International Journal of
Intelligent Systems and Applications in Engineering

ISSN:2147-6799 www.ijisae.org Original Research Paper

# Using Word Embeddings for Ontology Enrichment

**İzzet Pembeci[1]***

*Accepted 11th July 2016* *DOI: 10.18201/ijisae.58806*

***Abstract:*** Word embeddings, distributed word representations in a reduced linear space, show a lot of promise for accomplishing Natural Language Processing (NLP) tasks in an unsupervised manner. In this study, we investigate if the success of word2vec, a Neural Networks based word embeddings algorithm, can be replicated in an agglutinative language like Turkish. Turkish is more challenging than languages like English for complex NLP tasks because of her rich morphology. We picked ontology enrichment, again a relatively harder NLP task, as our test application. Firstly, we show how ontological relations can be extracted automatically from Turkish Wikipedia to construct a gold standard. Then by running experiments we show that the word vector representations produced by word2vec are useful to detect ontological relations encoded in Wikipedia. We propose a simple but yet effective weakly supervised ontology enrichment algorithm where for a given word a few know ontologically related concepts coupled with similarity scores computed via word2vec models can result in discovery of other related concepts. We argue how our algorithm can be improved and augmented to make it a viable component of an ontology learning and population framework.

***Keywords:*** *Neural Language Models, Word Embeddings, Ontology Enrichment, Ontology Population.*

## 1. Introduction

Capability of collecting huge amounts of textual data from the Web resulted in new Information Extraction and Machine Learning algorithms for primary and secondary Natural Language Processing (NLP) tasks. The attractiveness of these new algorithms is that, in tandem with the nature of the data, they are mostly unsupervised or weakly supervised, thus eliminating the need of creating large labelled datasets for languages where NLP studies are not as mature as languages like English.

In their NLP from scratch approach, Collobert et al. showed that multilayer neural networks (NN) can be used for transforming words to feature vectors which are called word embeddings or word representations. Then, just with the help of these word vectors, they demonstrate how to solve standard NLP tasks like Part-of-Speech tagging, chunking (shallow parsing), Named Entity Recognition (NER), and Semantic Role Labelling in a quite effective manner [1]. Later, Mikolov et al. introduced new neural network based models which can be trained efficiently with billions of words and for vectors with much higher dimensions [2]. Distributed vector representations learned by their so called word2vec models proved to capture precise syntactic and semantic word relationships [3] in an impressive way.

The main goal of this study is to investigate the usefulness of word2vec models for ontology enrichment under a morphologically rich language like Turkish. Ontology population can be described as automatic construction of an ontological knowledge base for a specific domain in an unsupervised way, rather than manually building one. Again, Web and other domain-dependent digital corpora (i.e. hospital patient records) can be used as data sources for such a task [4]. Ontology learning or enrichment methodologies on the other hand are used for extending an existing ontology with additional instances and relations. Since for Turkish we do not have any benchmark, challenge or competition datasets like in English to evaluate our word embeddings approach, we used the Turkish Wikipedia (Vikipedi) to construct a golden standard. Using Wikipedia's rich semi-structured data in this manner coupled with word2vec's generality has the additional benefit of contributing to ontology population and learning literature where domain-independent, open, semi-automatic, and unsupervised or weakly supervised methods are favoured over other approaches [5], [6].

## 2. Word Embeddings Using word2vec

In statistical language modelling, word embeddings (or word representations) are used to group similar words together by representing each word as a k-dimensional vector in $\mathbb{R}^k$. A good embedding should result in word vectors such that the closer the vectors are (i.e. according to their cos distance) the more similar their corresponding words should be. Such word embeddings can be produced in two different but very similar ways using word2vec NN models. In Continuous Bag-of-Words (CBOW) architecture, the context, i.e. words surrounding the target word in a sentence, predicts the current word and in Skip-gram model the current word predicts the context words (Figure 1).

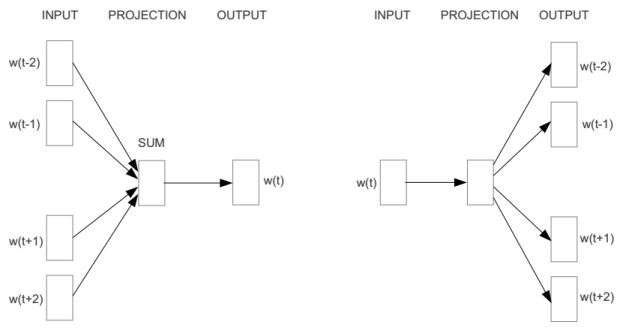



**Figure 1.** CBOW (left) and Skip-gram (right) models. w(t) is the current word, w(t+i) and w(t−i) are the context words.

[1] *Muğla Sıtkı Koçman University. Department of Computer Engineering.*
* *Corresponding Author: pembeci@gmail.com*

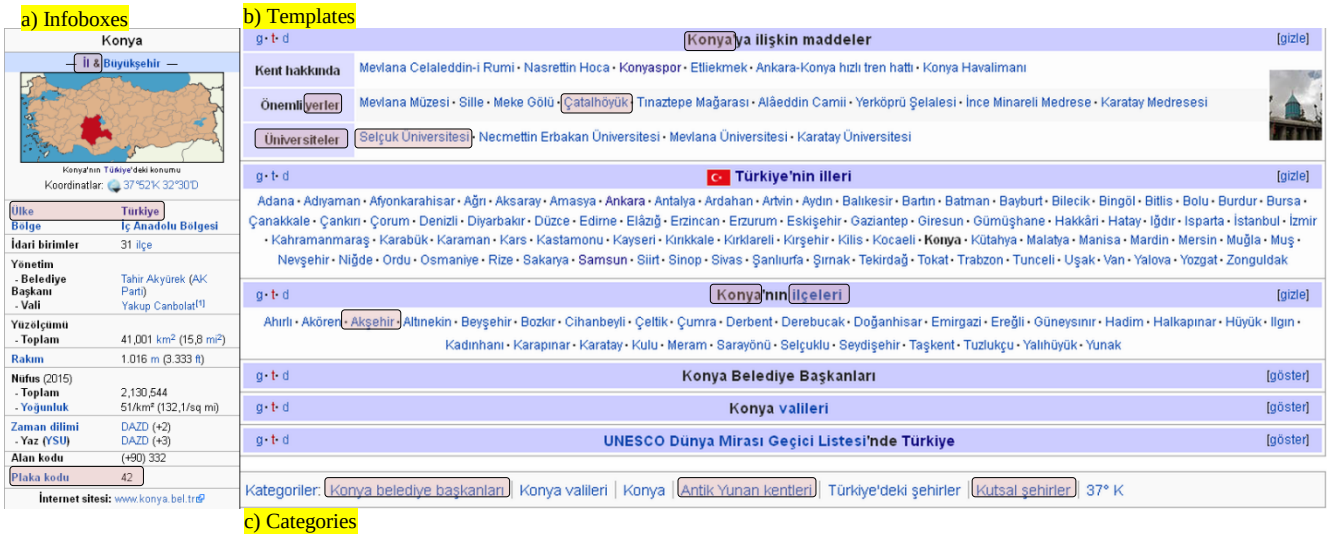



**Figure 2.** Components of Vikipedi Konya article: a) A city infobox b)Templates c)Categories. The parts marked show the places where ontological relations of Table 1 can be extracted from.

The network should be trained from a big corpus where each word in each sentence and its context is regarded as a training instance. Once training is completed the learned weights from the input layer to hidden layer (for Skip-gram) or from the hidden layer to output layer (for CBOW) are regarded as the corresponding word vectors. The theoretical training objective of these models is maximizing a softmax function over all the words in the vocabulary in each training step. Since this objective is impractical, Mikolov et al. proposed efficient approximations of hierarchical softmax and negative sampling [3]. They also incorporated a heuristics for subsampling of frequent words to reduce training time further. We refer the interested reader to [7] for details of how these NNs can be setup and trained.

word2vec impressed NLP researchers not only with its state-of-the-art performance at detecting similar words like Italy or France but semantic relations like capital-of-country or maleness-femaleness seem to be captured by the vector space it produces as well. For instance, the closest vector to the result of vector calculation *vec("Madrid")* − *vec("Spain")* + *vec("Turkey")* is found to be *vec("Ankara")* or *vec("king")* − *vec("man")* + *vec("woman")* is closest to *vec("quenn")*. Similarly *vec("Russia")* + *vec("river")* is close to *vec("Volga River")* or *vec("Germany")* + *vec("capital")* is close to *vec("Berlin")*. Using vector arithmetics, non-trivial syntactic similarities can be also detected: mouse-mice, ethical-unethical, lucky-luckiest, or swimming-swam.

The success of word2vec on analogical reasoning tasks mentioned above renewed research interest on word representations. GloVe algorithm [8] combines global matrix factorization methods used in latent semantic analysis with word2vec's local context window method to produce vectors and WordRank [9] uses a robust ranking model to achieve the same. [10] Describes how word2vec can be extended into the paragraph and document level from sentence level and Item2Vec [11] introduces an item-based collaborative filtering algorithm based on word2vec. By replacing words with vertices of a random walk in a graph, word2vec inspired algorithms can be even used to learn latent representations in a graph and then solve multi-label network classification tasks for social networks like Flicker and YouTube [12]. [13] Advocates and introduces a methodology for learning word representations in the space of Gaussian distributions instead of vectors. To explain and understand the reasons of word2vec and related algorithms' good performance [14] provides some empirical and theoretical insights and [15] shows that word2vec implicitly factorizes a word-context matrix, whose cells are the pointwise mutual information of the respective word and context pairs.

In this paper, we are looking at how much word2vec's, and thus word embeddings', success in NLP tasks can be reproduced in an agglutinative language like Turkish in which from the same root many words can be formed via very productive inflectional and derivational morphotactic. To explore this in the context of a non-trivial and complex NLP task, we extracted ontological concepts and relations from Vikipedi and compared word2vec word similarity measures on these relations. In addition to testing word2vec performance in a more challenging language, we conjecture that the success of such comparisons would mean using word2vec as a useful component of a domain-independent ontology population and enrichment framework is a viable approach.

## 3. Wikipedia as an Ontology Source

Researchers used crowdsourced content of Wikipedia as a resource for ontology learning and population successfully in many different ways [16][17][18][19]. Knowledge bases (or knowledge graphs), which are in fact large-scale, multilingual, spatially and temporally enhanced ontologies, like DBpedia [20] or YAGO2 [21], were also built by the help of Wikipedia.

What makes Wikipedia unique and very useful for these purposes is not just that the articles in Wikipedia are high volume, high quality, and comprehensive but also the community put a lot of effort to enrichen the articles with components like infoboxes or categories to increase site's usability and navigation. In addition to the internal links between articles, these components can be exploited for extracting ontological instances and relations by using lexico-syntactic patterns [22][23], or graph theoretical means [24][25]. Consider the Vikipedi article for city of Konya[1]. It has an

[1] https://tr.wikipedia.org/wiki/Konya

infobox (Figure 2a) at the top of the article, templates (Figure 2b) and categories (Figure 2c) at the end. The wiki mark-up (content) of these components and also their existence can be used to infer many ontological relations of Konya like the ones demonstrated in Table 1.

| Relation | Extracted from |
|---|---|
| Konya *is_a* City | existence of infobox |
| Konya *located_in* Turkey | infobox content |
| Konya *license_plate_number* 42 | infobox content |
| Selçuk Üniversitesi *is_a* University | template header |
| Çatalhöyük *located_in* Konya | template header & content |
| Akşehir *is_a* district | template title & content |
| Akşehir *district_of* Konya | template title & content |
| Konya *is_a* Ancient Greek City | category |
| Konya *is_a* Holy City | category |
| Ahmet Hilmi Nalçacı *mayor_of* Konya | category page content[2] |

**Table 1.** Ontological relations that can be extracted from the components of Vikipedi Konya article. See Figure 2 to get an idea about how these relations can be extracted from article's page.

## 4. Our methodology

To create our corpus for training the word2vec model, we scraped 265 thousand columns from 28 different Turkish newspapers and 1734 columnists. Some basic preprocessing like removing symbols, html tags, numbers etc. left us with 107 million words (tokens) in 5.8 million sentences and a vocabulary size of 2 million word types. If we drop the less frequent words whose count is less than 20 then we are left with 323 thousand unique word types (%16 of original) and 100 million words (%93 of original). We intentionally did not perform any language specific preprocessing like stop-word removal, stemming or POS tagging before training to see word2vec's performance in a raw setup.

### 4.1. Training the word2vec model

As in [3], before we started training we constructed phrases (multiword expressions) from our corpus since most of the concepts we will be dealing with during our Vikipedi tests will consist of such collocations. We used a simple approach based on unigram and bigram counts. Two subsequent words $w_i$ and $w_j$ are assumed to form a phrase if their score as given below is greater than a threshold parameter:

$$score(w_i, w_j) = \frac{(count(w_i w_j) - \delta) \times N}{count(w_i) \times count(w_j)}$$

Here, $\delta$ is used to eliminate infrequent collocations (we set it to 10) and $N$ is the total vocabulary size. We set the scoring threshold as 20. We apply phrase construction in two passes to be able to capture trigrams and four grams as well. For instance, in the first pass new york, barack_obama, usa_president, or ahmet_hamdi will be constructed and since these are considered as word units in the second pass, now we can detect phrases like new_york_times, new york_mets, ahmet_hamdi_tanpınar, or usa president barack obama.

Although optimizing the word2vec parameters was outside the scope of this paper, we did some experiments similar to the ones explained in the next section to chose the most fundamental parameters in a sensible manner and consistent with our dataset. We observed that Skip-gram performs better than CBOW as reported in other papers which involve NLP tasks more semantic oriented like ours. We did not see much difference with negative sampling or hierarchical softmax so we chose the former because of less training time. We set context window size to 10 (in each direction) to cover almost all of a sentence since in Turkish verbs are at the end, negative sample numbers to 5, and subsampling threshold frequency as 0.001. We trained the model with shuffled documents in each iteration not to introduce any bias for some words when the learning rate is high at the beginning of the training. We trained for 100-dimension word vectors (higher values like 500 actually produced worse initial similarity results and training took much longer).

| NODES | |
|---|---|
| **Label** | **Purpose** |
| W | Word node. Title of an Article page. |
| $W_R$ | Word node. Title of a redirection page. |
| $W_D$ | Word node. Title of a disambiguation page. |
| $W_S$ | Word node. Surface form of a link. |
| C | Category node. Title of the page. |
| T | Template node. Name of the template. |

| EDGES | | | |
|---|---|---|---|
| **Label** | **weight** | **Source (from)** | **Target (to)** |
| *link* | 0/1 | W | { W, $W_R$, $W_D$ } |
| *redirect* | 0/1 | $W_R$ | { W, $W_R$, $W_D$ } |
| *disambiguate* | 0/1 | $W_D$ | { W, $W_R$, $W_D$ } |
| *surface* | 0/1 | $W_S$ | { W, $W_R$, $W_D$ } |
| *cat_word* | 0/1 | C | { W, $W_R$, $W_D$ } |
| *word_cat* | 0 | W | C |
| *rel_cat* | 0 | C | C |
| *cat_templ* | 0 | C | T |
| *word_templ* | 0 | W | T |
| *templ_word* | 1 | T | W |

**Table 2.** Types of nodes and edges of $\boldsymbol{G}$ which encodes the relations found between different types of pages in Vikipedi

### 4.2. Obtaining ontology information from Vikipedi

In order to create a gold standard of concepts related to each other based on Turkish Wikipedia, we used the dump file[3] produced on 2016-05-01 consisting of 1.4 million pages. After filtering the meta pages, specific namespaced pages (i.e. community portals) media/file descriptions etc., we are left with 765 thousand article and category pages. Next, going over these pages we created a directed graph $\boldsymbol{G}$ where nodes refer to articles, categories, or templates and edges refer to the relations between them as encoded in Vikipedi. As a result, $\boldsymbol{G}$ consists of 6 different types of nodes and 8 type of edges as described in Table 2. Only edges with target nodes of type W has weight 1, others have weight 0. Weights will be useful for our traversing algorithm.

The main node type in $\boldsymbol{G}$ is actually W which corresponds to the real article pages. The other ones are required to traverse $\boldsymbol{G}$ to find ontologically related words to a given word as will be explained shortly. Redirection pages ($W_R$) help Wikipedia users to search for

[2] https://tr.wikipedia.org/wiki/Kategori:Konya_belediye_başkanları

[3] https://dumps.wikimedia.org/trwiki/20160501/

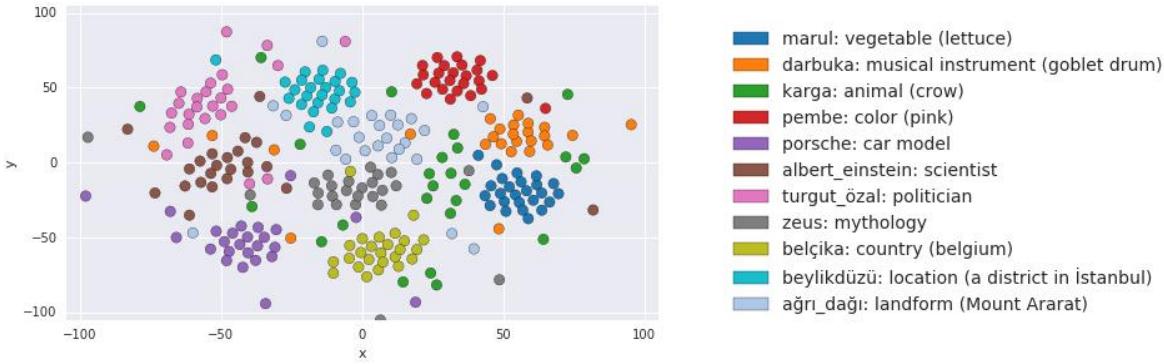



**Figure 3.** The distribution of vectors of randomly chosen words similar (score ≥ 0.60) to seed words (listed at right) visualized in 2D, preserving their closeness in the original vector space.

an alternative name of a concept and to be redirected automatically to the page whose title considered to be more authoritative by the editors. For instance, the page "ABD" ("USA") will be redirected to the page "Amerika Birleşik Devletleri" (United States of America). Disambiguation pages ($W_D$) have a similar purpose but this time the entry point cannot be mapped to another article unequivocally (i.e. editors cannot be sure of the intent of the user) so these pages list all the possibilities as links to articles. As an example, "Menderes" disambiguation page contains 14 alternatives from divergent categories like geography (a river, a national park), location (a district, a village, a neighbourhood), people, universities or airports. $W_S$ nodes are extracted from link texts. Sometimes in an article one cannot use the real title as a link in a sentence (i.e. in "Dolphins are smart marine mammals.", plural "marine mammals" is a link to the singular title "Marine mammal") or it is more convenient, readable to refer to the article in another way (i.e. "President Obama" is a link to the article "Barack Obama"). We need to know these connections and have corresponding edges since in our corpus, we will very possibly encounter such alternative usages. Since Wikipedia is not perfect and evolves in time, an old link, redirection page or disambiguation item can refer the user to pages which are not actual article pages but further redirection pages etc. that need to be followed through to get to the desired article.

We extract the *word_cat* relations from the category links at the bottom of the articles (Figure 2c). Category pages (C nodes) are designed to help the user to find and discover related articles. These pages can both contain links to normal articles (*cat_word* edges) and to other related categories (*rel_cat* edges). We manually filtered some of the categories which were too broad, and/or not encoding a very meaningful or useful ontological relation (i.e. "People born/died in year X" or "Turkish words borrowed from French").

Finally, we extracted the templates (T nodes) from the top and bottom of the article markup. The topmost templates in an article are generally used for category specific infoboxes (Figure 1a) and the bottom ones as seen in Figure 2b and Table 1 also carry ontological value. We found out that templates inside the articles are generally more editorial (i.e. "this needs a citation", "newspaper source") and would not help much for our purposes. We did not include the links found in template pages since these may be very irrelevant (i.e. "Zaman Dilimi" (timezone) link in Konya's city infobox) but templates are still useful since we can deduce that their neighbours (i.e. articles sharing the same template) are instances of the same ontological category. *templ_word* edges are the counterparts of each *word_templ* edge found (i.e. bidirectional links).

The idea behind constructing a graph like ***G*** is starting from a word node and traversing the graph from that node for the closest neighbours will give us ontologically related words to the original word. Most of the related words found will be in the same ontological category as the original word (i.e. they share the same hypernym Y's as in "X is a kind of Y") but we will encounter other relations as well. After achieving such a list then we can run experiments and measure how much words in this list are considered to be similar (i.e. words' vectors are closer) to the source node word by the word2vec algorithm. Higher similarity scores will indicate that word2vec is a feasible option as the basis of an ontology enrichment algorithm.

The output of our traversal algorithm customized for ***G*** will be denoted in this paper as $neighbors_{w,d}$ which can be interpreted as all neighbours of starting node word w whose distance to w is at most d steps. $neighbors_{w,d}$ has the following properties:

- A word v is in $neighbors_{w,d}$ if and only if, type of v is W and there is a path from w to v whose weight total is at most d. Thus, we are not interested in category or template nodes in our final results since these do not correspond to ontological concepts.
- We follow *rel_cat* edges only if the ratio of the degree of the C nodes in question are below a threshold.
- We follow *link* edges only if the link is bidirectional. That is there should be also a *link* edge from v to w.

The first property means that we are using nodes of type $W_R$, $W_D$, C, T as stepping stones only. We are not counting them while checking the d-steps criteria or returning them as neighbours. $W_S$ has only outgoing edges so they will not be part of any path and will be only helpful during the experiments in case we need to know alternative usages of a word. $W_R$, $W_D$ nodes will be also helpful for this. The last two conditions are added not to dive into the regions of ***G*** during traversal which may be only marginally related to the original word.

## 5. Experiments and Results

In addition to using $neighbors_{w,d}$ to prepare the test data for our experiments, we will refer to *sim(w1,w2)* as the similarity score computed by our word2vec model which is the cosine distance between the normalized vectors corresponding to w1 and w2. Consequently, a similarity score closer to 1.0 will mean more similar words. We will also use *most_similar(w, topn)* notation for the *topn* words whose similarity score to w is the highest. *model_word(w)* will map a word in ***G*** to its counterpart in word2vec

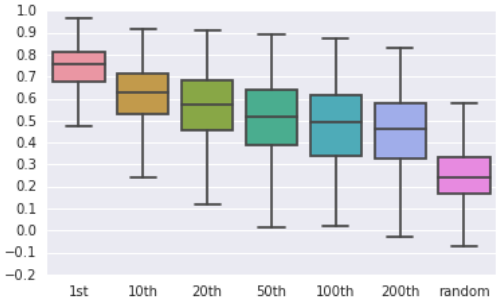



**Figure 4.** Distribution of similarity scores for the ontologically related words of every word in G according to their ranks.

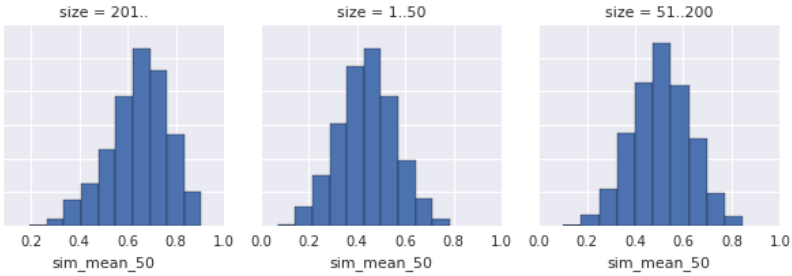



**Figure 5.** Histograms of mean similarity scores of the first 50 most similar words grouped by how many words (size of test set) are found to be related to the seed word in $\boldsymbol{G}$. For the words with more ontological relations (i.e. higher size), the scores tend to be higher.

vocabulary. Most of the time this will be the same word but sometimes we need to follow redirection and disambiguation links in reverse. For instance, the title/node "Bitlis_(il)" ("Bitlis_(province)") will not be found in word2vec vocabulary but it has an incoming edge from the "Bitlis" disambiguation node which is of course how it is mentioned in our corpus.

**Experiment 1:** Our first experiment was about finding out if word2vec can achieve good separation between words which belong to different ontological categories. If it fails to do so there was not much point to pursue this research further. To test this we selected 11 seed words belonging to diverse categories. Then for each seed word w, we get $neighbors_{w,2}$ and selected 25 words randomly from this set whose similarity score with the seed was higher than 0.60. We used t-SNE [26] to visualize the closeness of selected words' vectors in 2D space. As it can be seen in Figure 3, the results were pretty encouraging since words generated from the same seed show strong clustering effects. Additionally, the clusters of people (a scientist and a politician), the clusters of a vegetable and an animal, and clusters of a district and a mountain in Turkey were close to each other since the contexts of these pair of words can be expected to be more similar than the others. Clusters of non-Turkish proper name seeds "Porsche", "Zeus", "Belgium" and "Albert Einstein" also span the same bottom left region. Admittedly, not all categories produce such sharp clusters. Movie names for instance tend to be underrepresented in our corpus. word2vec as being an NN algorithm needs a lot of examples to converge a word's vector to a point in space, so vectors of less frequent words are more distributed (i.e. more randomly placed) in vector space. Additionally, some movie names (i.e. Independence) are used mostly in non-movie contexts which eventually determine their vector representation. The latter phenomenon can be also observed with our seed word "karga" ("crow") in Figure 3 since animal and bird names can be used in literal ways as in idioms. This produces many different and unrelated contexts for such words and thus making the clustering effect weaker.

**Experiment 2:** Next, we wanted to look at how much the good results of previous experiment can be generalized for the other concepts in $\boldsymbol{G}$. To do that, first for each node w in $\boldsymbol{G}$ with label W we get its test set ($neighbors_{w,1}$ but if it contains less than 20 words then $neighbors_{w,2}$). Then, we mapped each word in the test set to its counterpart u by *model_word* and apply *sim(w,u)* to get the similarity scores. Finally we sort the words u according to their scores. What we wonder is what will be the distribution of these scores for 1st,10th,20th,50th, 100th and 200th ranked similar words. Figure 4 shows these distributions as boxplots. As expected for the most similar word (1st) word2vec returns quite high similarity scores. What may be surprising is that there does not seem to be much difference between the scores of the 20th and 100th words and even for the 200th word there are quite a number of seed words where the score is greater than 0.7. The rightmost boxplot in Figure 4 shows the distribution of similarity scores for randomly produced 5000 pair of words to show that scores returned are relevant to detect relatedness. The good performance of the 200th words can be explained by the fact that if a word in G has that much ontological relations then we can expect that in the corpus there also many example sentences for that word and similar words. Many similar words and contexts means word2vec can do better inferences between two words and more confidently return high similarity scores. On the other hand if the test set size of a word w is small (i.e. less related pages, categories etc. in Wikipedia) than it is more likely that w does not belong to a strong ontological category. It also helps that as an encyclopaedia, Wikipedia is rich for named entities like people or place names which are much easier to be categorized and these concepts are also mentioned in our newspapers corpus frequently.

To test the relation of scores and test set size (i.e. how much connected a node in G is which in turn implies how rich its set of ontological relations) we also looked at the mean scores of the first 50 closest words for different groups of test set size. As it can be seen in Figure 5, the words with test set size greater than 250 have the best scores (mean of group 0.64, max 0.91), meanwhile when the test set size is less than 50 we get relatively worse scores (mean of group 0.44, max 0.78).

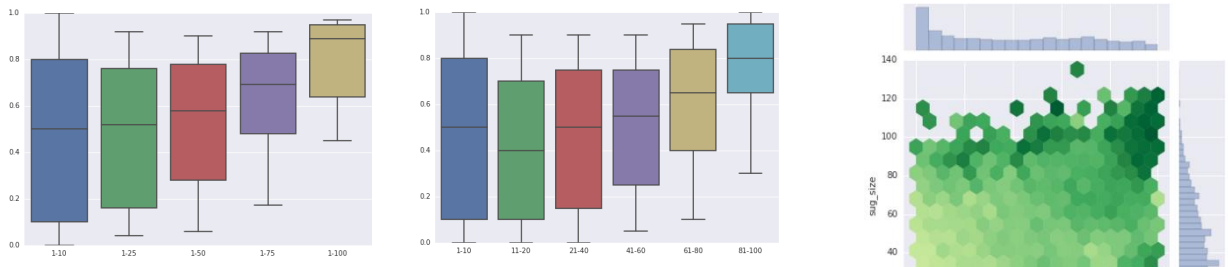
**Experiment 3:** Assuming we have concluded by the previous experiments that word2vec similarity results, i.e. closeness of word vectors, are capturing ontological relations between words, we now present an algorithm that can be used for ontology enrichment. Let's assume that in an existing ontology for a concept $w$, we already have η other related concepts. We will call this set $related(w)$. We first find out new candidates that may be related to $w$ using word2vec:

$$V(w) = related(w) \cup \{w\}$$

$$\alpha_v = \begin{cases} 1 & , if\ v = w \\ \alpha & , otherwise \end{cases} \quad (0 < \alpha < 1)$$

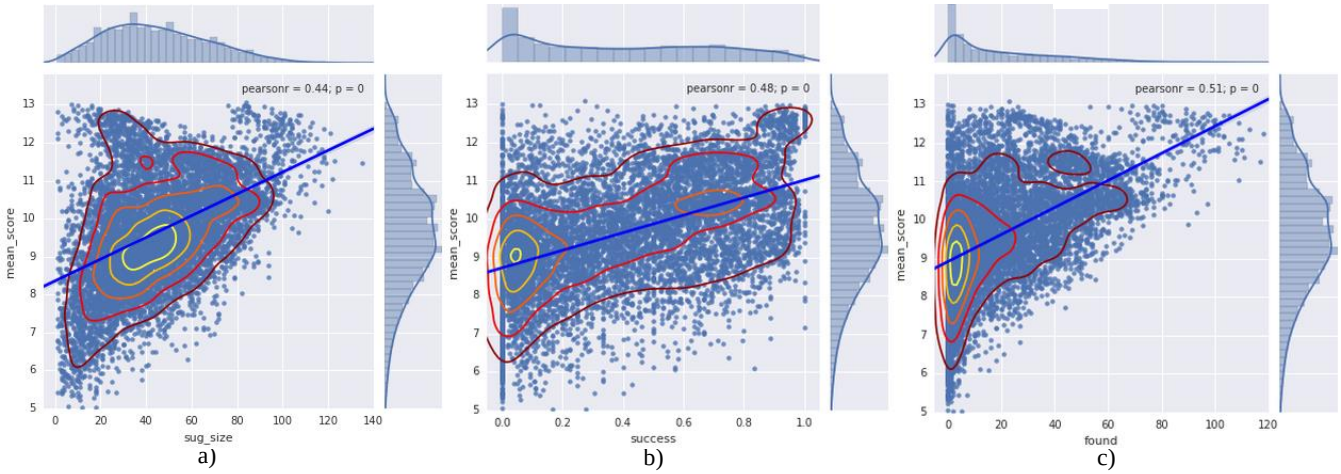
$$candidates(w) = \bigcup_{v \in V(w)} most\_similar(v, topn = \alpha_v * \delta)$$

Then for each word $u$ in $candidates(w)$ we calculate a relatedness score:




a) b) c)

**Figure 6.** Distribution of success rate results for each word: a) how success rate changes for the first 1st- nth words with highest score, b) how it changes for slices of rank (suggested words whose score puts them between nth-mth) c) how it changes according to how many words are suggested by the algorithm. Darker cells mean more results.




a) b) c)

**Figure 7.** Scatter plot, regression and kernel density estimations for comparing the distributions of $|selected_w|$, $success\_rate_w$, |success_w| (x axis from left to right) vs. $mean\_score_w$ (y axis).

$$score(u,w) = \begin{cases} \mu * sim(u,w) & ,if\ u \in most\_similar(w, topn = \delta) \\ 0 & ,otherwise \end{cases}$$

$$score(u) = score(u,w) + \sum_{\substack{r\ in \\ related(w)}} sim(u,r)$$

$$avg_w = mean(\{\, score(u)\ \forall\, u \in candidates(w)\,\})$$

$$selected_w = \{\, u \in candidates(w)\ if\ score(u) > avg_w\,\}$$

We select $u$'s as new concepts related to $w$, if $score(u)$ is higher than $avg_{score}$. The constant $\delta$ will control how many candidate words will be evaluated. Higher values can be expected to increase recall but decrease precision. $\mu$ and $\alpha$ will control how much similarities to the original word will have an effect on the score and the initial candidates set. In a more complex algorithm, these constants may depend on $w$ and selected by checking the graph properties of $w$ and $related(w)$.

To test our enrichment algorithm, first we selected 12K word nodes from G and then for each node w, selected η=12 related concepts randomly from $neighbors_{w,1}$ and used the rest of $neighbors_{w,1}$ as a hold-out set. We set $\delta$ as 400 and $\alpha$ as 0.2. We set $\mu$ as 2.5 to give a chance to some words which are not detected as very similar to $w$ by word2vec but regarded to be similar to most of the words in $related(w)$. Note that some of the words in $candidates(w)$ will not exist in $\boldsymbol{G}$ and they will be eliminated right away. With this setup and algorithm, we get the suggestions for each word w in our test set. For some of the words we tested either *model_word* did not return any results, or there were not η related concepts in $\boldsymbol{G}$ to run the algorithm, or $candidates(w)$ did not contain any words in $\boldsymbol{G}$. After these misses, we are left with results for 9K words to evaluate our algorithm. For each $selected_w$ set which contains newly discovered ontologically related concepts by our word2vec based population algorithm, we check how many of them are really in hold-out set of $neighbors_{w,1}$ and denote these successful guesses as $success_w$ and the ratio of $|success_w|/|selected_w|$ as $success\_rate_w$.

The total of newly discovered relations are 178K, where the mean of $|selected_w|$ is 44.9 and mean of $|success_w|$ is 20.5 giving us a mean $success\_rate_w$ of 40.1%. Figure 6a shows how $success\_rate_w$ is distributed according to the first nth highest scored guesses (1-10, 1-25, 1-50, 1-75, and 1-100). The boxplots show the interquartile range (25%-75%), the median, and the whiskers correspond to the 10% and 90% percentiles. As observed in Experiment 2, the more guesses returned by our algorithm the higher success rate is. This can be also observed in Figure 6b which this time shows the same distribution not cumulatively but in various rank slices (1-10, 11-20, 21-40, 41-60, 61-80, 81-100). For the words where $|success_w| \geq 80$, we can see that the median success rate jumps to around 80% even for the guesses ranked worse than 80. This phenomenon shows that there are certain types of concepts (e.g. country and politician names) which are found in our corpus both frequently and in similar contexts and these concepts are also highly connected and well covered in Wikipedia as it is being an encyclopaedia. The good news is our algorithm was successful to detect these concepts with only the help of word embeddings without any built-in language or domain knowledge. Figure 6c shows how the success rate is distributed against $|selected_w|$. The darker hexes correspond to higher values of ($success\_rate_w$ , $|selected_w|$) in our experiment's results. As it can be seen in the graph, there is a concentration of results where $success\_rate_w > 0.8$ or $|selected_w| > 75$. Next, we checked if these results are really an

effect of the good similarity scores returned by word2vec for the ontologically related concepts or not. To do that, we calculated $mean\_score_w$ which is the mean score of the first 40 (in average ~50% percent of words in a test set) highest scored guesses in $selected_w$ for each $w$ in our test set. Different values of $mean\_score_w$ for words $w_1$ and $w_2$ can be seen as a proxy of how much confident word2vec was when declaring two words as similar to $w_1$ or $w_2$. Figure7 compares distribution of $mean\_score_w$ vs. various other variables in our experiment's results. Each blue dot corresponds to an experiment data point for a particular $w$, contour lines show the kernel density estimations of the bi-variate distribution, and blue line is the linear regression line between the two variables. First, in Figure 7a at left, we looked at the correlation between $mean\_score_w$ and $|selected_w|$. We see that higher similarity scores mean more concepts are detected from $\boldsymbol{G}$. This can potentially cause more failures (i.e. detected concept is in G but not in out hold-out set), but on the contrary the $success\_rate_w$ is also increasing with better similarity results as shown in Figure7b. As a result of more guesses which are also more correct $|success_w|$ is not surprisingly correlated with $mean\_score_w$ too.

## 6. Conclusion

We show that word embeddings produced by the word2vec NN models can be quite effective for detecting ontologically related concepts and present an algorithm demonstrating how distance between word vectors regarded as similarity scores can be employed for ontology enrichment. Since this was an initial investigation of whether this methodology would be viable for ontology related NLP tasks, there are a lot of potential for improvement.

Firstly, as demonstrated in Figure 7 and discussed above, better similarity scores will result in more extracted relations which turn out to be also more correct. Word2vec has a lot of parameters and various running options that can be fine tuned to achieve better separation of vectors, thus better similarity scores. Additionally, preprocessing of the corpus can be also improved to give the model the ability to distinguish words with multiple meanings. As it can be seen in Figures 7b and 7c, we have a lot of cases where the success rate is zero dragging down our overall performance results. Manual inspection of these cases shows that some of these failures are due to Wikipedia including a word in sense a, but our corpus dominantly including the same word in sense b. Because of this observation, we believe that any corpus preprocessing that may help with word-sense disambiguation, e.g. applying morphological analysis, POS tagging etc., will eliminate most of these cases and improve performance.

Manual inspection also shows that most of the failures may be avoidable with post-processing $selected_w$. Some failures are because of common typos or different spellings of foreign names (e.g. Syria's president's last name is written as Esad or Esed in newspapers but the latter relatively new and possibly incorrect usage is not present in Vikipedi). Some of them due to offering a phrase that makes sense in our corpus (e.g. for "belgium", phrase "denmark_norway" is found as related) but have no place in an encyclopaedia as a separate concept. We used Wikipedia as golden standard to fully automate our methodology but of course Wikipedia is neither perfect nor complete. Accordingly, we also observed that our algorithm returns many concepts which could be regarded as ontologically related to the original word but marked as failure either because Vikipedi lacks the concept overall or the concept was not properly categorized or redirected by the editors. For instance, there is no redirection or disambiguation page "Morales" so when it is returned in lieu of Bolivian President "Evo Morales" by our algorithm, it is marked as a failure directly without checking the relation. Similarly, Vikipedi looks like missing many Turkish color names and colloquial medical terms. We are planning a human panel study to evaluate the extent of such failures which may be actually regarded as success.

Second source of improvements can come from using the full power of the graph $\boldsymbol{G}$ we constructed. Graph properties and algorithms like degree centrality, betweenness centrality, eigenvector centrality, random walks, clique detection etc. can be used for:

- Better detection and avoidance of category and templates which are too broad to represent meaningful ontological relations.
- Customizing the parameters of our enrichment algorithm for the current w, instead of using global constants. For instance, $\alpha_v, \delta$ can be set higher for more connected word nodes to detect more relations.

Our current algorithm detects ontological relations but does not specify which relation. This can be also solved by combining word2vec's similarity scores and current ontological knowledge in Wikipedia. Even a naïve approach like assigning the relation by comparing the discovered concept's word2vec similarities to the known related words in different categories may produce acceptable results. We concentrated on ontology enrichment in this study but with these kind of improvements and adding clustering to the mix, word2vec also shows promise to be a core component of a framework for ontology population from scratch.

## Acknowledgements

We want to thank Bekir Taner Dinçer for setting up the high performance machine on which we were able to perform this study's intensive time-consuming computations. We thank Fatma Aşık for scraping the data set we used in a clean way. Finally, our thanks goes to Radim Řehůřek for implementing and open-sourcing a fast word2vec implementation in the python Gensim topic detection library [27] and to Gordon Mohr for his very helpful suggestions and explanations on Gensim's API.

## References

[1] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. 2011. Natural language processing (almost) from scratch. The Journal of Machine Learning Research. 12:2493-537.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR.

[3] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. 2013. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems (pp. 3111-3119).

[4] Petasis G, Karkaletsis V, Paliouras G, Krithara A, Zavitsanos E. 2011. Ontology Population and Enrichment: State of the Art. In Knowledge-Driven Multimedia Information Extraction and Ontology Evolution (pp. 134-166). Springer-Verlag.

[5] Zouaq A, Gasevic D, Hatala M. 2011. Towards Open

Ontology Learning and Filtering. Information Systems. 36(7):1064-81.

[6] Tanev H, Magnini B. 2008. Weakly supervised approaches for ontology population. In Proceeding of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge (pp. 129-143).

[7] Rong X. 2014. word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.

[8] Pennington J, Socher R, Manning CD. 2014. Glove: Global Vectors for Word Representation. In EMNLP 2014 (Vol. 14, pp. 1532-1543).

[9] Ji S, Yun H, Yanardag P, Matsushima S, Vishwanathan SV. 2015. WordRank: Learning Word Embeddings via Robust Ranking. arXiv preprint arXiv:1506.02761.

[10] Le QV, Mikolov T. 2014. Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053.

[11] Barkan O, Koenigstein N. 2016. Item2Vec: Neural Item Embedding for Collaborative Filtering. arXiv preprint arXiv:1603.04259.

[12] Perozzi B, Al-Rfou R, Skiena S. 2014. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710). ACM.

[13] Vilnis L, McCallum A. 2015. Word representations via gaussian embedding. In Proceedings of International Conference on Learning Representations 2015.

[14] Arora S, Li Y, Liang Y, Ma T, Risteski A. 2015. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. arXiv preprint arXiv:1502.03520.

[15] Levy O, Goldberg Y. 2014. Neural word embedding as implicit matrix factorization. In Advances in Neural Information Processing Systems 2014 (pp. 2177-2185).

[16] Tamagawa S, Sakurai S, Tejima T, Morita T, Izumi N, Yamaguchi T. 2010. Learning a large scale of ontology from Japanese wikipedia. In Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE/WIC/ACM International Conference on 2010 Aug 31 (Vol. 1, pp. 279-286). IEEE.

[17] Wu F, Weld DS. 2008. Automatically refining the wikipedia infobox ontology. In Proceedings of the 17th international conference on World Wide Web (pp. 635-644). ACM.

[18] Janik M, Kochut KJ. 2008. Wikipedia in action: Ontological knowledge in text categorization. In Semantic Computing, 2008 IEEE International Conference (pp. 268-275). IEEE.

[19] Kim HJ, Hong KJ. 2015. Building Semantic Concept Networks by Wikipedia-Based Formal Concept Analysis. Advanced Science Letters. 21(3):435-8.

[20] Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, van Kleef P, Auer S, Bizer C. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web. 6(2):167-95.

[21] Hoffart J, Suchanek FM, Berberich K, Weikum G. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. Artificial Intelligence. 194:28-61.

[22] Hearst MA. 1992. Automatic acquisition of hyponyms from large text corpora. InProceedings of the 14th conference on Computational linguistics-Volume 2 (pp. 539-545). Association for Computational Linguistics.

[23] Maynard D, Funk A, Peters W. 2008. Using lexico-syntactic ontology design patterns for ontology creation and population. In Proc. of the Workshop on Ontology Patterns.

[24] Yeh E, Ramage D, Manning CD, Agirre E, Soroa A. 2009. WikiWalk: random walks on Wikipedia for semantic relatedness. InProceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (pp. 41-49). Association for Computational Linguistics.

[25] Zesch T, Gurevych I. 2007. Analysis of the Wikipedia category graph for NLP applications. InProceedings of the TextGraphs-2 Workshop (NAACL-HLT 2007) (pp. 1-8).

[26] Van der Maaten L, Hinton G. 2008. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research. 9(2579-2605):85.

[27] Rehurek R., Sojka P. 2010. Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.