Word2sem: automatic generation of semantic relations between entities using DBpedia and word2vec

Mathieu Garchery
Master student - Computer science
University of Passau, Germany
garcherym@gmail.com

ABSTRACT

We present word2sem, a system to automatically generate semantic relations between DBpedia entities using word2vec vector representations. We propose a measure to quantify the specificity/generality of relation types and use it as an adaptive threshold to see if a relation fits a particular type according to their vector representations. To evaluate our model, we try to reconstruct existing relations between entities from DBpedia. We also show that our method could be used to discover new, unknown semantic associations.

Keywords

Natural language processing, relation extraction, word embeddings, Open Linked Data Cloud

1. INTRODUCTION: FROM WORD2VEC TO SEMANTIC RELATIONS

Word2vec, a powerful statistical model for NLP.

In recent years, word2vec [11, 14] has emerged as a powerful tool for natural language processing (NLP), by computing vector representations of words, called word embeddings. Such word vectors are based on the word's context, the intuition behind this being that words appearing in very similar contexts have similar usages and thus similar meanings. Word2vec actually includes two different methods to compute word embeddings: continuous bag of words (CBOW) and skip-gram. In CBOW, a word is predicted given its context, and skip-gram is based on the opposite operation - the context is predicted given the word [11].
Two main aspects of word2vec explain why it has become a largely used model for NLP. First, it works in a totally unsupervised way, meaning that no labeled data is required as ground truth: raw natural text is sufficient. Secondly, complex NLP tasks can be performed very easily through mathematical operations on word vectors obtained with word2vec.

Use cases of word2vec.

For example, searching for synonyms of a given word can be done by simply looking for nearest neighbors in the vector space [13]. Because of the mathematical nature of the model, it is also possible to quantify the distance between two words. Another possible use case of word2vec is machine translation, done by aligning the vector representations of the same words in different languages [12]. This approach works because word representations are defined according to the surrounding contexts, which can be similar in different languages because they describe the same reality. Finally, a further possible usage of word2vec models - and the focus of this research - is analogy computation. It has been shown in [14, 13] that word vector spaces obtained with CBOW or skip-gram tend to contain linearities, corresponding to linguistic regularities like similar relations between entities and analogies, i.e. "A is to B as C is to D" (see example in figure 1).

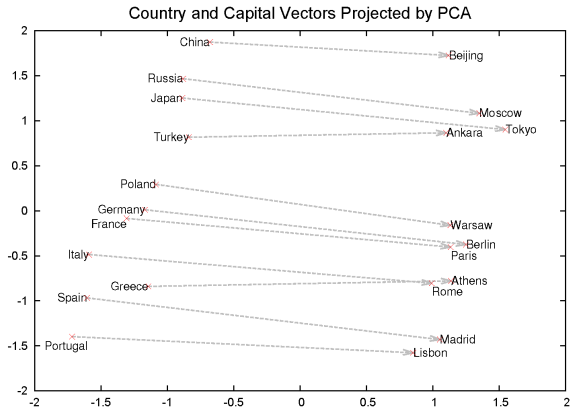

Figure 1: Linear structures in a word vector space: instances of the same semantic relation (here country/capital city association) have similar representations (image source: [14]).

Word2vec and DBpedia for characterization and automatic extraction of semantic relations.

As already mentioned, word2vec is capable of finding analogies between concepts. An example of such analogy search is presented in [14] with the question: "man is to king as woman is to what?". The expected answer ("queen") can be found easily in the word vector space by simple arithmetic operations on the vectors representing the concepts mentioned in the question:

$$v(queen) \approx v(king) + v(woman) - v(man) \qquad (1)$$

This operation consists in applying the vector corresponding to the relation type ("female equivalent of") to the concept "king", giving "queen" as nearest neighbor result. In word2vec, analogies can be computed mathematically but the nature of the underlying relations stays implicit and cannot be derived from mathematical operations. This leads to

the following research questions: How to characterize semantic relations (i.e. linearities in word vector spaces)? How to build a consistent vectorial representation of semantic relations? Can we use such representations and the previous equation to discover new semantic relations between concepts? Our approach - which we call word2sem - is to use DBpedia - a popular knowledge base built on Wikipedia representing facts with RDF triples (base entity, relation, target entity) - as a ground truth for semantic relations between entities. By building a word2vec model of those entities (instead of simple words or concepts), we are able to label relations in our vector space and aim at building a system capable of finding semantic relations in a statistical way. Our approach can be divided in two steps: we first generate vectors for relation types[1], and secondly we try to apply these vectors to entities in the vector space in order to discover valid instances of the given relation type.

Summary.

The rest of this paper is organized as follows: in section 2, we summarize previous work related to our approach and describe the principle of word2sem[2]. Then in section 3, we present three word embedding datasets on which we conducted our experiments. In section 4 and 5, we describe the two main steps of word2sem, which are respectively generating vector representations for relation types, and automatically discovering semantic relations thanks to vector representations of entities and relation types. We also evaluate the performance of our system in these two tasks, taking DBpedia as ground truth. Finally, in section 6, we assess the advantages and shortcomings of word2sem and outline some future improvement possibilities.

2. WORD2SEM, RECONSTRUCTING DBPEDIA SEMANTIC RELATIONS WITH WORD2VEC

Our goal is use both word2vec and DBpedia to automatically retrieve new semantic relations between entities. First, we describe previous attempts at characterizing relations obtainable with word2vec, then we summarize some approaches to explore and discover new relations in knowledge bases (DBpedia in particular), and take an inventory of research projects combining both resources. We also describe SemanticExtractor, a previous system aiming at characterizing word2vec analogies.

2.1 Related work on relation extraction with DBpedia and word2vec

Semantic relations in word2vec.

While word2vec makes it easy to find relations between concepts by looking for regularities in the generated vector space, it does not offer simple characterization methods for the identified relations. In [8], the author tries to characterize word2vec relations with WordNet [5] relation types (synonymy, antonymy, hyper- and hyponymy, mero- and holonomy) and concludes that word2vec relations are essentially hypernyms, synonyms and hyponyms. Additionally, [1] compares lexical approaches (like WordNet) and distributional semantic models (DSM, like word2vec) for similarity and relatedness between words, and shows that DSMs can be used to perform searches for words missing in WordNet. In [4], the authors use condensation and projection of word vectors in the hope of maximizing semantics captured in word vectors. They also try to map word vectors to entity (often composed of several words) vectors.

Exploration of relations between entities in DBpedia.

DBpedia is a popular knowledge base containing facts represented as RDF triples. All these facts form a directed graph in which nodes are entities and edges are relations from one entity to another. Because this graph is huge (DBpedia contains 3 billion RDF triples[3]), navigating in this graph in order to find specific relations between entities has become both very useful and non-trivial. [10] proposes an algorithm to find paths between two given entities in the DBpedia graph, and the authors of [7] suggest to rank semantic relations by relevance, just like search engines do with web pages. Besides exploring existing relations [9], other systems aim at discovering new relations that are missing in the graph, like [3] which uses neural tensor networks to derivate new relations between entities.

Combining knowledge bases and word vectors.

In [2], the authors extend the word2vec model to add information about entity relations from knowledge bases in the training objective, so as to obtain word embeddings with richer semantics. They link entity vectors and RDF triples (subject, relation, object) by the following equation:

$$v(s) + v(r) \approx v(o), \text{ if the RDF triple (s,r,o) is valid} \quad (2)$$

This assumption is also used in [15], where the authors build entity embeddings from a knowledge base in the context of relation extraction tasks. Our approach goes the other way around, by trying to complete a knowledge base with missing relations using word2vec embeddings (see 2.2).

SemanticExtractor: a previous attempt at discovering new word2vec analogies.

SemanticExtractor [6] is a system aiming at discovering new analogies between DBpedia entities with the use of word2vec. It uses an annotated corpus to build a word2vec model of DBpedia entities. Its principle of functioning is the following (to find new "A is to B as C is to D" analogies):

1. choose a base entity A
2. find DBpedia relations with head = A (tail is called B), where B is in the word2vec model
3. find entities similar to A (called C), based on common DBpedia types
4. apply AB vectors on C entities to find potential D candidates
5. filter out low-quality D candidates (based on cosine similarity between AB and CD)

[1]By relation type, we refer to abstract relations like "is the capital of". A relation type can have concrete relations (or relation instances), in this case Berlin/Germany or Paris/France for example.

[2]https://github.com/mgarchery/word2sem

[3]http://wiki.dbpedia.org/about

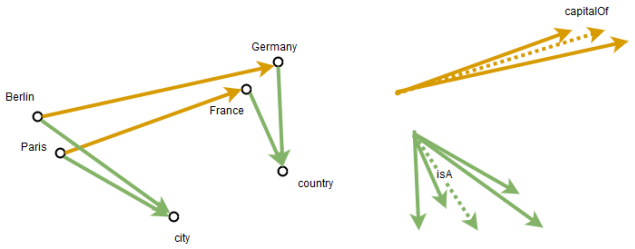

Figure 2: Intuitive representation of word2sem. Here, six relation instances of two relation types are represented. The dotted vectors represent relation types ("isA" and "capitalOf"). As "capitalOf" is more specific than "isA", it is expected to have more consistent vector instances. We use a dispersion measure of these vector instances to compute an adaptive similarity threshold for each relation type.

However, this system has several drawbacks. First, DBpedia only contains relations between entities and not analogies, meaning that the new, discovered analogies cannot be verified automatically and have to be evaluated manually, which is very time consuming and not necessarily objective. Secondly, in practice many incoherent analogies are generated: most of the potential D entity candidates (found by applying the fourth step of the algorithm) are not relevant. This is probably due to the fact that new relations are generated based on one single valid relation (A-B) on one hand, and because the cosine similarity threshold used to filter the results has a fixed value on the other hand. Therefore, some relations generate lots of new analogies (too low similarity threshold) although they are very specific while other relations generate none (too high similarity threshold). Our new approach called word2sem overcomes these problems by:

- generating new relations (thus convertible to RDF triples) instead of analogies, making automatic evaluation with DBpedia possible,
- aggregating relations of the same type to compute their vector representation, therefore reducing noise,
- using a variable similarity threshold depending on the relation type.

2.2 Principle of word2sem and contributions

Like previously explained, word2sem, our new approach to generate semantic relations between DBpedia entities using word2vec linearities, has been greatly influenced by SemanticExtractor. However, as this last system performed badly in practice, some important improvements have been made. Instead of analogies, simple relations (i.e. RDF triples: subject entity, relation type, object entity) are used for simplification and to allow automatic evaluation on DBpedia. Like SemanticExtractor, word2sem uses a word2vec model of entities. This is an important prerequisite to link DBpedia and word2vec representations of same entities altogether. Because building such a model is complex and time-consuming, we choose to use already existing datasets (described in 3). Based on a word2vec model of DBpedia entities, word2sem generates semantic relations in two steps:

1. Relation instances are aggregated by relation type, and a vector representation is computed for each relation type, which is simply the mean vector of all its instances. The specificity of each relation type is evaluated by a dispersion score. This allows to have specific similarity thresholds for all relation types.
2. Semantic relations between entities are generated in a supervised (when taking DBpedia as ground truth) or unsupervised (when looking for missing relations) way. Vector representations of relation types and their similarity thresholds are used, to make sure that new relations are plausible with respect to the observed data (see figure 2).

The first step, relation types characterization, is described in detail in section 4. There, we also discuss what are the most (and least) specific relation types present in DBpedia according to our dispersion score to make sure this measure truly represents the "specificity" of relation types.
The second step, reconstruction of semantic relations, is described in section 5. We evaluate whether the generated relations are valid RDF triples according to DBpedia, and propose a method to discover new relations of given types and/or involving particular entities.

Our main contributions are:

- proposing a dispersion measure to assess the specificity of relations in word2vec,
- using this measure to analyze what are the most/least specific relation types in DBpedia,
- proposing a method to reconstruct existing (and possibly discover new) semantic relations between entities with word2vec and DBpedia.

3. DATASETS

In this section, we describe three datasets, on which our experiments were conducted. These datasets were built in the context of the research described in [16], and all contain word2vec embeddings of DBpedia entities.

DBpedia dataset (without categories).

The first entity embeddings dataset is built on the DBpedia graph, representing all links between entities that are expressed in RDF triples. Traditionally, word2vec is applied on natural text, which is nothing more than a sequence of words. Here, the authors of [16] have to generate a sequence of entities to be able to apply word2vec. They choose to perform a random walk on the DBpedia graph and keep track of the visited nodes (entities) to generate entity sequences. Then, they apply word2vec (skip-gram, with a vector dimensionality of 400) to get vector representations of the visited entities. The underlying intuition is that DBpedia entities have (more or less distant) neighbors in the graph, just like a word has neighbors in a text.

DBpedia dataset (with categories).

The second entity embeddings dataset is built on the DBpedia graph as well, using the same method. However, the graph on which the random walk is performed to generate entity sequences is slightly different. The authors of [16] choose to enrich this graph with information about the categories of the entities, assuming that two unrelated entities

(i.e. that are not linked by any RDF triple) still are neighbors if they belong to the same category. To make this explicit, they add new edges in the DBpedia graph between entities of the same category, thus allowing shorter connections between them. The word2vec model is generated with the same parameters as before (skip-gram, vector dimensionality of 400).

Wikipedia dataset.

The third entity embeddings dataset is built on Wikipedia instead of DBpedia. This is not a problem as DBpedia is initially a "structured version of Wikipedia", and both resources share the same entities. The random walk is this time performed on the Wikipedia hyperlink graph. In this graph, one page is connected to another if there is a hyperlink to the second page in the content of the first page. A random walk algorithm is applied on this graph and a word2vec model is generated with the same parameters as for the DBpedia datasets (skip-gram, vector dimensionality of 400).

4. CHARACTERIZING RELATIONS AND EVALUATING THEIR SPECIFICITY

In this section, we describe the first task of word2sem, which consists in characterizing existing relations with DBpedia relation types, and generating vector representations for the relation types. We also propose a measure to evaluate their "specificity". We present results obtained with our three datasets and verify that our specificity score is valid.

4.1 Description of our method

We here describe our method to generate vector representation of DBpedia relation types, with the prerequisite of having a word2vec model containing DBpedia entities:

1. pick some entities contained in the word2vec model
2. for each entity e, find RDF triples (e,r,t) where e is the head and compute the vector corresponding to the relation:
$$v(r) = v(t) - v(e) \quad (3)$$
3. aggregate relation vectors by relation type, each relation type is represented as the average vector of its relation instances (see figure 2)
4. compute a relation specificity/consistency score for each relation type using the average cosine similarity for all relation pairs of the given relation type

When applying the described method, we hope to generate representative vectors for relation types. The average cosine similarity between relation pairs is expected to reflect the specificity of the relation type. It is also used in the similarity threshold for the semantic relation reconstruction (see 5).
To compute the results described in next section, we picked 50 000 random entities for each dataset. All found relations are kept and aggregated into relation types. Relation types with less than three instances are discarded, because they do not have enough pairs of elements to compute the average cosine similarity. Additionally, we consider the absolute value of the specificity score.

Table 1: 10 most specific relation types (DBpedia dataset without categories)

Relation type	Count	Avg. cos. sim.
property/instrument(s)_	4	0.924
property/teamsInvolved	5	0.768
property/localScenes	13	0.699
property/stations	4	0.676
property/pastmajorleague	27	0.666
property/majorIslands	4	0.660
property/largestwin	4	0.654
property/schoollocations	4	0.639
property/team2Player	8	0.611
property/rd2t3Loc	4	0.601

Table 2: 10 least specific relation types (DBpedia dataset without categories)

Relation type	Count	Avg. cos. sim.
ontology/subsequentWork	1733	0.00003
property/association	21	-0.00006
ontology/previousWork	1683	0.00007
ontology/wikiPageRedirects	153	0.00008
property/northeast	137	0.00009
property/flagbearer	34	0.00009
property/nextAlbum	24	-0.00011
ontology/partner	13	-0.00011
property/mayor	10	-0.00014
property/topScorer	9	0.00016

4.2 Results and discussion

DBpedia dataset (without categories).

The ten most specific and least specific relation types according to our average cosine similarity score are presented in tables 1 and 2 respectively. The most specific relation types actually seem to be quite specific (for example "teamsInvolved", which is probably related to sports, or "schoollocations") in comparison to the least specific relation types which contain much broader elements ("subsequent work", "association", "northeast"). It is sometimes very difficult to understand to what do specific relation types correspond when only their URI is available (like "rd2t3Loc").

DBpedia dataset (with categories).

The ten most specific and least specific relation types according to our average cosine similarity score are presented in tables 3 and 4 respectively. Here, some specific relations are almost impossible to understand according to their labels (like "rt", "ht", "rpos" or "hpos"). However, the ones with explicit labels actually seem to be specific ("nativeClients", "headquarter"). In the least specific relations, we find geographical properties ("nearS", "nearW", "nearE", "southeast", "northwest") and broad associations like "previousWork" or "precededBy". From the most and least specific relation types, it seems that our average cosine similarity score does represent specificity quite well.

Table 3: 10 most specific relation types (DBpedia dataset with categories)

Relation type	Count	Avg. cos. sim.
property/nativeClients	4	0.923
property/rt	5	0.917
property/ht	6	0.897
property/rpos	6	0.778
property/hpos	6	0.778
property/instrumental	4	0.772
property/activities	6	0.736
property/stations	10	0.699
property/headquarter	4	0.682
property/largestwin	4	0.652

Table 4: 10 least specific relation types (DBpedia dataset with categories)

Relation type	Count	Avg. cos. sim.
ontology/previousWork	1205	-0.00003
ontology/spouse	243	-0.00003
ontology/subsequentWork	1238	0.00008
property/nearS	68	-0.00008
property/nearW	59	0.00008
property/precededBy	95	0.00010
property/nearE	61	-0.00012
ontology/secondDriver	8	-0.00015
property/southeast	100	0.00016
property/northwest	111	0.00016

Table 5: 10 most specific relation types (Wikipedia dataset)

Relation type	Count	Avg. cos. sim.
property/zoosubsectio	7	0.944
property/positionOrQuorum	5	0.914
property/victories	4	0.905
property/unrankedSubclassis	7	0.851
property/assemblyconstituencies	7	0.843
property/secondRace	5	0.834
property/thirdRace	7	0.830
property/firstRace	5	0.829
property/thirdTeamRace	4	0.792
property/industries	10	0.788

Table 6: 10 least specific relation types (Wikipedia dataset)

Relation type	Count	Avg. cos. sim.
ontology/previousWork	944	0.00006
ontology/subsequentWork	946	0.00014
property/precededBy	77	0.00027
property/currentTeam	11	0.00030
property/spouse	210	0.00039
property/southwest	103	0.00041
property/foundedBy	5	0.00044
property/nearE	38	0.00048
property/champion	15	0.00048
property/nearW	44	0.00065

Wikipedia dataset.

In the ten most specific relation types obtained with the Wikipedia dataset (table 5), we find associations related to (sport?) races, like "secondRace", "firstRace", "thirdRace", "thirdTeamRace", and other very specific relationships ("positionOrQuorum", "zoosubsectio", "assemblyconstituencies"). In the relation types ranked as least specific according to the average cosine similarity (table 6), we again find relative geographical locations ("southwest", "nearE", "nearW") and the very broad "subsequentWork" and "previousWork", like in the other two datasets.

Overall considerations about specificity and occurrence count.

In all three datasets, it is noticeable that specific relations are much more rare (have lower occurrence counts) than general relations. This is logical, because if a relation type has many instances, the dispersion between all pairs of instances (which is quantified by the average cosine similarity between pairs) is more likely to be high. In this work, we implicitly use the term "specificity" of a relation to assess how constraining, how restrictive this relation is. But the previous results suggest that specificity may also be defined as the inverse probability of a relation, as very general relation types are less constraining and therefore more likely to be valid between any pair of entities. However, one could also argue that some relation types have a very low specificity score even with few relation instances (for example "topScorer" in table 2, "secondDriver" in table 4 and "foundedBy" in table 6, which all have less than 10 instances). Here, the average cosine similarity score does not reflect specificity so well, as only "foundedBy" seems like a general relation type.

5. RECONSTRUCTION OF SEMANTIC RELATIONS

We previously described a method to generate vector representations of DBpedia relation types using word2vec, and we showed how to compute a "specificity" score for each relation type. The next step of our system is to reconstruct semantic relations between entities using those relation type vectors and specificity scores. This reconstruction of semantic relationships can be done in a supervised way (i.e. regenerating existing associations) or unsupervised way (i.e. discovering new associations). We first describe our algorithm to generate relations between entities, then we present and discuss our results for supervised reconstruction of semantic relations on three datasets (see section 3), and finally we discuss unsupervised generation of new associations.

5.1 Description of our method (supervised reconstruction of DBpedia relations)

We here present our method to reconstruct DBpedia relations between entities in a mathematical way using word2vec. The method is the following:

1. define E, subset of base entities (will be head entities of the relations)
2. define RT, subset of relation types to look for
3. for each top-n relation candidate (e1,r,e2) with e1 in E and r in RT (i.e. $v(e2) \approx v(e1) + v(r)$, taking the n nearest neighbors for e2):

if $cosSim(v(e2) - v(e1)) > avgCosSim(v(R)) - stdCosSim(v(R))$, where R is the relation type of r, add (e1,r,e2) to the results set

4. compare generated relations and existing DBpedia relations (ground truth)

In the third step, $cosSim(v(e1), v(e2))$ is the cosine similarity between the vectors representing e1 and e2, $avgCosSim(v(R))$ is the average cosine similarity between pairs of relations instance of type R (it corresponds to the specificity score described in section 4), and $stdCosSim(v(R))$ is the standard deviation of the same distribution. This means that the difference between the average cosine similarity and the standard deviation of the same distribution constitutes a variable selectivity threshold depending on the specificity of the relation type. This formula is of course a bit arbitrary and one could also choose to use $avgCosSim(v(R)) - 2 * stdCosSim(v(R))$ or $avgCosSim(v(R)) - 3 * stdCosSim(v(R))$.
To select relation candidates, a parameter corresponding to the number of nearest neighbors in the word2vec space is used: our method tried to apply the relation vector r to the base entity e1, to see if any relevant entity e2 can be found. Instead of just taking the nearest neighbor entity, we explore n closest entities. Intuitively, this "top-n" parameter can serve as balance between precision and recall. A lower value of n means exploring less, but more precise relations, whereas as higher value can retrieve more results with possible precision loss.
Lastly, one important question is how to select the base entity and relation type sets. As for the set of base entities, we choose to randomly pick some concepts. For the relation type set, several choices are possible:

- use existing relation types (from a DBpedia Spotlight query) for each entity, so as to guide our system on the relations to discover, like in our supervised evaluation (see next section). This means that a new relation (that does not exist in DBpedia) can only be found if another relation of the same type already exists. This is not a problem here since we are trying to regenerate existing relations to evaluate our system on a ground truth,

- use most specific relation types (according to the "specificity" score defined in section 4). This allows to have high confidence results in the case of unsupervised discovery of new relations (i.e. when no ground truth is available),

- combine both into a hybrid strategy, that means using the most specific relations types of the existing relations for each entity. This approach is not explored in this work but could be an interesting extension for future work.

In any case, one could also design the base entity and/or relation type sets manually in order to focus on particular entities and/or relation types.

5.2 Results and discussion

In this section, we discuss results obtained with the previously described method when aiming at reconstructing existing semantic relations between entities. We take a subset of 1000 randomly selected entities as base entity set, and the types of existing relations for each entity as relation type set. For each dataset of the three datasets, we use several values of the top-n parameter (2, 5 and 10). For each configuration, we evaluate our system based on precision, recall and F1 score, taking DBpedia relations as ground truth. Thus, true positives are valid DBpedia relations found by word2sem, false positives are invalid relations found by word2sem and false negatives are valid relations not found by word2sem.

Table 7: Reconstruction of existing DBpedia relations: precision, recall and F1 scores

	Top-n = 2		
	pre.	rec.	F1
DBpedia (without categories)	15.4%	7.5%	10.1%
DBpedia (with categories)	18.8%	9.2%	12.3%
Wikipedia	13.1%	5.9%	8.1%
	Top-n= 5		
	pre.	rec.	F1
DBpedia (without categories)	11.8%	22.7%	15.5%
DBpedia (with categories)	**13.2%**	**26.9%**	**17.7%**
Wikipedia	9.6%	17.2%	12.3%
	Top-n = 10		
	pre.	rec.	F1
DBpedia (without categories)	7.1%	30.9%	11.6%
DBpedia (with categories)	8.0%	35.1%	13.1%
Wikipedia	6.7%	26.7%	10.7%

The performance scores of all tested configurations are reported in table 7. It is quite easy to see that the top-n parameter has a great influence on the selectivity of our system: a lower top-n value gives a low recall and high precision, while a higher top-n value gives high recall and low precision. Therefore, F1 score is a well-suited measure to find a balance between these two, and determine the optimal value of the top-n parameter. In our experiments, we only tested three different values (2, 5 and 10) for top-n, but this is sufficient since the optimal value of this parameter seems to be around 5.
More generally, the scores are quite low (the best performing configuration reaches 17.7% F1 score), and the DBpedia dataset with categories (see 3) performs better than the two others for any tested top-n parameter value. The low scores reflect the difficulty of the task (reconstructing human-crafted semantic associations from plain mathematical models).

5.3 Discovering new relations?

In the previous section, we evaluated our system to assess its capability to regenerate existing relations between entities according to a ground truth. We now try to use word2sem to discover unknown/inexistent/missing relationships in DBpedia.
To do so, we proceed in an unsupervised way by trying to generate relations of the most specific relation types (determined with the same method as in 4). By focusing on the most specific relation types, we expect to have few results

(low recall) with a high confidence (high precision). Unfortunately, a reliable evaluation is impossible because too few results are retrieved with this method (there are almost no true positives). This can be explained by the fact that specific relations are consistent but very rare, and general relations are abundant but not consistent. Therefore, very few relations are retrieved, making any evaluation impossible.

6. CONCLUSIONS AND FUTURE WORK

Word2sem: an automatic detector of semantic relations for DBpedia.

In this paper, we present word2sem, a new model for extracting semantic relations between DBpedia entities in a fully automatic way using word2vec vector spaces. We generate vector representations of entities and relations with word2vec. Additionally, we introduce a measure of the "specificity" (as opposed to "generality") of relation types (i.e. abstract relations) and verify its relevance. We then use this measure as an adaptive threshold value for the cosine similarity between entity vectors, so as to determine if a relation between two entities is likely to be of a given relation type. By doing so, we can label relations from one entity to another with particular types, based on the vector representations of the relation and both entities. We show that our system reaches 17.7% F1 score at reconstructing DBpedia relations when focusing on particular relation types.

Future work.

If reconstruction of existing relations is feasible, discovery of new relations is much more difficult, especially if the types of the relations to look for are not specified. To improve our model, we suggest exploring following extension possibilities:

- find a "smart" choice of entity and relation subsets to explore. One could focus on particular entities and/or relation types, in order to reduce the search space and discover new domain-specific relations between entities (targeted search). Another possibility would be to use selectional restrictions (i.e. constraints) linking the entities and the relation types,
- explore other measures to quantify the specificity/generality of a relation type based on its relation instances,
- compare performance when changing the similarity threshold definition slightly (i.e. use $avgCosSim - 2 * stdCosSim$ or $avgCosSim - 3 * stdCosSim$ or other dispersion measures).

Although the current performance of word2sem is quite low, it seems to be a promising approach for a complex NLP task: generating semantic relations between entities automatically. As there still is room for improvement, we do believe better results can be achieved by extending word2sem.

7. REFERENCES

[1] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics, 2009.

[2] A. Celikyilmaz, D. Hakkani-Tur, P. Pasupat, and R. Sarikaya. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. *genre*, 2010.

[3] D. Chen, R. Socher, C. D. Manning, and A. Y. Ng. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*, 2013.

[4] H. Chen and J. Tan. Entity extraction using condensed words-to-entity vector transformation. 2015.

[5] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[6] M. Garchery and L. Laporte. Semantic extraction from natural language using deep learning and linked open data. 2016.

[7] C. Halaschek, B. Aleman-Meza, I. B. Arpinar, and A. P. Sheth. Discovering and ranking semantic associations over a large rdf metabase. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1317–1320. VLDB Endowment, 2004.

[8] A. Handler. *An empirical study of semantic similarity in WordNet and Word2Vec*. PhD thesis, Columbia University, 2014.

[9] J. Lehmann, J. Schüppel, and S. Auer. Discovering unknown connections-the dbpedia relationship finder. *CSSW*, 113:99–110, 2007.

[10] O. Liu. Relation discovery on the dbpedia semantic web. *Framework*, 2009.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[12] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[14] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, volume 13, pages 746–751, 2013.

[15] J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973*, 2013.

[16] S. Zwicklbauer, C. Seifert, and M. Granitzer. *DoSeR - A Knowledge-Base-Agnostic Framework for Entity Disambiguation Using Semantic Embeddings*, pages 182–198. Springer International Publishing, Cham, 2016.