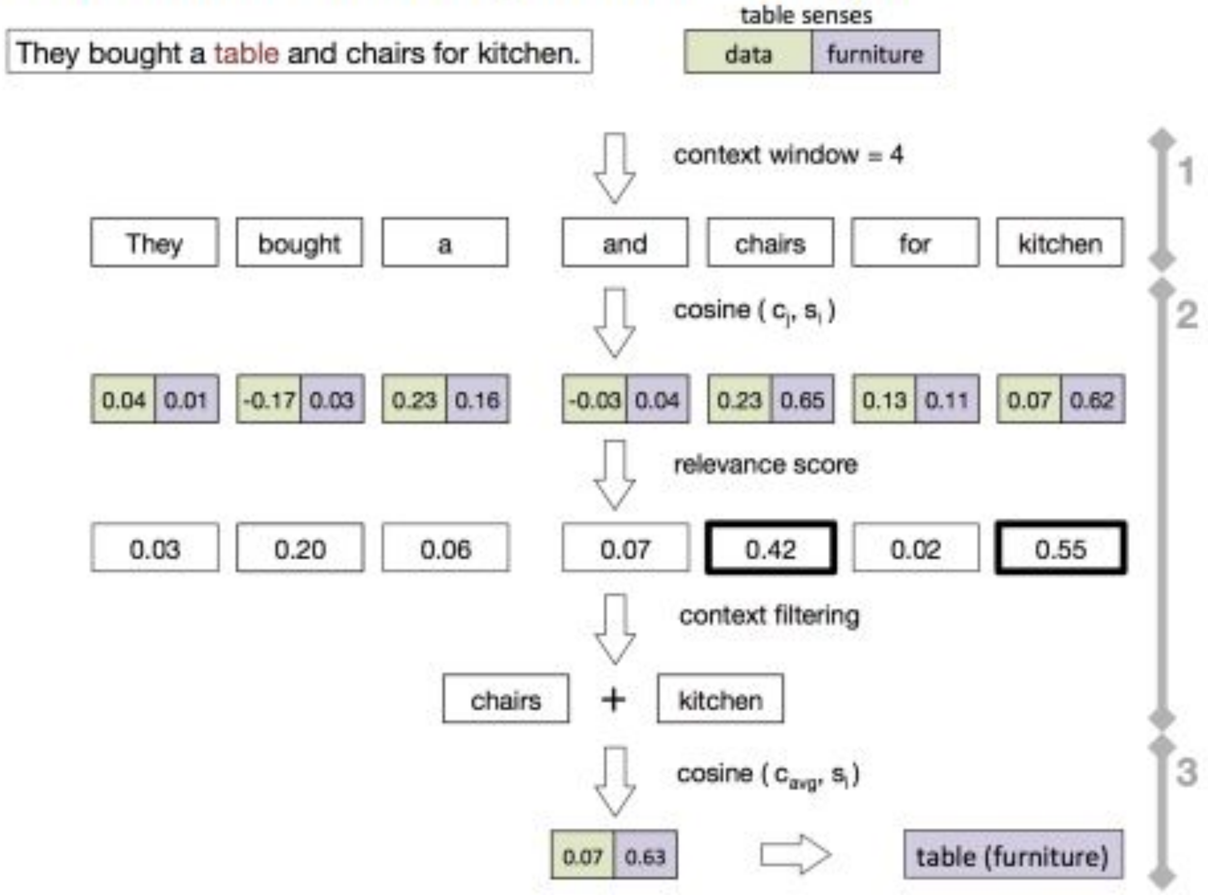
## Word Sense Disambiguation

1. Context Extraction
   - use context words around the target word

2. Context Filtering
   - based on context word's relevance for disambiguation

3. Sense Choice
   - maximize similarity between context vector and sense vector

## Word Sense Disambiguation: Example



Finally, with an inventory of senses in hand for each word, we must disambiguate. We have two choices for how to do this. One is to maximize the probability of the context given the sense—basically CBOW word2vec. $\bar{\mathbf{c}}_c$ is the mean *context embedding* of the context words.

$$s^* = \arg\max_i P(C \mid \mathbf{s}_i) = \arg\max_i \frac{1}{1 + e^{-\bar{\mathbf{c}}_c \cdot \mathbf{s}_i}}$$

The other choice is to maximize the similarity between the sense and the context. This time, we take the mean of the word embeddings of the context words.

$$s^* = \arg\max_i sim(\mathbf{s}_i, C) = \arg\max_i \frac{\bar{\mathbf{c}}_w \cdot \mathbf{s}_i}{|| \bar{\mathbf{c}}_w || \cdot || \mathbf{s}_i ||}$$

As a last trick, they compute the discriminative power of each word, taking only the $p$ most discriminative for their model. Their score function is either of the two functions we argmaxed above.

$$\max_i score(\mathbf{s}_i, c_j) - \min_i score(\mathbf{s}_i, c_j)$$

## Experiments and Evaluation

- Datasets: Crowdsourced collection and SemEval dataset.
- Evaluation Method: Different configurations on large datasets

  **TWSI Dataset**:

  - 1012 frequent nouns with 2.26 senses per word.
  - 145,140 Annotated sentences from Wikipedia.
  - Sense inventory with list of words.
  - 79% skewed sense distribution.
  - TWSI dataset and balanced subset of data.
  - Balanced subset: 6165 contexts with 5 contexts per sense

## Evaluation metrics

Mapping between TWSI senses and System provided sense inventory.

Calculation of precision and Recall.

Baselines to facilitate interpretation of results:

- Upper bound of induced Inventory
- MFS of the TWSI inventory
- MFS of induced inventory
- Random sense baseline

## Evaluation on SemEval Dataset

**Evaluation Method:** Compare performance to state of the art unsupervised WSD systems.

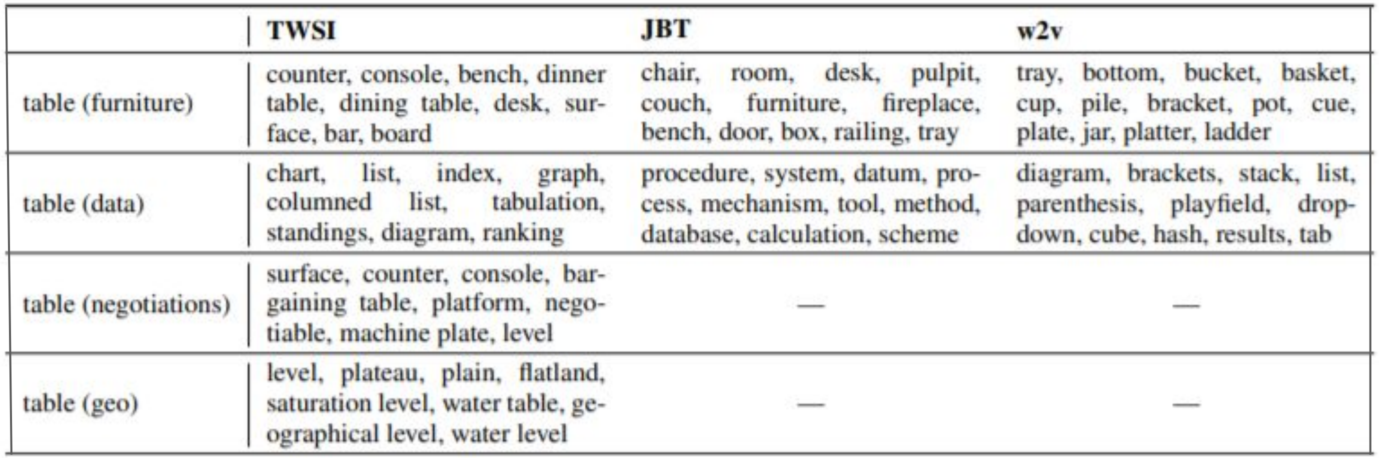
**SemEval Dataset:**

- 20 nouns, 20 verbs and 10 adjectives.
- 20-100 contexts per word equalling 4664.
- Cluster into groups corresponding to word sense.

**Evaluation Metrics:**

- Sense inventories: Jaccard Index, Tau, WNDCG.
- Cluster comparison measures: Fuzzy NMI and Fuzzy B-Cubed.

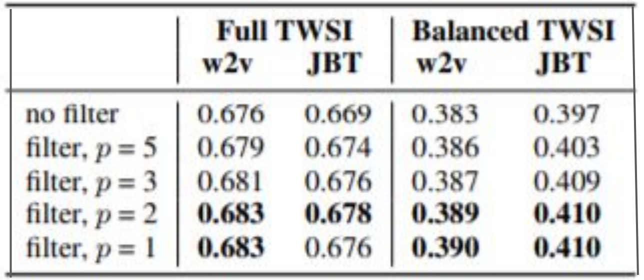
## Results



| | **TWSI** | **JBT** | **w2v** |
|---|---|---|---|
| table (furniture) | counter, console, bench, dinner table, dining table, desk, surface, bar, board | chair, room, desk, pulpit, couch, furniture, fireplace, bench, door, box, railing, tray | tray, bottom, bucket, basket, cup, pile, bracket, pot, cue, plate, jar, platter, ladder |
| table (data) | chart, list, index, graph, columned list, tabulation, standings, diagram, ranking | procedure, system, datum, process, mechanism, tool, method, database, calculation, scheme | diagram, brackets, stack, list, parenthesis, playfield, dropdown, cube, hash, results, tab |
| table (negotiations) | surface, counter, console, bargaining table, platform, negotiable, machine plate, level | — | — |
| table (geo) | level, plateau, plain, flatland, saturation level, water table, geographical level, water level | — | — |

Table 2: Word sense clusters from inventories derived from the Wikipedia corpus via crowdsourcing (TWSI), JoBimText (JBT) and word embeddings (w2v). The sense labels are introduced for readability.

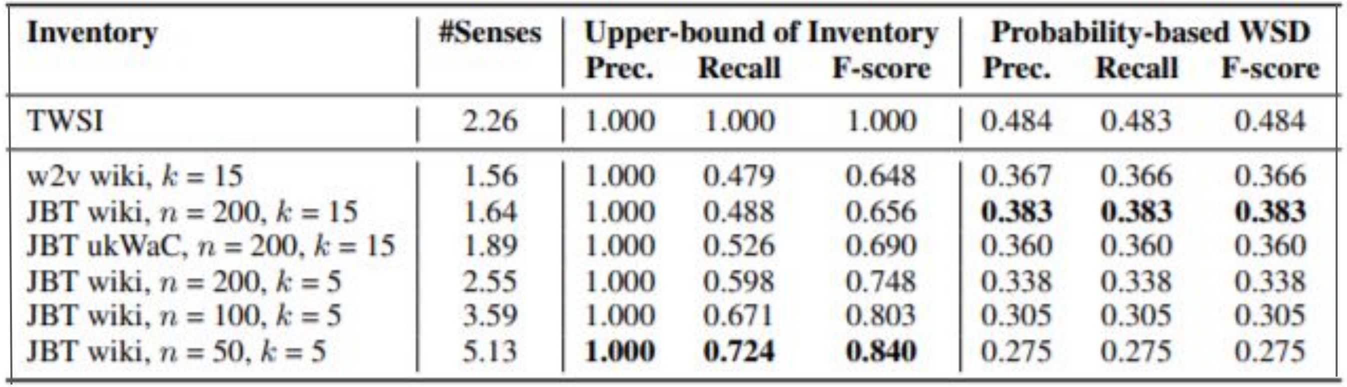
## Results



| | Full TWSI | | Balanced TWSI | |
|---|---|---|---|---|
| | **w2v** | **JBT** | **w2v** | **JBT** |
| no filter | 0.676 | 0.669 | 0.383 | 0.397 |
| filter, $p = 5$ | 0.679 | 0.674 | 0.386 | 0.403 |
| filter, $p = 3$ | 0.681 | 0.676 | 0.387 | 0.409 |
| filter, $p = 2$ | **0.683** | **0.678** | **0.389** | **0.410** |
| filter, $p = 1$ | **0.683** | 0.676 | **0.390** | **0.410** |

Table 3: Influence of context filtering on disambiguation in terms of F-score. The models were trained on Wikipedia corpus; the w2v is based on weighted pooling and similarity-based disambiguation. All differences between filtered and unfiltered models are significant ($p < 0.05$).

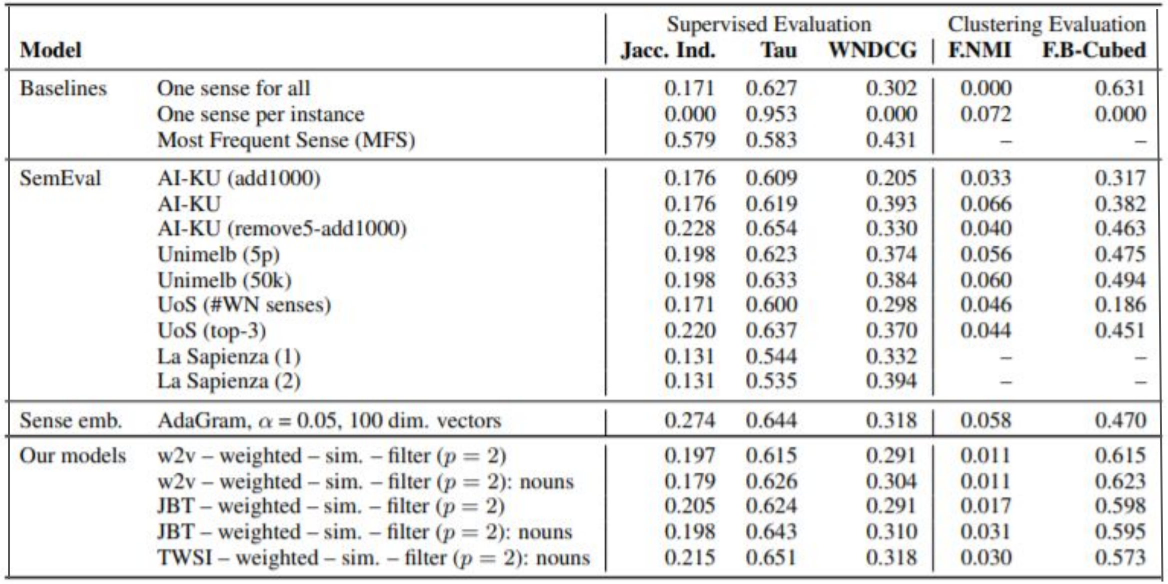
## Results



| Inventory | #Senses | Upper-bound of Inventory | | | Probability-based WSD | | |
|---|---|---|---|---|---|---|---|
| | | Prec. | Recall | F-score | Prec. | Recall | F-score |
| TWSI | 2.26 | 1.000 | 1.000 | 1.000 | 0.484 | 0.483 | 0.484 |
| w2v wiki, $k = 15$ | 1.56 | 1.000 | 0.479 | 0.648 | 0.367 | 0.366 | 0.366 |
| JBT wiki, $n = 200$, $k = 15$ | 1.64 | 1.000 | 0.488 | 0.656 | **0.383** | **0.383** | **0.383** |
| JBT ukWaC, $n = 200$, $k = 15$ | 1.89 | 1.000 | 0.526 | 0.690 | 0.360 | 0.360 | 0.360 |
| JBT wiki, $n = 200$, $k = 5$ | 2.55 | 1.000 | 0.598 | 0.748 | 0.338 | 0.338 | 0.338 |
| JBT wiki, $n = 100$, $k = 5$ | 3.59 | 1.000 | 0.671 | 0.803 | 0.305 | 0.305 | 0.305 |
| JBT wiki, $n = 50$, $k = 5$ | 5.13 | **1.000** | **0.724** | **0.840** | 0.275 | 0.275 | 0.275 |

Table 4: Upper-bound and actual value of the WSD performance on the sense-balanced TWSI dataset, function of sense inventory used for unweighted pooling of word vectors.

## Results



| Model | | Supervised Evaluation | | | Clustering Evaluation | |
|---|---|---|---|---|---|---|
| | | **Jacc. Ind.** | **Tau** | **WNDCG** | **F.NMI** | **F.B-Cubed** |
| Baselines | One sense for all | 0.171 | 0.627 | 0.302 | 0.000 | 0.631 |
| | One sense per instance | 0.000 | 0.953 | 0.000 | 0.072 | 0.000 |
| | Most Frequent Sense (MFS) | 0.579 | 0.583 | 0.431 | – | – |
| SemEval | AI-KU (add1000) | 0.176 | 0.609 | 0.205 | 0.033 | 0.317 |
| | AI-KU | 0.176 | 0.619 | 0.393 | 0.066 | 0.382 |
| | AI-KU (remove5-add1000) | 0.228 | 0.654 | 0.330 | 0.040 | 0.463 |
| | Unimelb (5p) | 0.198 | 0.623 | 0.374 | 0.056 | 0.475 |
| | Unimelb (50k) | 0.198 | 0.633 | 0.384 | 0.060 | 0.494 |
| | UoS (#WN senses) | 0.171 | 0.600 | 0.298 | 0.046 | 0.186 |
| | UoS (top-3) | 0.220 | 0.637 | 0.370 | 0.044 | 0.451 |
| | La Sapienza (1) | 0.131 | 0.544 | 0.332 | – | – |
| | La Sapienza (2) | 0.131 | 0.535 | 0.394 | – | – |
| Sense emb. | AdaGram, $\alpha = 0.05$, 100 dim. vectors | 0.274 | 0.644 | 0.318 | 0.058 | 0.470 |
| Our models | w2v – weighted – sim. – filter ($p = 2$) | 0.197 | 0.615 | 0.291 | 0.011 | 0.615 |
| | w2v – weighted – sim. – filter ($p = 2$): nouns | 0.179 | 0.626 | 0.304 | 0.011 | 0.623 |
| | JBT – weighted – sim. – filter ($p = 2$) | 0.205 | 0.624 | 0.291 | 0.017 | 0.598 |
| | JBT – weighted – sim. – filter ($p = 2$): nouns | 0.198 | 0.643 | 0.310 | 0.031 | 0.595 |
| | TWSI – weighted – sim. – filter ($p = 2$): nouns | 0.215 | 0.651 | 0.318 | 0.030 | 0.573 |

Table 5: The best configurations of our method selected on the TWSI dataset on the SemEval 2013 Task 13 dataset. The w2v-based methods rely on the CBOW model with 100 dimensions and context window size 3. The JBT similarities were computed using the Malt parser. All systems were trained on the ukWaC corpus.

## Conclusion

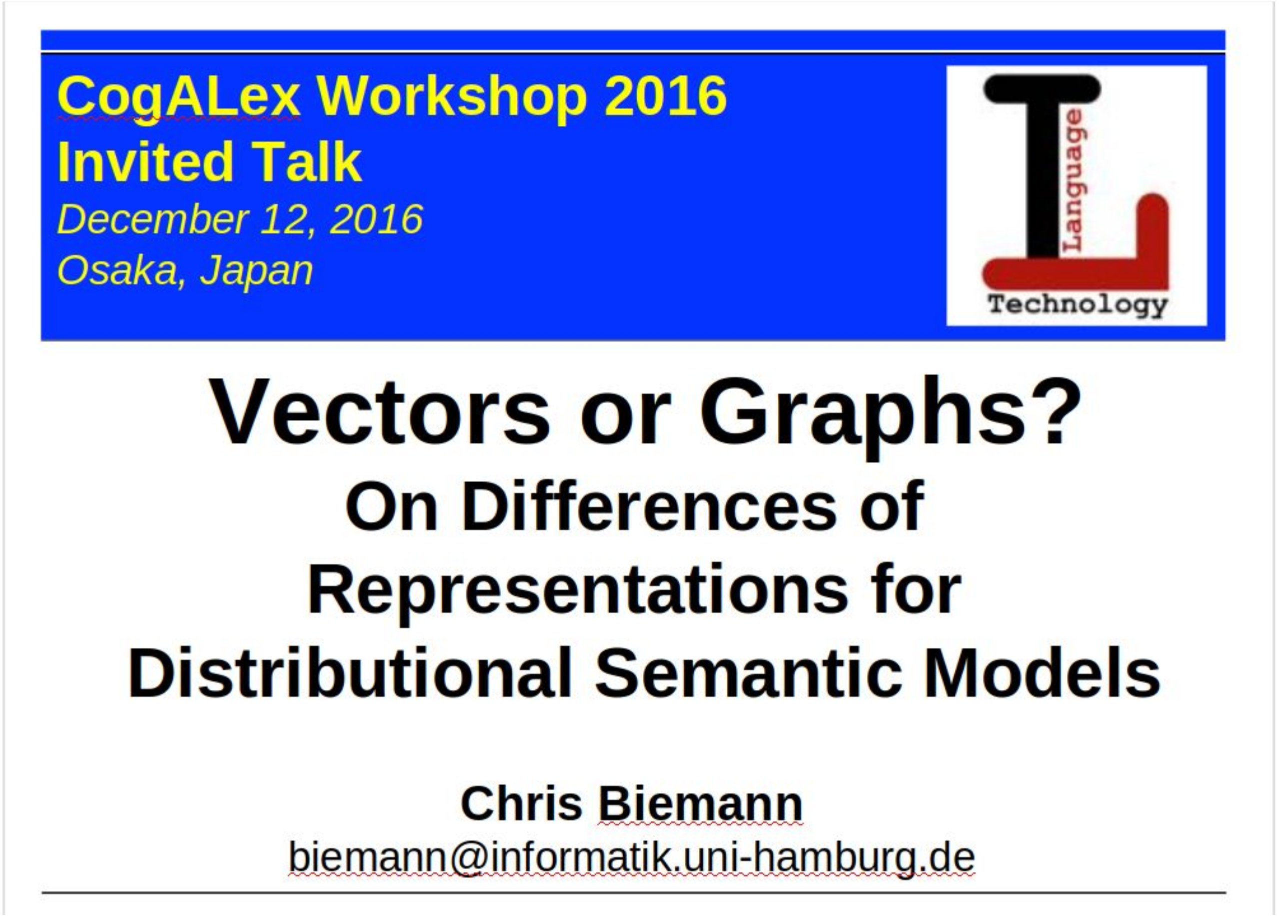
- Novel approach for learning **word sense embeddings**.
- Can use **existing word embeddings** as input.
- WSD performance **comparable to the state-of-the-art** systems.

**Source Code :** https://github.com/uhh-lt/sensegram

**Implementation Demo Online :**

http://ltbev.informatik.uni-hamburg.de/wsd/single-word

## Other work by the author:



CogALex Workshop 2016
Invited Talk
*December 12, 2016*
*Osaka, Japan*

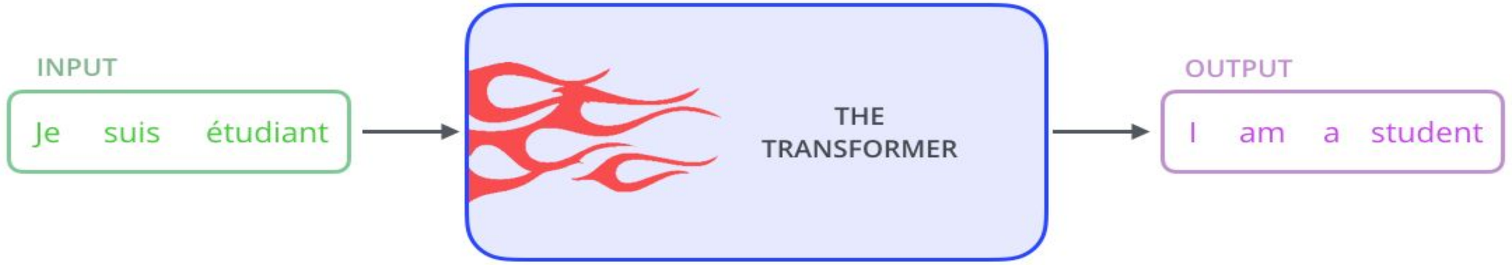
Language Technology

**Vectors or Graphs?**
**On Differences of Representations for Distributional Semantic Models**

**Chris Biemann**
biemann@informatik.uni-hamburg.de

## Future Developments in this field :

- In 2018 Google open sourced its state of the art language training model and data set BERT.
- It is first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus
- "Attention is all you need" paper  by Ashish Vaswani et al. of google uses attention mechanism of transformers to solve word sense disambiguation problems and has beaten all other existing models.



# References

https://arxiv.org/abs/1708.03390
https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/
https://machinelearningmastery.com/what-are-word-embeddings/
https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa
https://en.wikipedia.org/wiki/Artificial_neural_network
https://medium.com/@jayeshbahire/introduction-to-word-vectors-ea1d4e4b84bf
http://www.analytictech.com/networks/egonet.htm
https://en.wikipedia.org/wiki/Word2vec
https://en.wikipedia.org/wiki/Chinese_Whispers_(clustering_method)
https://en.wikipedia.org/wiki/Cluster_analysis
https://github.com/uhh-lt/sensegram