Wembedder

Finn Årup Nielsen

DTU Compute
Technical University of Denmark

28 October 2017

How do we find related items in Wikidata?

With Wikidata Query Service?

Genetically associated diseases

Other diseases with reported genetic association via genes, ordered according to number of co-associated genes.

Show 25 entries Search:

Count	Disease	Genes
14	bipolar disorder	NPAS3 // CACNA1C // ANK3 // MSRA // PTPRN2 // IFT88 // KCNMB2 // PHF8 // CNTNAP2 // ERC2 // COMMD10 // RIN2 // NLRC5 // MYO18B
5	obesity	PTPRN2 // CNTNAP2 // CTNNA3 // RIN2 // CSMD1
5	mental depression	NPAS3 // CDH13 // RORA // IFT88 // MYO18B
4	periodontitis	CDH13 // ERC2 // CSMD1 // NKAIN2
4	Alzheimer	RELN // CNTNAP2 // CSMD1 // NKAIN2
3	asthma	RORA // NOTCH4 // CTNNA3
2	coronary artery disease	TNIK // CSMD1
2	amyotrophic lateral sclerosis	ANK3 // KCNMB2
2	morbid obesity	TCF4 // SDCCAG8
2	major depressive disorder	CACNA1C // ANK3
2	multiple sclerosis	RELN // CSMD1
1	celiac disease/ allergic disorder	NKAIN2
1	smallpox	CSMD1
1	intracranial aneurysm	CNNM2
1	nicotine dependence	CTNNA3

Count some form of co-occurences with a SPARQL query in the Wikidata Query service.

Scholia is doing this for diseases and proteins with tailor-made SPARQL. Here for the disease schizophrenia.

Shows genetically associated diseases via the P2293 (genetic association) property.

Textual similarity in values and properties?

Textual similarity in values and properties?

?

Textual similarity in values and properties?

?

Bag-of-properties and bag-of-property-and-values?

with tfidf-like normalization?

and then standard information retrieval methods. . . (inner product, cosine similarity)

Textual similarity in values and properties?

?

Bag-of-properties and bag-of-property-and-values?

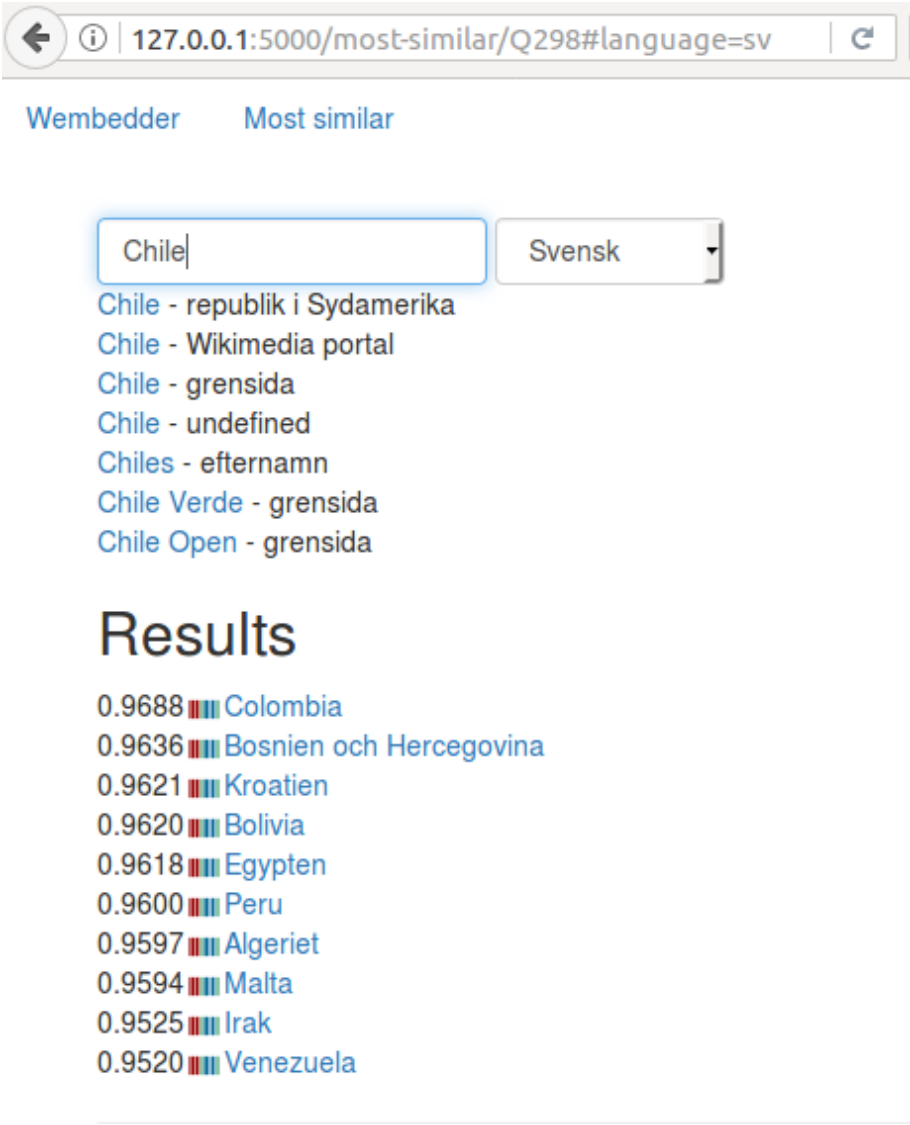
with tfidf-like normalization?

and then standard information retrieval methods. . . (inner product, cosine similarity)

"Propositionalization" (Ristoski and Paulheim, 2016)

But this is not what we are doing here

Wembedder



Wembedder: Web service with graph embedding.

Runs from Wikimedia Toolforge: https://tools.wmflabs.org/wembedder/

Multilingual query via Wikidata API and Javascript.

Call to Wembedder web service computes most similar Wikidata entities and returns an ordered list.

Multilingual labels from Wikidata API.

There is also an API

https://tools.wmflabs.org/wembedder/api/most-similar/Q2

JSON | Raw Data | Headers

Save Copy


```
model_metadata:
  filename:          "wikidata-20170613-truthy-BETA-cbow-size=100-window=1-min_count=20"
most_similar:
  0:
    item:            "Q313"
    similarity:      0.9384167790412903
  1:
    item:            "Q405"
    similarity:      0.9379734396934509
  2:
    item:            "Q319"
    similarity:      0.9377537369728088
  3:
    item:            "Q111"
    similarity:      0.9374039173126221
  4:
    item:            "Q3143"
    similarity:      0.9325475692749023
  5:
    item:            "Q2565"
    similarity:      0.9309490919113159
  6:
    item:            "Q844672"
    similarity:      0.9290982484817505
  7:
    item:            "Q308"
    similarity:      0.9283913969993591
  8:
    item:            "Q3169"
    similarity:      0.9253381490707397
  9:
    item:            "Q3359"
    similarity:      0.923356294631958
size_of_vocabulary:  609471
```


Wembedder API.

The API returns JSON in a simple format based on a query on the Q identifier.

URL schema: `/api/most-similar/Q2`

Similarity computation also available in the API, e.g.: `/api/similarity/Q2013/Q80`

. . . and the “word” vectors: `/api/vector/Q80`

Wembedder's simple approach 1

Truthy dumps → Quick statement-like → Gensim Word2Vec

Wembedder's simple approach 1

Truthy dumps → Quick statement-like → Gensim Word2Vec

Convert a line from `wikidata-20170613-truthy-BETA.nt.bz2` truthy dump:

```
<http://www.wikidata.org/entity/Q3719>
  <http://www.wikidata.org/prop/direct/P17>
  <http://www.wikidata.org/entity/Q30>
  .
```

to a quickstatement-like "3-word-sentence" representation

```
Q3719 P17 Q30
```

This latter file, `wikidata-20170613-truthy-BETA.trigrams`, is around 2 gigabytes uncompressed.

Wembedder's simple approach 2

Submit the trigram file to a standard Gensim Word2vec model and do a very short graph walk.

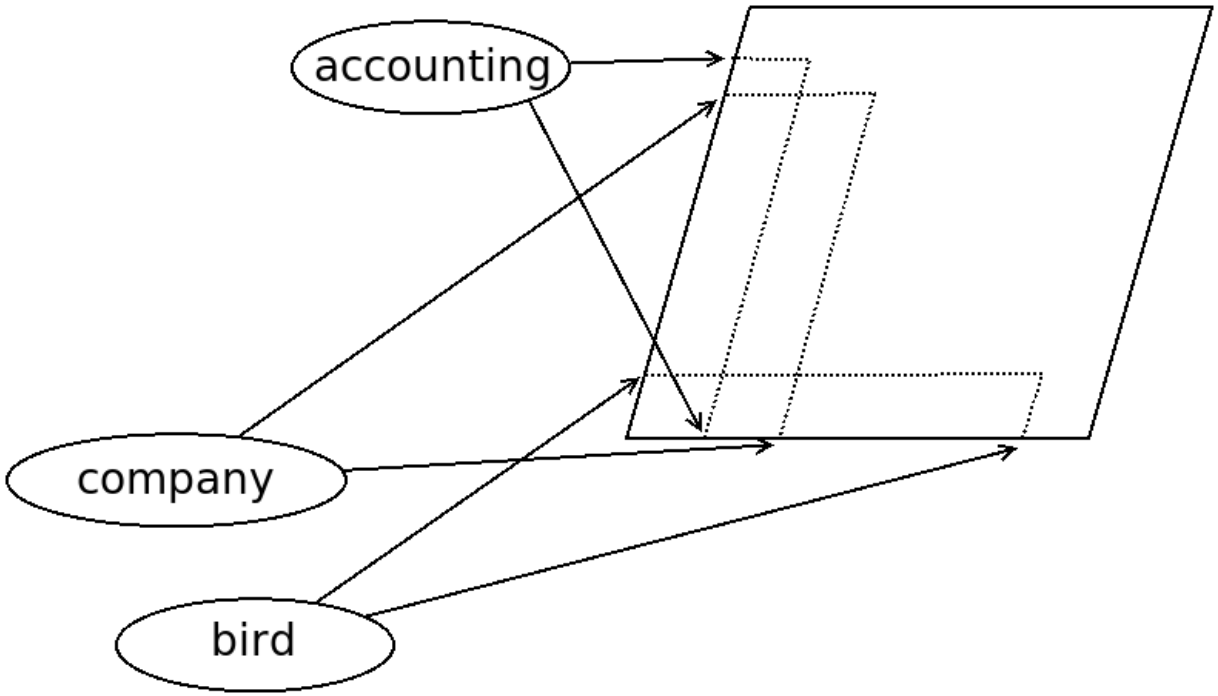
```
from gensim.models import Word2Vec
from gensim.models.word2vec import LineSentence

sentences = LineSentence('wikidata-trigrams.qs')
w2v = Word2Vec(sentences, size=100, window=1, min_count=20)
w2v.save('wikidata-trigrams')
```

Training takes several hours.

Upload pre-trained models to Zenodo: 10.5281/zenodo.823194 and 10.5281/zenodo.827338.

Inspiration: Word embedding



Word embedding (Mikolov et al., 2013; Al-Rfou et al., 2014)

Project words into a low-dimensional subspace.

Estimate the projection based on window sweeping through a corpus and model the relation between a word and its context.

Hopefully semantically related words appear near each other, so that "most similar" words can be based on simple distances, e.g., cosine similarity.

Inspirations and related

graph embedding (Q32081746)

Recently published works on the topic

Search:

Date	Work	Topics
2017-10-11	Wembedder: Wikidata entity embedding web service	Wikidata // Resource Description Framework // graph embedding // web service
2017-07-05	Complex and Holographic Embeddings of Knowledge Graphs: A Comparison	statistical relational learning // graph embedding
2017-03-11	On the Equivalence of Holographic and Complex Embeddings for Link Prediction	graph embedding
2016-01-01	RDF2Vec: RDF Graph Embeddings and Their Applications	graph embedding
2016-01-01	RDF2Vec: RDF Graph Embeddings for Data Mining	Semantic Web // graph kernel // graph embedding // data mining
2015-12-07	Holographic Embeddings of Knowledge Graphs	graph embedding

Edit on query.Wikidata.org

Graph embedding Scholia page

Instead of words, graph embedding projects the nodes of a graph, e.g., a knowledge graph.

Recent work from Ontodia: "The system finds and ranks properties related to a user query using distributional semantics" using fastText trained on Wikipedia (Wohlgenannt et al., 2017).

Problem: Memory

Memory: With 36 million item and 100 dimensional embedding space we have 3.6 giga parameters in the model.

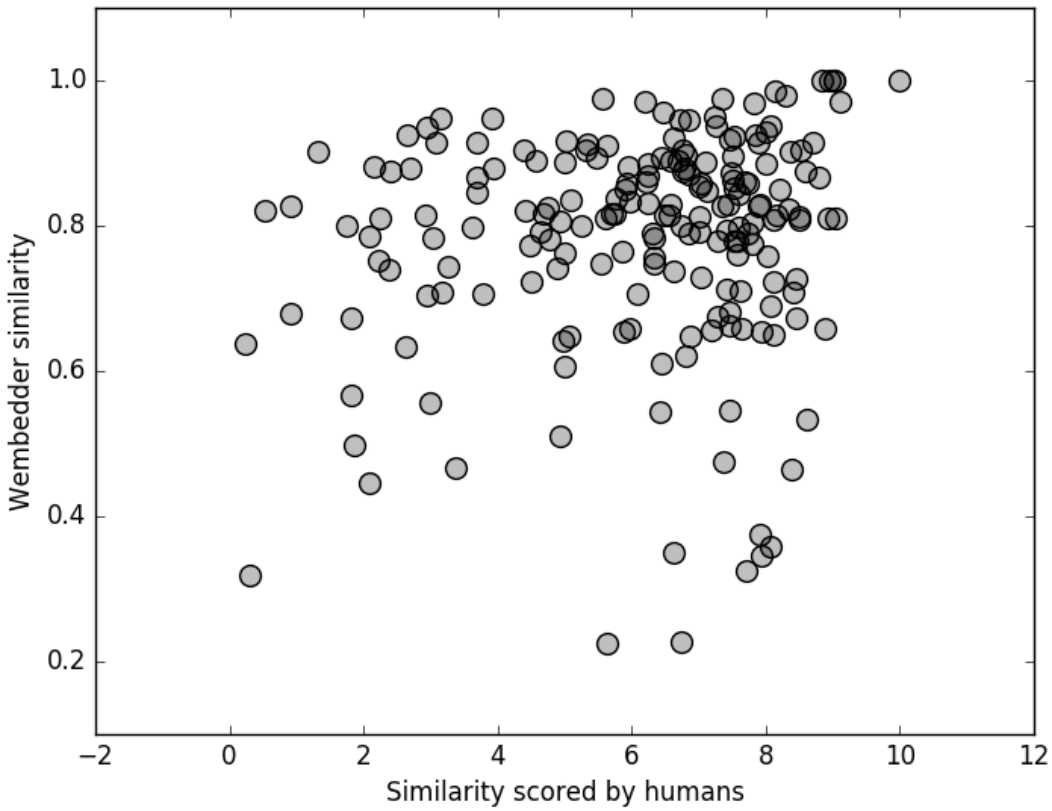
Models are restricted, by "min count" on 20, i.e., the entities must occure 20 or more times to get included in the trained model.

Leaves a vocabulary of only 609'471 entities!

Current stored Gensim models are 2 times approximately 600 megabytes.

Memory is particular a problem with running on Wikimedia Toolforge as "For Kubernetes the default limit is 2G for most runtimes".

Problem: Accuracy



Wordsim evaluation: Similarities scored by humans compared to Wembedder similiarity.

Possible improvements: increase iterations, bigrams for non-item-values, longer graph walks? Wait for Wikidata to become more dense?

Some of the results seems to be guided by the use of P180, e.g., a query shirt may return items such as "decubitus" and "gaze towards the viewer".

Wembedder

GitHub: https://github.com/fnielsen/wembedder

Canonical web site: https://tools.wmflabs.org/wembedder

But runs from http://127.0.0.1:5000/ via `python app.py`

Wembedder: Wikidata entity embedding web service (Q41799598)

Thanks

References

Al-Rfou, R., Perozzi, B., and Skiena, S. (2014). Polyglot: Distributed Word Representations for Multilingual NLP. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space.

Ristoski, P. and Paulheim, H. (2016). RDF2Vec: RDF Graph Embeddings for Data Mining. *The Semantic Web – ISWC 2016*, pages 498–514. DOI: 10.1007/978-3-319-46523-4_30.

Wohlgenannt, G., Klimov, N., Mouromtsev, D., Razdyakonov, D., Pavlov, D., and Emelyanov, Y. (2017). Using Word Embeddings for Search in Linked Data with Ontodia. *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks*.