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?

Count some form of co-occurrences 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: Web service with graph embedding.

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

Multilingual query viaWikidata 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

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 $\rightarrow$ Quick statement-like $\rightarrow$ 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.

```python
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.

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 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 occur 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”.

Finn Årup Nielsen
Problem: Accuracy

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

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”.

Finn Årup Nielsen
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

