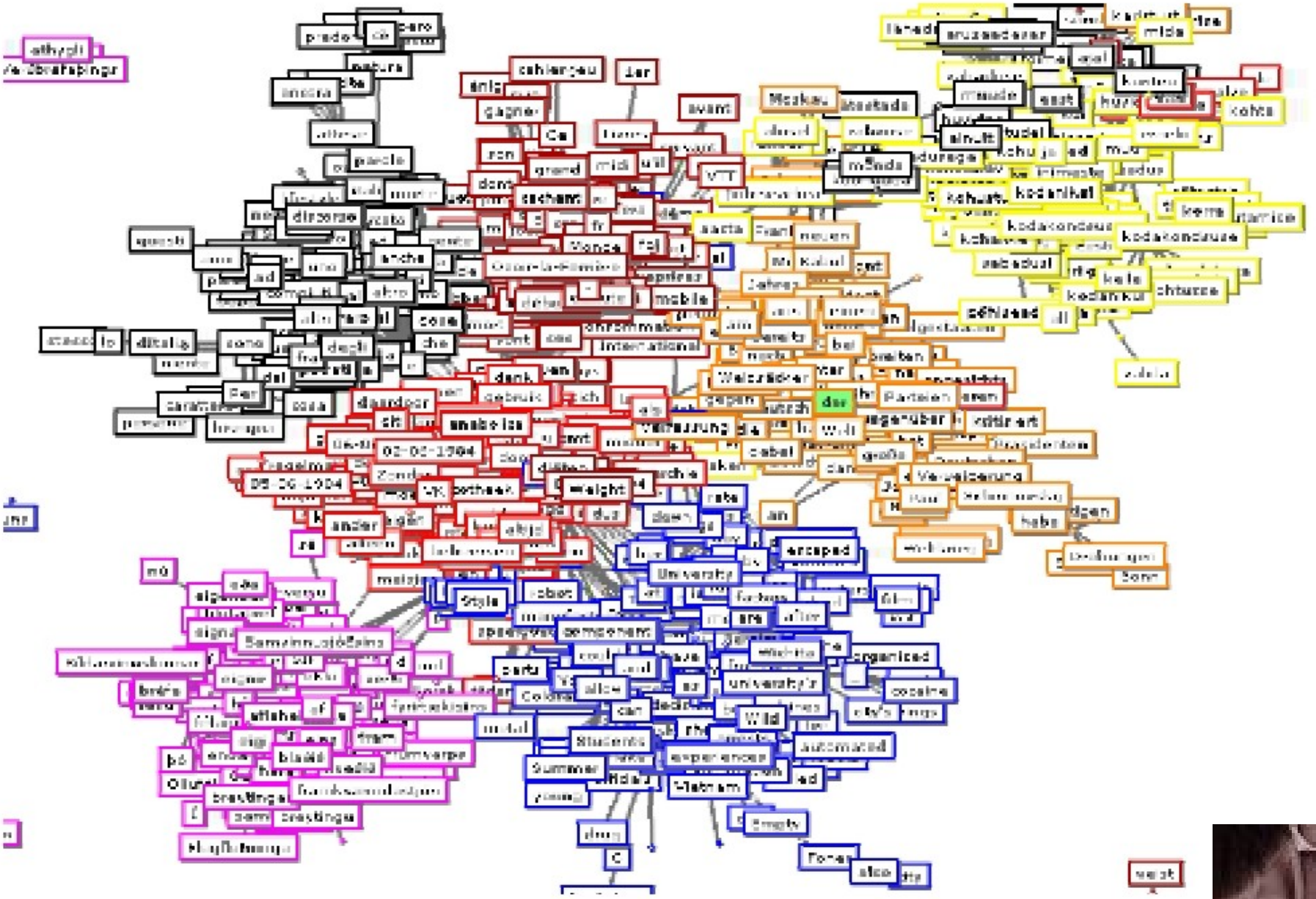
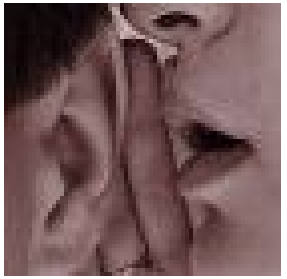


Chinese Whispers

Chris Biemann & Sven Teresniak: Chinese Whispers



Éva Mújdricza-Maydt

18.11.2010
Graph-based Methods for Natural Language Processing
Dozent: **Simone Paolo Ponzetto**

Ruprecht-Karls-Universität Heidelberg
Seminar für Computerlinguistik

Outline

Chinese Whispers Algorithm
Experiments
Demo

Remark: Resources are denoted at the end of the slides, but citations in the slides are not quoted by apostrohps.

Motivation

- only minimal linguistic knowledge
  → **knowledge-free** and **unsupervised** methods

- optimal representation of language data
  → **graph** models
    - no storage of zero-values or not relevant values

- possible tasks: **clustering** of objects: in NLP:
    - language separation
    - part-of-speech tagging
    - word sense induction
    - ...

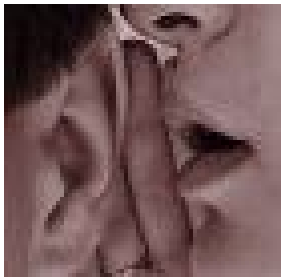
Chinese Whispers

- Chinese Whispers is a simple, but efficient **randomized** graph-clustering algorithm.
  - works on a **weighted**, **undirected** co-occurrence graph
    - *nodes*: words
    - *weighted vertices*: relations between words ~ co-occurrence
  - handles also **very large graphs**
  - run-time **complexity: linear** in the number of edges
  - output: a **non-hierarchical** (flat) **partitioning** of the graph

- Clustering is the process of grouping together objects based on their **similarity** to each other.
  - similarity → **weights of the vertices** according to the **co-occurrence significance** of two words

Chinese Whispers: Naming

- It is motivated by the game, where participants whisper words to each other.
  - The game's goal is to arrive at some funny derivative of the original message by passing it through several noisy channels.



- CW algorithm: the **same message** = the **same group** of elements
- ~ The **nodes** of the graph **whisper their label to each other**, until every node agrees with its adjacent nodes on some label.

Chinese Whispers: Input

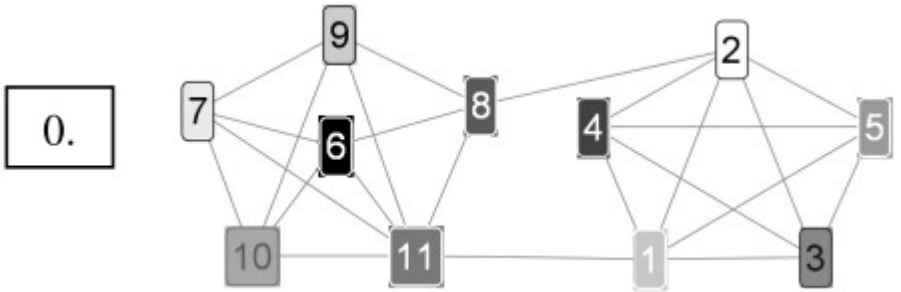
- sentence separated, tokenized text
  - tokens → *nodes*
  - co-occurrence significance (above a threshold) between tokens → *vertices*

- context range in co-occurrence calculation:
  - **sentence-based co-occurrences**: all (relevant) words in a sentence
  - **neighbor-based co-occurrences**: words in an x-word window left and right from the current word

- **co-occurrence significance**: according to *likelihood ratio*: Ted Dunning (1993): Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*. 19(1), pp. 61-74.

Chinese Whispers: Algorithm (v1)

```
Initialization:
    Assign different labels to every node in the graph;

For iteration i from 1 to total_iterations {

    For each word w in the graph, randomized order {
        label of w = highest ranked label in neighbourhood of w;


    }
}
```

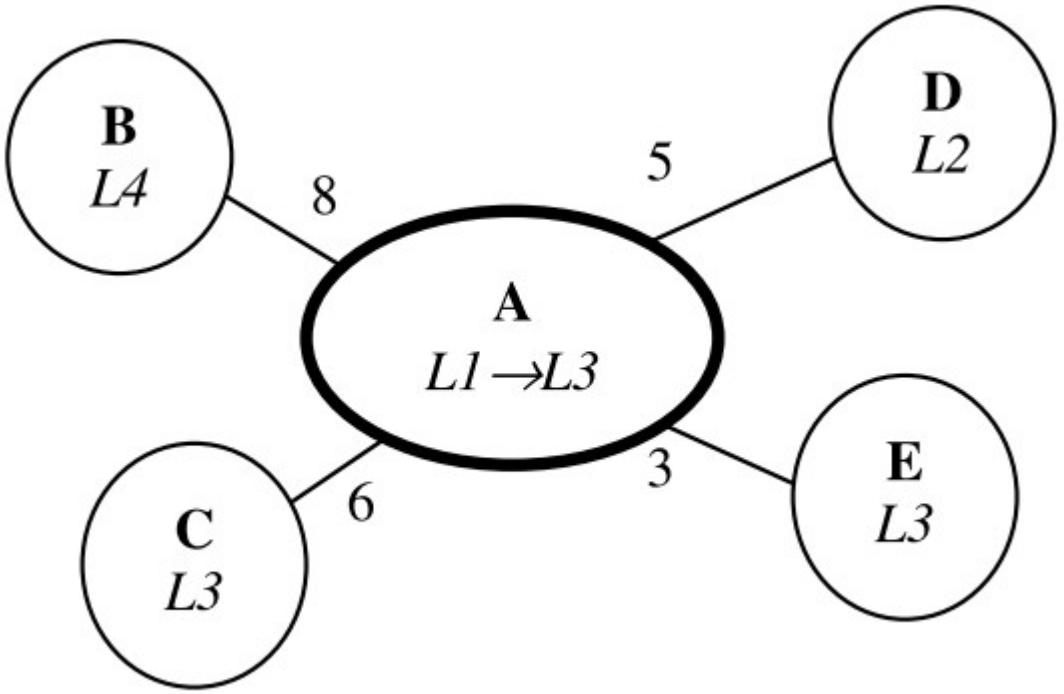


Chinese Whispers: Algorithm (v1)

```
Initialization:
    Assign different labels to every node in the graph;

For iteration i from 1 to total_iterations {

    For each word w in the graph, randomized order {
        label of w = highest ranked label in neighbourhood of w;


    }
}
```

- Main loop: Change the class labels of the current node according to the weights of their connected nodes (**neighbors**).
- Neighbor labels will be **rank**ed: For each class label in the neighborhood, the sum of the weights of edges to the word in question is taken as score for ranking. → The current node gets the label of highest score.

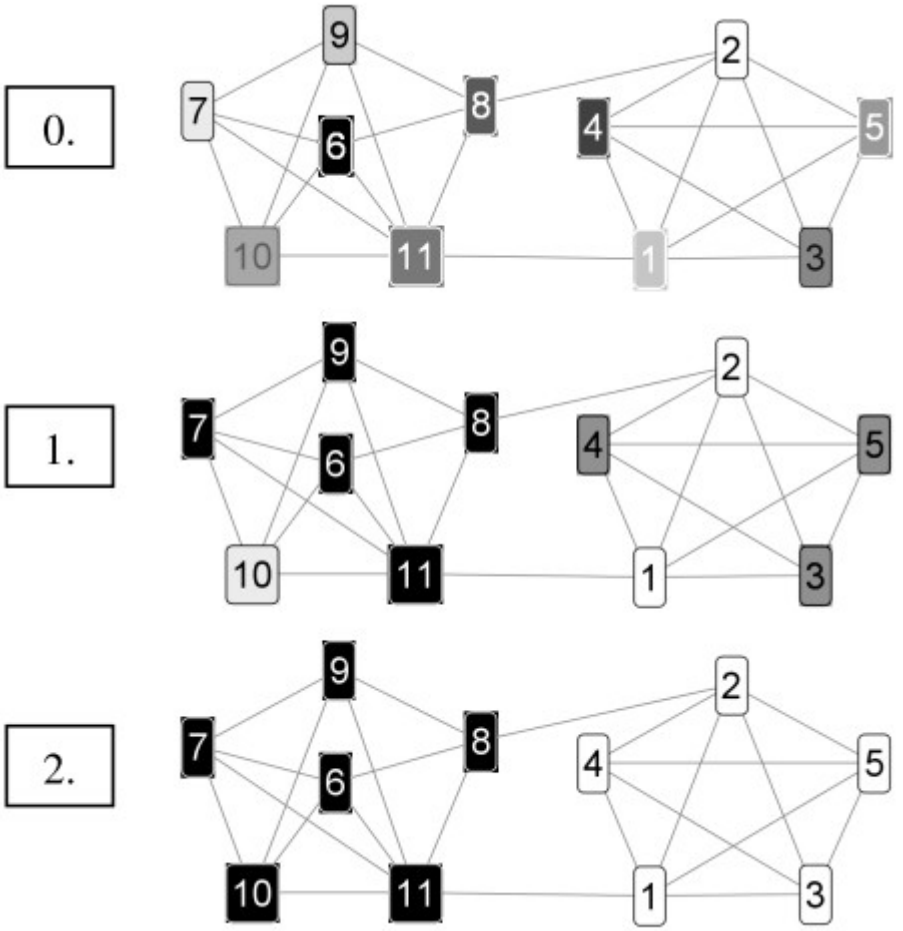
Chinese Whispers: Algorithm



- **update according local neighborhood**
- **update „top“**: The strongest class is the class whose sum of edge weights to the current node is maximal.
- if multiple strongest classes → random choice

e.g. for current node A
- strength(L3) = 6+3 = 9
- strength(L4) = 8 → change label L1 to L3
- strength(L2) = 5

Chinese Whispers: Algorithm



Chinese Whispers: Algorithm (v2)

```
Initialization:
    Assign different labels to every node in the graph;

For iteration i from 1 to total_iterations {
    mutation_rate = 1/(i²);
    For each word w in the graph, randomized order {
        label of w = highest ranked label in neighbourhood of w;
        with probability mutation_rate:
                label of w = new class label;
    }
}
```

- Difference: A word probably gets a random, yet unused label.
  - Reason: to avoid too greedy label updates in small graphs – Without the mutation, the whole graph would tend to get only one label.

Chinese Whispers: Updating labels

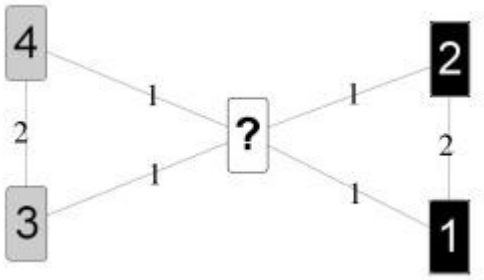
- Updating the class labels:
  - **continuous** / immediately: changes in the same iteration
  - **stepwise**: in the next iteration

  → Regions of the same class stabilize during the iteration and grow until they reach the border of a stable region of another class.

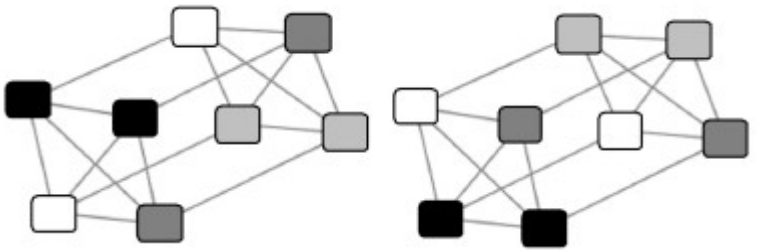
- Nodes that are not connected by any edge are discarded from the clustering process, which possibly leaves a portion of nodes unclustered.
- Convergence: formally, the algorithm does not converge → **stop criterion**
  - no more changes in the label set
    - after only a small number (< 20) of iterations, even in graphs of 1 million nodes and more
    - Weighted graphs converges faster than unweighted.
  - defined number of iterations

Chinese Whispers: Updating labels

- **Tie** (here for the middle node): equal strength for each possible label



- → converge to a pair of **oscillating** class matrices.
  - probably caused by the stepwise update of the class matrix



Chinese Whispers: Output

- The result of CW is a **hard partitioning** (~ one node gets only one label) of the given graph.
  - The **number of partitions emerges in the process**.
  - It is possible to obtain a **soft partitioning** (~ possibly more than one label per node) by assigning a class distribution to each node.
- CW is **non-deterministic**: The clustering the same graph several times can result in different outcomes.
- With increasing iterations, clusters become **self-preserving**: If a strongly connected cluster happens to be homogeneous with respect to class labels, it will never be infected by a few connections from other clusters.
- Comparison with Min-Cut (Wu & Leahy 1993):
  - CW does not find an optimal hierarchical clustering, but yields a non-hierarchical (flat) partition.
  - CW does not require any threshold as input parameter.

Experiments

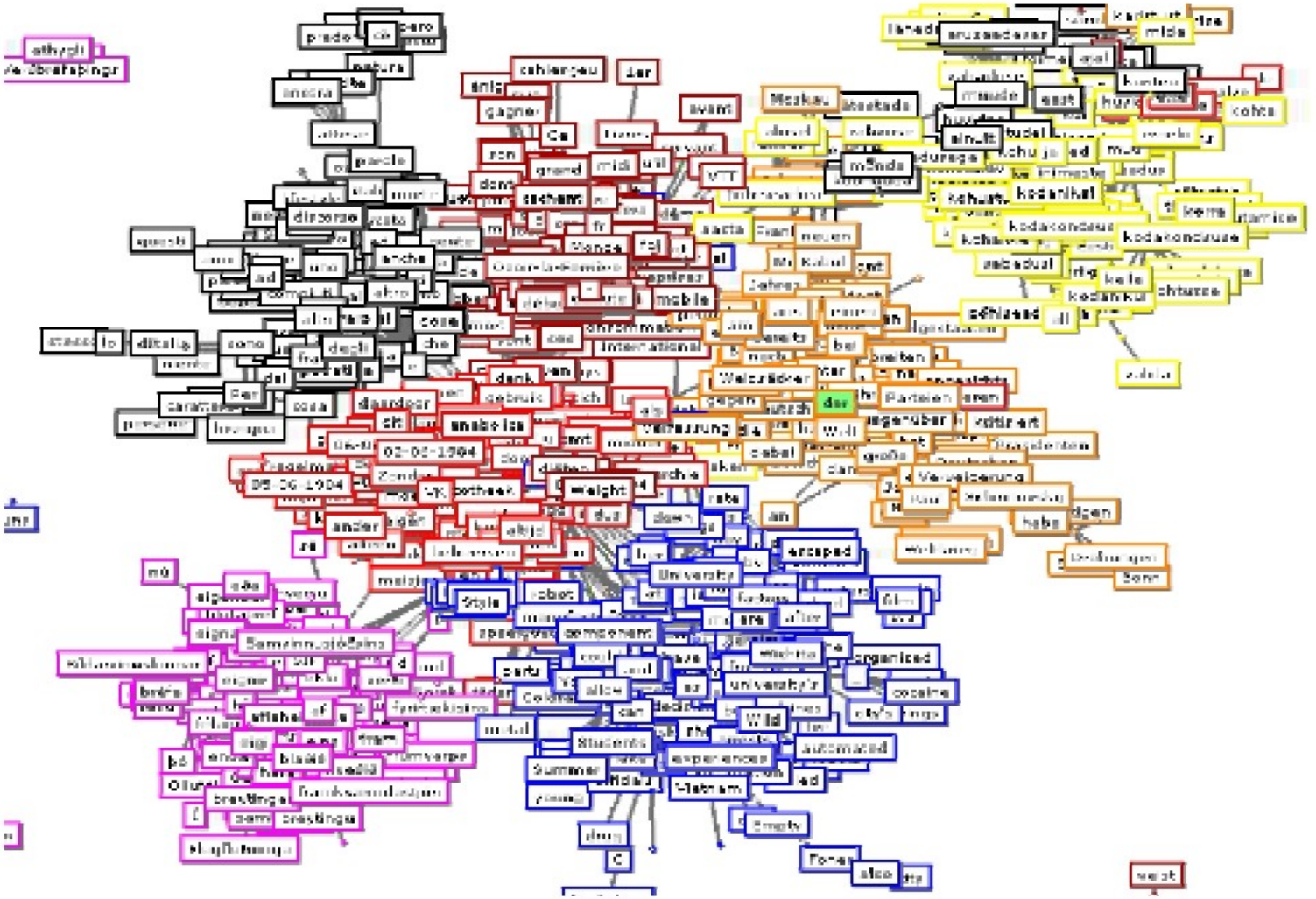
- on unweighted, synthetic graphs
- Experiments:
  - **language separation**
  - **word class acquisition**

Experiments: Language separation

- Co-occurrence type: sentence-based
- Corpus: equal fractions of **seven European languages** (*Dutch, Estonian, English, French, German, Icelandic and Italian*)
  - 100, 200, 500, 1’000, 5’000, 10’000, 50’000 and 100’000 sentences of each language
  - ambiguous words: used in more than one language → members of several graphs, connecting them

  → By CW-partitioning, the graph is split into its monolingual parts.
- Minimal size of a cluster: containing at least 1.8% of all words in the co-occurrence graph. All smaller clusters were regarded as noise.
- Language-ambiguous words are assigned to only one language (hard partitioning).

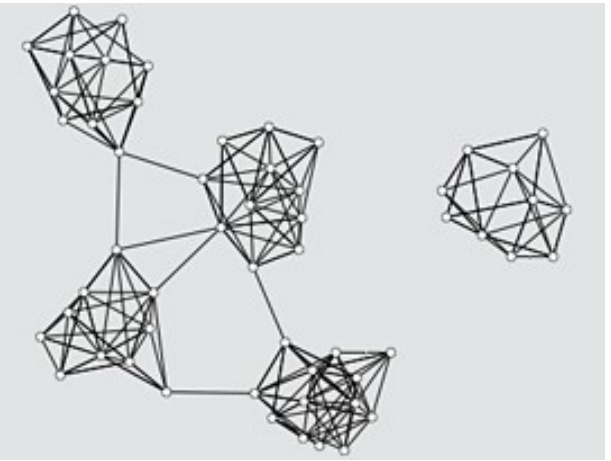
Experiments: Language separation



Experiments: Language separation
– small world graphs

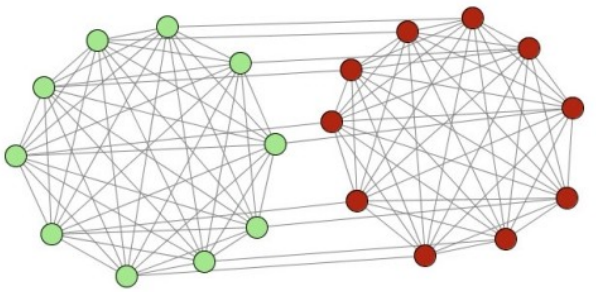
- The multilingual graph looks like a small world graph.
- **Small-world network/graph**:
  The typical distance *L* between two randomly chosen nodes (~ the shortest path length between the two nodes) grows proportionally to the logarithm of the number of nodes *N* in the network: $L(N) \propto \log(N)$

  = **Most pairs of nodes will be connected by at least one short path**.



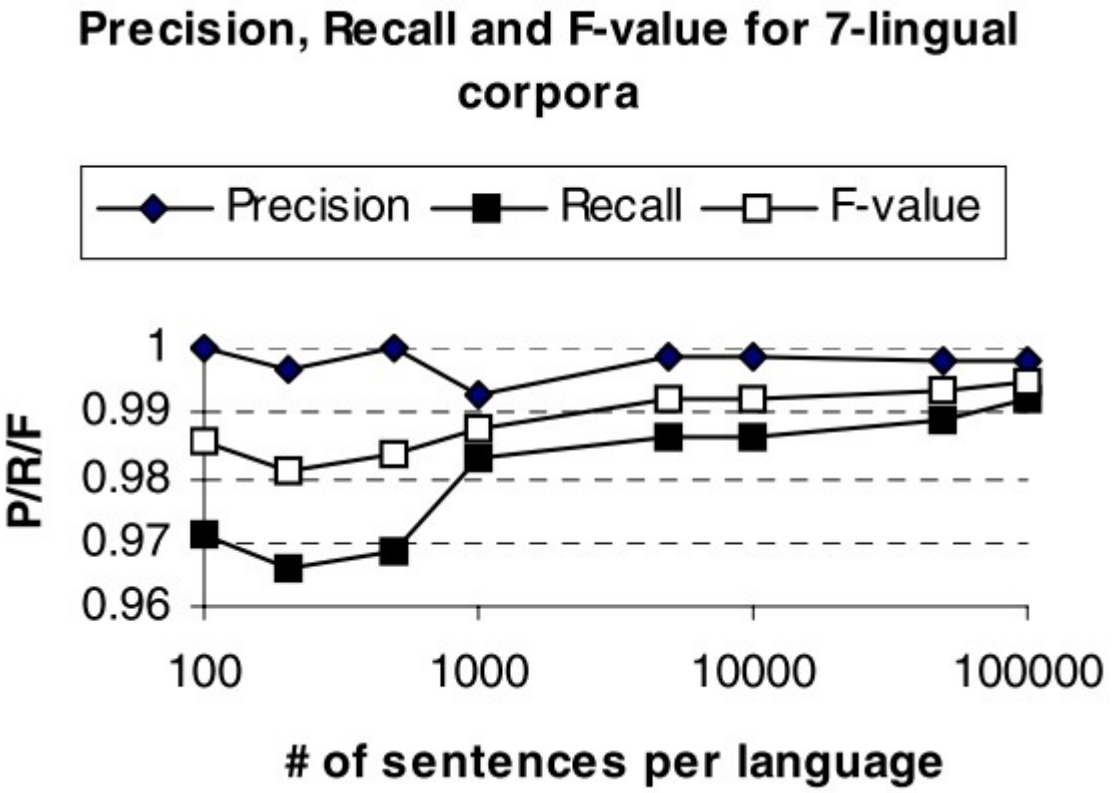
→ The mean-shortest path length is small.

→ contains **cliques** and near-cliques: sub-networks which have connections between almost any two nodes within them.



Experiments: Language separation

- Output: seven large clusters + many very small clusters (noise)
- **Best** results: **English**: Precision > 99.9%, Recall > 98.1% for all experiments
- Worst results: Estonian: Precision > 99.3%, Recall 86.7–93.7 % for all experiments



Experiments: Language separation
– error analysis

- Estonian corpus: legal texts are difficult to cluster.
  - about 3% of the texts are dates or law paragraph ciphers: e.g. *“10.12.96 jõust.01.01.97 - RT I 1996 , 89 , 1590.”*

- In all languages difficult:
  - enumerations of sport teams
  - short headlines
  - proper names → many company names were usually classified as English
  - bilingual sentences: e.g. *“Frönsku orðin "cinéma vérité" þýða "kvikmyndasannleikur".”* (Icelandic)

Experiments: Language separation
– error analysis

- Experiments with fewer than 100 sentences resulted in more than 7 (ca. 11) clusters.
  - Reason: The significant co-occurrences were too small in numbers and too noisy in quality.

- Similar languages are only slightly more difficult to separate than languages of different families.
  - (Except corpora with too large bias towards the main language.)
  - Even German dialects could be identified in large German corpora.

Experiments: Word Class Acquis.

- Corpus: British National Corpus (BNC), excluding the most frequent 2000 words.
- Edges: between words sharing at least four left and right neighbors.
- Golden standard: Lexicon with the most frequent tag for each word in BNC

- The largest clusters:

size	*tags:count*	*sample words*
18432	NN:17120 AJ: 631	secret, officials, transport, unemployment, farm, county, wood, procedure, grounds, ...
4916	AJ: 4208 V: 343	busy, grey, tiny, thin, sufficient, attractive, vital, ...
4192	V: 3784 AJ: 286	filled, revealed, experienced, learned, pushed, occurred, ...
3515	NP: 3198 NN: 255	White, Green, Jones, Hill, Brown, Lee, Lewis, Young, ...
2211	NP: 1980 NN: 174	Ian, Alan, Martin, Tony, Prince, Chris, Brian, Harry, Andrew,

Experiments: Word Class Acquis.

- In total, CW produced 282 clusters, of which 26 exceed a size of 100.
- The weighted average of **cluster purity** (i.e. the number of predominant tags divided by cluster size): 88.8%.
- As POS tagging is not a system for its own sake, but serves as a preprocessing step for systems building upon it, the names and the number of categories are very often not important.

Conclusions

- Unsupervised clustering of language data is possible and works with the same high accuracy as the well-known supervised approaches.
  - Only requirement on data: **word** boundaries and **sentences** for the calculation of significant co-occurrences.
- CW:
  - **parameter-free, unsupervised method**
  - is **robust** with respect to the number and the mass distribution of the involved data
  - almost perfect results, reliable results **also with small data sets**
  - is capable of **handling very large graphs** in **reasonable time**
  - **chooses the number of classes on its own** and can **handle clusters of different sizes**
  - **non-deterministic**

Demo: GUI

- The implementation is written in Java and platform independent.

  http://wortschatz.uni-leipzig.de/~cbiemann/software/CW.zip

- Main panel:
  - input: the source of the graph to be clustered: files (nodes-file + edges-file) or database
  - destination of the output: files or database
  - displaying the outcome of the algorithm: interactive graph window or diagram (latter showing quantitative characteristics of the result)
- Graph:
  - only **undirected** graphs, i.e. weight(n_1, n_2) = weight(n_2, n_1)
  - edge weights: positive integer values
    - if all weights have the same value, then the graph is unweighted

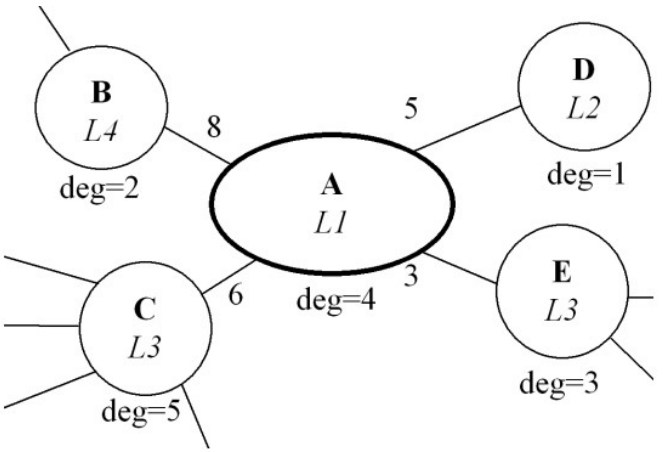
Demo: Output

- Output as **graph**:
  - views: labeled and unlabeled nodes
  - the size of the nodes is changeable
  - zoom function
  - shows results of each iteration step by step
  - Different colors represent different classes. – if mouse over a node, you get information on to what extent the node belongs to different classes
  - subgraph view

- Output as **diagram**: draws the distribution of cluster sizes in a graph.
  - Also a comparison of several algorithm options in the diagram is possible.

Demo: Options

- Custom changing of algorithm parameters (expert panel):
  - minimum/maximum degree of nodes
  - minimum edge weight threshold
  - number of iterations
  - algorithm strategy, mutation
  - update strategy



- Algorithm strategies:
  - **top**: sums over the weights of neighborhood's classes
  - **dist** log, dist nolog: downgrades the influence of a neighboring node by its degree (number of edges of node): weight / degree
  - **vote**: the same as top but needs a minimum threshold for a class change to take place

Demo: Options

- Update strategies:
  - **stepwise**: The updates are not performed immediately, but take effect in the next iteration.
  - **continuous**: A node can spread its class in the same iteration as it received it. The continuous option converges faster.

- Default setting: *top, keep class rate 0, mutation constant with rate 0* (no mutation) and *continuous update*.

References

- Chris Biemann & Sven Teresniak (2005): Disentangling from Babylonian Confusion – Unsupervised Language Identification. *Proceedings of CICLing-2005, Mexico City, Mexico,* pp. 762-773.
- Chris Biemann (2006): Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. *Proceedings of Textgraphs-06, New York, USA*, pp. 73-80.
- Chris Biemann (2007): Unsupervised Natural Language Processing using Graph Models. *Proceedings of the NAACL-HLT 2007 Doctoral Consortium, Rochester*, pp. 37-40.

- Chinese Whispers: http://wortschatz.uni-leipzig.de/~cbiemann/software/CW.zip
- Chinese Whispers User Manual: http://wortschatz.informatik.uni-leipzig.de/~cbiemann/software/CW.html
- Small-world network: http://en.wikipedia.org/wiki/Small-world_network; http://burak-arikan.com/wp-content/uploads/2008/09/graph-smallworld-scalefree.png