A look inside Babelfy: Examining the bubble

Minh Le

Filip Ilievski
Reproducibility

“Reproducibility is a defining feature of science, but the extent to which it characterizes current research is unknown.”

(Estimating the reproducibility of psychological science, 2015)
“According to the replicators’ qualitative assessments, only 39 of the 100 replication attempts were successful.”

(http://www.nature.com/news/over-half-of-psychology-studies-fail-reproducibility-test-1.18248)
Replication is h.a.r.d.

“According to the replicators’ qualitative assessments, only 39 of the 100 replication attempts were successful.”

“But, whether a replication attempt is considered successful is not straightforward.”

([http://www.nature.com/news/over-half-of-psychology-studies-fail-reproducibility-test-1.18248](http://www.nature.com/news/over-half-of-psychology-studies-fail-reproducibility-test-1.18248))
Everyone has a replication story to tell

... And here is ours!
Babelfy

- Solution published in 2014
Babelfy

- Solution published in 2014
- State-of-the-art reported performance
Babelfy

- Solution published in 2014
- State-of-the-art reported performance
- Great algorithm
  - Multilinguality
  - Joint decision making
  - Combined knowledge from multiple sources
    - NLP (EL, WSD)
    - Semantic Web (BabelNet)
Babelfy

But: No open source available
Idea: Reimplement

- No open source available
- Curiosity how the algorithm works in detail
- Accelerate research by opening the source code
Algorithm

1. Preprocessing (run once)
   1.1. Input: BabelNet graph (undirected, labeled graph of concepts and entities)
   1.2. Re-weight the graph
   1.3. Random walk with restart
   1.4. Output: set of related concepts and entities

2. Disambiguation (for each document)
   2.1. Generate candidates
   2.2. Construct referent graph
   2.3. Apply Densest subgraph algorithm
   2.4. Disambiguate with a threshold
1) Graph -> Concept set
Algorithm

1. Preprocessing (run once)
   1.1. Input: BabelNet graph (undirected, labeled graph of concepts and entities)
   1.2. Re-weight the graph
   1.3. Random walk with restart
   1.4. Output: set of related concepts and entities

2. Disambiguation (for each document)
   2.1. Generate candidates
   2.2. Construct referent graph
   2.3. Apply Densest subgraph algorithm
   2.4. Disambiguate with a threshold
2) Disambiguation

Thomas and Mario are strikers playing in Munich.
2) Disambiguation:
Graph construction

Thomas and Mario are strikers playing in Munich.
2) Disambiguation: Densest subgraph

Thomas and Mario are strikers playing in Munich.
Experimental setup

We ran experiments on:

- a University server
- 16/32GB RAM
- 1TB disk space
- 16 cores CPU @ 2.60 Ghz
Experiments

KORE-50 dataset: 50 ‘difficult’ sentences
1. Preprocessing (run once)
   1.1. Input: BabelNet graph (undirected, labeled graph of concepts and entities)
   1.2. Re-weight the graph
   1.3. Random walk with restart
   1.4. Output: set of related concepts and entities

2. Disambiguation (for each document)
   2.1. Generate candidates
   2.2. Construct referent graph
   2.3. Apply Densest subgraph algorithm
   2.4. Disambiguate with a threshold

Execution time: KORE-50

- Preprocessing: 3.5 days
- Disambiguation: 10.3 hours
## Results: KORE-50

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognition</td>
<td>0.82</td>
<td>0.77</td>
<td>0.79</td>
</tr>
<tr>
<td>Disambiguation</td>
<td>0.41</td>
<td>0.40</td>
<td>0.40</td>
</tr>
<tr>
<td>Disambiguation (reported by Babelfy)</td>
<td>/</td>
<td>/</td>
<td>0.715</td>
</tr>
</tbody>
</table>
Discussion

- significant accuracy mismatch
  - Reported F1-score: 71.5%
  - Our F1-score: 40%
Discussion

- significant accuracy mismatch
- algorithm gaps that matter
  - partial matching for entities, exact matching for concepts: but how do we know if something is an entity?
  - different parameters for EL and WSD
  - parameters tuned per dataset :-(
  - what happens if the linking conference is too low?
Discussion

- significant accuracy mismatch
- algorithm gaps that matter
- computationally quite heavy

For KORE-50:
- 3.5 days to preprocess data
- 17 mins to generate candidates
- 10 hours to disambiguate
Discussion

- significant accuracy mismatch
- algorithm gaps that matter
- computationally quite heavy
- ok, implementation could be improved
  - currently we use Python + MongoDB + up to 32GB RAM
Conclusions

Combining knowledge is an attractive idea but coping with that knowledge is additional challenge!

Babelfy is a great effort which should have an open source version which should be reproducible

Given:
the detail in the paper & the computational costs, reproducing is very difficult
Do we know what we think we know?

Reproducibility is a defining feature of science

Is reproducibility appreciated in the real-world?

Scientific claims should gain trust and credence by replicability of their supporting evidence
Join us!

https://github.com/filipdbrsk/BabelfyReimplementation
Appendix
Discussion on computation

- a lot of requests for generating candidates
- a lot of candidates
- big graphs
- computationally super heavy (better implementation needed):

For KORE-50:

- 3.5 days to preprocess data
- 17 mins to generate candidates
- 10 hours to disambiguate
Discussion on algorithm

partial vs exact matching: when to use what

partial matching for entities, but how do we know if something is an entity?

different hyperparameters for EL & WSD -> very confusing

what happens if the confidence is too low -> unreported

tuning of the accuracy per dataset: especially on partial vs full matching
Discussion on implementation

- Python is a poor choice for speed
  - GIL (Global Interpreter Lock) → no multithreading
- MongoDB is scalable but not fast enough
- More RAM
The algorithm in a nutshell

Finding semantic signature:

- Synset graph: an edge between any pair of synsets having a relation of any type
- Structural weighting: edge weight is proportional to the number of directed triangles
- Semantic signature: random walk with restart
The algorithm in a nutshell

Finding candidates:

- tight integration with the background knowledge
- every phrase which contains a noun is checked for existence in BabelNet
The algorithm in a nutshell

Disambiguating:
Implementation

Structural weighting: 1st attempt

for u in vertices
    for v in adjacent vertices of u
        for w in adjacent vertices of v
            is u adjacent to w?

→ O(E^2)
Implementation

Structural weighting: 2nd attempt

reverse all edges
for u in vertices
    for v in adjacent vertices of u
        intersection(adjacent vertices of v,
                     reversed-adjacent vertices of u)
→ O(E)