

**UCLA**

**UCLA Electronic Theses and Dissertations**

**Title**

Zero-Shot Relation Extraction from Word Embeddings

**Permalink**

<https://escholarship.org/uc/item/4nj47424>

**Author**

Goldstein, Orpaz

**Publication Date**

2018

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

**Zero-Shot Relation Extraction from Word Embeddings**

A thesis submitted in partial satisfaction  
of the requirements for the degree of Master of Science  
in Computer Science

by

Orpaz Goldstein

2018

© Copyright by

Orpaz Goldstein

2018

ABSTRACT OF THESIS

**Zero-Shot Relation Extraction from Word Embeddings**

by

Orpaz Goldstein

Master of Science in Computer Science

University of California, Los Angeles, 2018

Professor Guy Van den Broeck, Chair

Word embeddings learned from text are well-known to capture relational information. However, extracting such relations and their associated vectors is typically performed manually, to illustrate what knowledge is embedded in the space. We propose an automated approach to mine word embeddings for sets of entities of the same type, as well as relationships that hold between them. Our approach starts from a single seed entity and extracts a relational representation from the surrounding vector space. It does so without any relational supervision. Experiments show that our extraction algorithm outperforms spectral clustering and indeed is able to extract high-quality relations from noisy embeddings.

The thesis of Orpaz Goldstein is approved.

Yizhou Sun

Majid Sarrafzadeh

Guy Van den Broeck, Committee Chair

University of California, Los Angeles

2018

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Motivation and Related Work</b>	<b>1</b>
<b>3</b>	<b>Extraction Algorithm</b>	<b>2</b>
3.1	Unsupervised Objective Function . . . . .	3
3.2	Bootstrapping using seed entity . . . . .	3
3.3	Finding neighbors in lower dimensions . . . . .	4
<b>4</b>	<b>Analysis</b>	<b>5</b>
4.1	Complexity . . . . .	5
4.2	A word on hyper parameter tuning . . . . .	6
<b>5</b>	<b>Experimental Evaluation</b>	<b>6</b>
5.1	Full Extraction Example . . . . .	6
5.2	Comparing Sets to Ground Truth . . . . .	6
5.3	Comparing Triplets to Ground Truth . . . . .	7
5.4	Information sharing . . . . .	9
5.5	1-N and N-1 Relations . . . . .	10
5.6	Non textual vector space . . . . .	10
<b>6</b>	<b>Conclusion and Future Work</b>	<b>12</b>

## List of Figures

1	2D projection of bootstrapping using seed entity . . . . .	3
2	t-SNE projection of the GloVe vector space, before and after reducing the dimensionality, with an emphasis on countries. Blue dots in the plot are countries and red triangles are demonyms. . . . .	5
3	Precision-Recall Curves: each of the extraction models on the Skip-Gram or GloVe embedded space compares with the Wikidata ground truth relationship. Our method surpasses both compared methods and is able to produce much better recall in addition to holding to high precision longer. . . . .	9

## List of Tables

1	Embedded spaces statistics . . . . .	6
2	Full extraction description for one triplet $(A, r, B)$ . . . . .	7
3	Ground truth sets statistics . . . . .	7
4	Area under curve of precision recall line. Individual sets compared with WikiData ground truth . . . . .	8
5	Area under curve of precision recall line. Relations extracted compared with WikiData ground truth . . . . .	8
6	Multimodal vector space relation extraction I . . . . .	11
7	Multimodal vector space relation extraction II . . . . .	11



# Zero-Shot Relation Extraction from Word Embeddings

## 1 Introduction

Relation extraction is the task of finding triplets  $(h, r, t)$  such that a specific relation  $r$  holds between head entities  $h$  and tail entities  $t$ . The ability to extract these triplets from unstructured corpora of text is necessary when automatically completing knowledge bases or populating knowledge graphs. Current methods tend to rely on predefined relationships, partially assembled knowledge graphs and pre-trained models that have been assembled under various levels of supervision. Relational embedding models construct a vector space that encode the relationship between entities, typically uses established public knowledge bases such as Wordnet or Freebase as input to their training phase and are therefore relying on these sources of data to be reliable and available.

Creating vector representations of words from unstructured text in models such as Skip-Gram Mikolov et al. (2013) or GloVe Pennington et al. (2014) allows us to capture some underlying pattern of the language. It is often believed that within these models, there already exists an implicit representation of relationships between groups of entities, even as they originate from an unstructured corpus of text. Most unsupervised vector-representation models use the probability of a word in text appearing next to other words as a training device to create a space where words with a similar meaning tend to cluster together. When querying such a constructed vector space, simple linear operations on entity vectors allow us to take advantage of the shape of the space and find patterns in this representation of a text corpus.

Using a vector-embedded space, we show that bootstrapping relation extraction is possible. First, we provide an algorithm that, given an arbitrary seed entity, finds sets of head and tail entities that share a relationship. Second, we provide an algorithms to extend these sets while exploiting the shape of the vector-embedded space, the common features in a set of entities, and the ability to reduce the dimensionality of the space.

A naive approach to one- or zero-shot relation extrac-

tion would be to construct sets of entities by examining the neighborhood of some arbitrary head and tail entities. While this may yield results with high recall, we are able to show that this method of constructing entity sets produces low-precision noisy extractions. Moreover, since naive construction of these sets assigns no importance to the choice of head - tail relation, we lose the ability to score extractions in terms of their relational data representation. Instead, the approach we propose in this paper is able to bootstrap and iteratively expand sets of related entities. As we show experimentally, this further improves precision and recall of our extractions tested against a ground-truth knowledge graph.

## 2 Motivation and Related Work

We are interested in automatically extracting relations between entities that exist in a corpus of text represented as a word embeddings vector space. We are specifically interested in finding multiple sets of entities that are pairwise connected by a meaningful relationship. In order to automatically discover relations in text, we will investigate the case where no prior information is given and show how bootstrapping information that will lead to relation extraction is possible.

Our approach makes no changes to the output of any of the word embeddings models used. In addition, we have no access to the original corpus of text that was used to train the word embeddings models that are used here, and use the stock pre-trained vector spaces that are presented with the models.

Zero-shot bootstrapping of triplets using only an unstructured corpus of text with no prior information allows for unsupervised translation of that text into an explicitly constructed relational representation. Contrary to knowledge base completion methods, here we do not require a pre-established source of data and are able to create a relational representation using only a corpus of text.

In Levy et al. (2017); Yan et al. (2009); Min et al. (2012) a zero-shot model is discussed, where relations could be extracted by reading comprehension over the corpus of text that is provided to the model. We believe some advantages exist in extracting relations from word embedding over extraction directly from source text: In a case where we would be interested in using a private

---

<sup>0</sup>Code accompanying this paper can be found in the following repository: [https://github.com/orggol/zero\\_shot\\_extraction\\_from\\_embeddings](https://github.com/orggol/zero_shot_extraction_from_embeddings)

data source, translation into embedding will allow us to use the word embeddings created from text without allowing access to a private text corpus. Moreover, since embeddings are already a generalized form of the entities originating in the raw text, we are able to more efficiently extract specific entities. Lastly, since extraction from word embeddings happens in a vector space, we can refer to knowledge encoded within the space in terms of a linear model. In turn this will allow us to relate such information to machine learning models and their behavior when using word embeddings.

The majority of relation extraction and knowledge base completion methods use either publicly available relational data bases or pre-established relations that are the basis for an algorithm to be built upon. Breaking free from this dependency, relation extraction will be independent of pre defined relations, and therefore will be free to discover more esoteric relations in a corpus, in addition to the more mundane relations. These specialized relations might have been otherwise missed solely due to the fact that they were not predefined or do not exist in the public data bases. Furthermore, the ability to provide a model that is not constrained by the availability of an external source of data, let alone our inability to control its quality, will add to the practicality and robustness of such a model.

Examining a vector space that is the product of an unstructured word embedding method, it is straightforward for us to manually find pairs of entities that will have a corresponding relation. In fact many examples exist in the literature of such pairs that are used to illustrate the capabilities of word embeddings vector space: [Speer et al. \(2016\)](#) uses "fire : hot :: ice : cold" as an example to SAT like analogies and Conceptnet's ability to recognize them, [Lai et al. \(2016\)](#) picks various examples to emphasize semantic properties on different word embedding models and [Levy and Goldberg \(2014\)](#) examines a number of relations to show dependency-based context extraction example. On the other hand, allowing for autonomous exploration of a vector space is normally not the emphasis of these papers. Since finding pairs of entities that share a relation in a vector space boils down to mathematical operations on vectors in a vector space, a natural observation was that we might be able to traverse a vector space while actively evaluating that which defines a relation between entities in a vector space. i.e: if we accept that a relation is a vector between two entities in a space, then we could search for other entities that share the same relation vector between them.

Once we are able to define what we believe a valid relation looks like in a vector space, we are most interested in translating a vector space into a relational representation of entities and relations. Treating this problem as a zero shot learning problem, we wish to allow our model to traverse a vector space with no initial input, and be able to consider each entity in turn as a seed entity to start our search from. Such an unsupervised approach will allow us to extract relations from a corpus of text without any need for human intervention, labeling, or preprocess-

ing of the data. Moreover, since unstructured word embedding models receive text as input and output a vector space without any preprocessing of the text, then using our method as a next step in the pipeline, we could convert a corpus of text to a relational representation in a streamlined unsupervised operation.

**Knowledge Base Completion** For the task of knowledge base completion, [Socher et al. \(2013\)](#) addresses the problem of the inability to efficiently reason with an existing knowledge base. The model in this paper converts an existing knowledge base into an embedded vector space using a neural network classifier receiving triplets of entities and their relations as training data. The resulting trained model is able to reason over learned data as well as predict new relationship triplets to be added to the knowledge base. This model is able to use word representations in order to improve accuracy.

Another knowledge completion approach, [Lin et al. \(2015\)](#) models entities and relations into two completely separate spaces. The translation from a head to tail entity (and vice versa) is performed using a predefined relation projection matrix that connects the entity space and the relation space. Both approaches make use of both Wordnet and Freebase as their input to their models and as ground truth for evaluating predicted completion of the knowledge base.

**Relation Extraction** Using a matrix factorization approach combined with surface pattern discovery and utilizing established knowledge bases, [Riedel et al. \(2013\)](#) represents the problem of relation extraction as a matrix completion task. This paper shows a model that is able to estimate, using a given head, tail entities and a relation, the probability that the head and tail entities are suitable to the relation.

A distant supervision approach to relation extraction is described in [Mintz et al. \(2009\)](#). In this paper a model is trained to extract new relations using the Freebase knowledge base and a Wikipedia dump. This model is able to extract new relations that do not appear in Freebase using a lexical and syntactical extraction, and the distant supervision assumption that if two entities participate in a relation, any sentence that contain those two entities might express that relation. Another distant supervision model is described in [Min et al. \(2013\)](#). Extending the Multi-Instance Multi-Label model, and learning from positive and unlabeled bags. By treating unlabeled data as unlabeled instead of negative examples, they are able to improve results from earlier MIML models.

### 3 Extraction Algorithm

Our goal is to construct triplets  $(A, r, B)$  where  $A, B$  are matrices, and each column vector in  $a_i \in A$  is a head entity corresponding to a tail entity represented as a column vector in  $b_i \in B$ . There exists one relation vector  $r$  that transfers from entity  $a_i$  to entity  $b_i$  for each  $i$ . An example to this would be,  $A = \{\textit{Belgium, England, France}\}$ ,

$B = \{\text{Belgian, English, French}\}$  and the  $r$  vector representing the *demonyim* relation.

$$\begin{array}{c}
 \begin{array}{c} N \text{ Head entities} \\ \begin{bmatrix} w_{11} & \cdots & w_{1N} \\ w_{21} & \cdots & w_{2N} \\ \vdots & \vdots & \vdots \\ w_{D1} & \cdots & w_{DN} \end{bmatrix} \\ \text{Dimensions} \end{array} \\
 \begin{array}{c} \text{Relation} \\ \text{Vector} \\ \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_D \end{bmatrix} \\ r \end{array} \\
 \begin{array}{c} N \text{ Tail Entities} \\ \begin{bmatrix} w_{11} & \cdots & w_{1N} \\ w_{21} & \cdots & w_{2N} \\ \vdots & \vdots & \vdots \\ w_{D1} & \cdots & w_{DN} \end{bmatrix} \\ \text{Dimensions} \end{array}
 \end{array}
 \quad A \quad B$$

Using a constructed vector-embedded space such as the outputs of Skip-Gram or GloVe, we show that it is possible to bootstrap these sets, using the dimensionality and the composition of these spaces. Since in contrast to other relation extraction solutions, we will not use a pre-made bank of entities and relations, we show first that using an arbitrarily chosen initial seed could lead to the discovery of initial sets of entities bound by a single relation. Using these initial sets we expand and discover additional entities from each of the sets that fit the initial definition of the triplet  $(A, r, B)$ .

### 3.1 Unsupervised Objective Function

The ability to grade the quality of a discovered set is essential for us to be able to compare these sets to one another. A method of scoring we propose will grade the level of adherence of entities in bootstrapped sets to the relation that was used to start the process. That is, we will use the difference in position between where we search for an entity and the position where it is actually found. The following scoring function acts as our unsupervised objective function, where a 0 error would imply no difference existed when retrieving entities using the relation vector.

As part of bootstrapping our triplets  $(A, r, B)$ , we score the quality of our set by measuring the distance between the presumed position for a head entity (a vector in the same set as our seed entity) reached by adding the relation vector  $r$  to the tail entity vector (a vector in the same group opposite the seed), and the actual entity vector discovered by locating the nearest entity to that position. The error of the entire triplet  $(A, r, B)$  will be therefore

$$\text{Error} = \frac{1}{n} \sum_{i=1}^n |b_i + r - a_i|, \quad \forall a_i \in A, b_i \in B$$

This error gives us a measure of accumulated distance traveled in the embedded space in order to assign entities that will fit both sides of our relation. This error is used to refine our search for entities in addition to being a criteria for choosing which discovered sets we wish to expand.

### 3.2 Bootstrapping using seed entity

let  $X$  be our word embedding matrix and  $w$  be a randomly chosen entity/vector representing a word from our text where  $w \in X$ , and let  $Sim(w)$  be a set of the  $n$  most similar vectors in  $X$  to  $w$  based on cosine similarity measure.

$$Sim(w) = \{Top-n_{v \in X}(\cos(w, v))\}$$

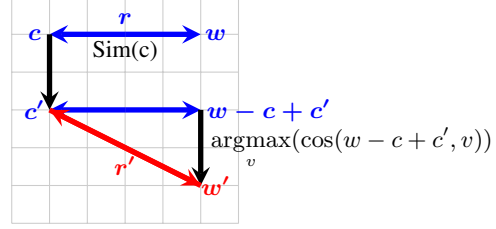


Figure 1: 2D projection of bootstrapping using seed entity

Next, we will treat vectors from  $Sim$  as a set of candidate entities for sharing a relation with  $w$ . Since in word embedding models, words that appear in similar contexts cluster together, it is fair to assume that words in the same neighborhood will likely share a relation. In order to examine a pair of entities and establish whether their relationship could be used to bootstrap finding our triplets, we will try and identify additional pairs of entities that are both similar to the original and candidate entities, as well as share their relationship. A valid pair therefore will have a relationship vector that is as similar as possible (above some constant  $\alpha$ ) to the original pair's relationship vector.

Concretely, let  $c$  be a candidate vector,  $Sim(c)$  the set of  $n$  most similar vectors to  $c$ . Let us look at one vector  $c' \in Sim(c)$  and use vector math to retrieve  $w'$ .

$$w' = \underset{v \in X}{\operatorname{argmax}}(\cos(w - c + c', v))$$

Now that we have two head-tail pairs of one potential relation we can measure its accuracy. Naming the two pairs we have  $rel = w - c$  and  $rel' = w' - c'$ . A high cosine similarity of these vectors will hint a real discovered relation. We will define some  $\alpha$  such that:

$$rel \text{ is a relation if } \cos(rel, rel') \geq \alpha$$

Repeating this process for every  $c' \in Sim(c)$  we end up with a set of pairs of entities that have a common relationship. A triplet  $(A, r, B)$  is considered good if it has more than two entities in it.

**Example** Let us choose an arbitrary seed  $w = sailing$ . Looking at  $Sim(w)$  where  $n = 1$ , we will use the closest entity to  $w$  as our candidate vector,  $c = surfing$ . Looking at  $Sim(c)$  where  $n = 2$ , we find the two closest entities to  $c$  which we discover to be  $\{surf, surfer\}$ . Now that we have  $w, c$  and  $c'$  we can finally find  $w'$ . Starting with  $c' = surf$ , by using operations on the vector space as outlined above, we move in the vector space to find that  $w' = sail$ . Comparing  $rel$  and  $rel'$  we find that their cosine similarity is above our defined  $\alpha = 0.6$  and therefore is considered a discovered relation. Next we follow the same steps when  $c' = surfer$  to receive  $w' = sailboat$ . At this point our sets look as follows:  $A = \{sailboat, sailing, sail\}$ ,  $B = \{surfer, surfing, surf\}$  and  $r$  is the vector representing the unnamed relation between the two sets. Our example triplet  $(A, r, B)$  had the best objective function score amongst the sets bootstrapped using the seed  $w = sailing$ .

### 3.3 Finding neighbors in lower dimensions

After we have found our initial triplet  $(A, r, B)$ , we would like to increase the size of each of the sets in order to discover as many similar entities as possible from each of our sets. To achieve this, we will use the common features of entities in each of the sets in order to find more similar entities to those that were found during the bootstrapping phase. Reducing the space of  $X$  to contain only the most similar common features of the entities in each set to produce  $X'$  we can then continue to search the space for similar entities and score them.

In order to produce  $X'$  we examine each of the constructed entity sets from our triplet separately. For each of the sets, we wish to investigate which features are common between the entities in the set and are most important in defining the entities who belong in that set. Using pairwise combinations, where we consider every possible combination of two entities on the set, we extract the most similar features for each such combination and define a *Mask* for every combination, which is a subset of features that are most similar between the two entities in the combination. Looking at the intersection of all such *Masks*, we receive a subset of the features most suitable to reduce the original space into. Let  $C$  denote the set of all pairwise combinations of entities from a set we are expanding. Then for each pair  $(c_{i1}, c_{i2}) \in C$  we will extract

$$Mask_i = Bottom\_n(Abs(c_{i1} - c_{i2}))$$

In effect  $Mask_i$  consists of the  $n$  features that are closest to 0 in the above calculation.

The final step to complete the search for a subset is looking at the intersection

$$Int = Mask_1 \cap Mask_2 \dots \cap Mask_n$$

Using  $Int$  we reduce the original space  $X$  into a subspace  $X'$  containing only the features that are in  $Int$ .

Using the reduced space  $X'$  we can continue searching for similar entities more efficiently. Looking in the neighborhood of each of the entities in our set, we compare each of the neighbors to the mean vector of all the entities in the set. A neighbor that scores above some  $\beta$  is appended to the set. Let  $A$  be the set we are expanding and  $\hat{m}$  be the mean vector of  $A$ . then for each  $a \in A$

$$Neighbor = Top\_n_{v \in X'}(\cos(a, v))\}$$

$Neighbor$  is the set of nearest neighbors of  $a$  in the new set  $X'$  and for each  $z \in Neighbor$ ,  $z$  is a valid addition to the set if it is similar to  $\hat{m}$  above some defined threshold  $\beta$

$$z \text{ is valid if } \cos(z, \hat{m}) \geq \beta$$

One problem that could arise from intersecting a large number of *Masks*, is that the intersection might become extremely small, and therefore will have an aggressively reduced dimension. Such a reduced dimension might in turn cause our expansion process to become inefficient as multiple features might be ignored. However, since the size of *Masks* directly relate to the quality and size of our

bootstrapping, we can say that controlling  $\alpha$  will affect the number of features considered in expansion. That is, lowering our  $\alpha$  will increase sizes of bootstrapped sets, which in turn will produce smaller *Masks* and smaller expansions. Conversely increasing  $\alpha$  will produce smaller bootstrapped sets, larger *Masks* and larger expansions.

**Detailed Example** All the examples used in the bootstrapping and expansion of sets are real world experiments, conducted using the GloVe vector-embedded space.

**Phase 1** Using our previously discovered set  $\{sailboat, sailing, sail\}$ , we will have 3 combinations of pairs:  $(sailboat, sailing)$ ,  $(sailboat, sail)$ ,  $(sailing, sail)$ . Extracting a maximum of  $n$  common features from each pair of vectors representing the pair of words we get:

$$Mask_1 = Bottom\_n(Abs(sailboat - sailing)),$$

$$Mask_2 = Bottom\_n(Abs(sailboat - sail)),$$

$$Mask_3 = Bottom\_n(Abs(sailing - sail)).$$

**Phase 2** Using the 3 extracted sets of common features, we now can calculate  $Int$ , the intersection of 3 sets of common features.  $Int = Mask_1 \cap Mask_2 \cap Mask_3$ . The result is a set of features that exist in all possible pairs of  $\{sailboat, sailing, sail\}$ .  $Int$  will be used next to reduce the original vector space to contain only the features in  $Int$ .

Figures 2a and 2b show how dimension reduction affects the entities we are interested in extracting. using t-SNE [van der Maaten and Hinton \(2008\)](#) visualization tool for high-dimensional data, we are initializing 50 random 'country' vectors and 20 random 'demonym' vectors using their 2 principle components. Cosine similarity is used to measure distances between vectors, and determines the organization of neighbor visualization in our plots. As we can see in the plots, in the original space our target group 'countries' can be seen fairly mixed with a separate group 'demonym'. After reducing our original space based on features that appeared in the country entities we bootstrapped, we can see the country entities converge nicely when compared to the same demonym entities that are now better distinguished compared to the original space.

**Phase 3** In our example  $A = \{sailboat, sailing, sail\}$ . The mean vector  $\hat{m}$  is the mean of the three vectors representing the words in  $A$ . In the case where  $n = 5$ , we are looking for the 5 closest neighbors of each of our entities in  $A$  in the new vector subspace  $X'$ . For *sailboat* we find the 5 nearest neighbors are  $\{sailboat, sail, yacht, sailing, catamaran\}$ . We continue doing the same for *sailing* and *sail*. All entities' cosine similarity with the mean vector  $\hat{m}$  is above our chosen  $\beta = 0.6$  and are therefore considered valid. Following



the same steps for  $B = \{surfer, surfing, surf\}$  in the same case where  $n = 5$ , we find nearest neighbors for *surfer* and *surfing*. For *surf* we find the 5 nearest neighbors are  $\{surf, surfing, surfer, surfers, surf\}$ . All entities but one's cosine similarity with  $B$ 's mean vector  $\hat{m}$  is above our chosen  $\beta = 0.6$  and are therefore considered valid. *surfs* however had a cosine similarity of 0.58 with the master set's mean vector and therefore is considered invalid and is not added to the expanded group.

## 4 Analysis

### 4.1 Complexity

Unsupervised extraction of our relational sets, would potentially need to pass every single entity in the space to consider it as a seed for our bootstrapping process. Given that a well trained word embedded space typically has a vocabulary in the order of millions of words and that a simple iterative search for words of a similar nature or words that share a potential relation could be quadratic in the size of the embedded space, any naive solution will therefore be unusable as a practical model.

Our method of searching relations is fairly flexible, and could be increased or decreased in complexity by carefully choosing the size of neighborhood we consider in each of our algorithms steps. For each of the two main steps of our algorithm, we will expand on the complexity.

#### Bootstrapping using seed entity

We start with obtaining  $Sim(w)$  for our seed value by way of cosine similarity or matrix-vector multiplication. Our matrix contains a vocabulary of  $n$  vectors. Each vector has a dimension of  $d$ . Therefore  $Sim(w)$  will cost  $\mathcal{O}(nd)$ . Keeping a constant number of nearest neighbors ( $n=100$  was used for the results in this paper). For each of the neighbors we then similarly obtain  $Sim(c)$  for the same complexity of  $\mathcal{O}(nd)$ . Again keeping a constant number of neighbors ( $n=10$  for this paper), we finally obtain  $w'$  again for each of them, costing us  $\mathcal{O}(nd)$ . Since all other filtering and calculations are done in  $\mathcal{O}(1)$  time, we can calculate  $\mathcal{O}(nd + c_1nd + c_2nd)$  for a final complexity of  $\Theta(nd)$

#### Finding neighbors in lower dimensions

Expanding our sets involves obtaining some information about common features of the entities in our sets, and performing calculations on all pairwise combinations of entities. Since it is fair to assume that each bootstrapped set is fairly small, and never in the order of an entire vector-embedded space, we will treat it here as a constant number of entities. Therefore, calculating the mean vector of each of our sets as well as finding all pairwise combinations will all take  $\mathcal{O}(1)$  time. The costly part of expanding our sets is finding the *Neighbor* set previously described. We need to find a *Neighbor* set for each of our pairwise combinations of our entities in the bootstrapped set, costing us  $\mathcal{O}(nd)$ . Keeping a constant number of neighbors ( $n=1000$  was used for this paper), we then check validity for each of the neighbors for a low  $\mathcal{O}(1)$

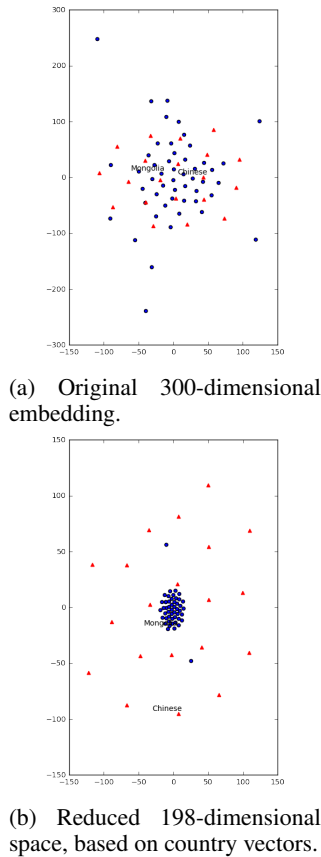


Figure 2: t-SNE projection of the GloVe vector space, before and after reducing the dimensionality, with an emphasis on countries. Blue dots in the plot are countries and red triangles are demonyms.

time. The final calculation for this is then  $\mathcal{O}(c_1 * nd)$  for a final complexity of  $\Theta(nd)$

## 4.2 A word on hyper parameter tuning

In this section we have introduced both  $\alpha$ , that is a threshold parameter in the bootstrapping process. And  $\beta$ , that is a threshold parameter in the expanding process. Each of the two parameters carries a different meaning to the relation extraction process and can be tuned to fit a specific use-case.  $\alpha$  is the threshold acting as the acceptance criteria of newly discovered (head, tail) pairs that share the original relationship with the bootstrapped set. Higher  $\alpha$  will reduce the amounts of sets a specific seed entity produces as well as the number of pairs accepted into the set. consequently sets produces will usually carry a more well defined relation.  $\beta$  is the threshold controlling acceptance of entities discovered in the subspace during the expansion phase. Increasing this parameter will result in more entities collected during expansion, but will increase the chance of noise treated as valid entities. In addition to  $\alpha$  and  $\beta$ , we use  $top_n$  in three places:  $Sim(w)$ ,  $Sim(c)$ , and in finding *Neighbors*. Values for  $n$  are hardcoded as 100,10,1000 respectively, and were chosen based on our testing for specific relationships with entities that cluster within these ranges. A valid  $n$  could be any number of neighbors we are interested in investigating, up to the amount of entities in the vector space.

For this paper, we were interested in tuning  $\alpha$  to produce enough sets from a seed entity, so that the specific relation we were interested in would be included in the bootstrapped sets. As a first step, we will check a random set of seed values and try to find a general rule for a value that corresponds with a certain amount of relations found, following that with identifying a more specific rule for our needs regarding the testing in this paper. Initially we generated a random array of 10 english words:  $\{real, rich, whimsical, route, action, giddy, clean, identify, smoke, attend\}$  and ran the bootstrapping process, looking for a value which will produce a minimum of 2 sets containing 2 distinguishable relationships on both Skip-Gram and GloVe embedded spaces. Gradually descending  $\alpha$  values from 0.9 in multiples of 0.1. Once we hit 0.5 we were able to produce 2 separate relational sets for all entities. As a next step, we wanted to refine this result for our experiments, and therefore repeated the experiment with arbitrary country and demonym entities, this time increasing  $\alpha$  from 0.5 in multiples of 0.05. This time we were able to get away with  $\alpha = 0.6$  for the same desired results.

Our tuning of  $\alpha$  in this case, was an optimal constraint on a solution size we were willing to accept for this paper. A future user of this model might run bootstrapping multiple times with varying  $\alpha$  values in order to evaluate what solution size is optimal for their own agenda.

The effects of tuning  $\beta$  will be explored in the experiments phase, where it will be used to shift the balance between precision and recall.

Table 1: Embedded spaces statistics

Model	Dimensionality	Vocabulary
Skip-Gram	300	3M Words
GloVe	300	2.2M Words

## 5 Experimental Evaluation

The experiments in this paper are conducted using two of the most popular models for creating vector embeddings from text: Skip-Gram and GloVe to test our method of bootstrapping relation extraction. Both models take in unstructured text corpus as input and outputs a vector-embedded space. Each entity in the resulting space is a vector representing a single word. Vector representation of words tend to cluster with words of similar meaning. In order to show that our method generalizes, we used stock pre-trained model for each of the two models. Each of the two models are evaluated on bootstrapping triplet sets  $(A, r, B)$  using an arbitrary seed entity, followed by expansion of the discovered sets.

### 5.1 Full Extraction Example

Table 2 gives us a glimpse into how our relational representation is bootstrapped and expanded. Showing the entities which are bootstrapped in bold text and entities which are expanded in plain text, we are able to see the different stages of constructing our sets. Entities who have no corresponding tail/head in the other set, are entities who might still have a match on the other set, but are a weaker match than another entity. As an initial small bootstrapped set is expanded, a few things are worth noting. We can see that as an initial bootstrapped set is expanded, it picks up many more similar entities. And while these entities are in most cases compatible with the common theme of the set, some entities are picked which we choose to count as noise, although in some cases the gist of the entity fits as a part of a relational triplet. For example, as can be seen in table 2, some entities are picked during expansion that are a misspelling of the original entity, others are named correctly in the original entities' language. Another option are initials or a different common name for the same entity. All these are counted as invalid entities, together with actual unrelated entities, in our following experiments and as we compare with various data sets. If we have counted these related entities in our experiments, keep in mind our results would have been higher.

### 5.2 Comparing Sets to Ground Truth

In order to measure the quality of the sets discovered, we use Wikidata; A free community edited knowledge base that aspires to provide a queryable source for information mining. Wikidata is the latest and largest collaboratively edited knowledge base that provides a documented-oriented database where each entity has a unique identifier and its information statements take the form of key value pairs, allowing us to mine certain types

Table 2: Full extraction description for one triplet  $(A, r, B)$

seed $w = \text{belgium}$		
	country set	demonym set
<b>Skip-Gram</b>	<b>Belgium</b>	<b>Belgian</b>
	Belguim	-
	<b>Croatia</b>	<b>Croatian</b>
	<b>France</b>	<b>French</b>
	<b>Italy</b>	<b>Italian</b>
	<b>Switzerland</b>	<b>Swiss</b>
	Algeria	Algerian
	Argentina	Argentine
	Czech_Republic	Czech
	Denmark	Danish
	Serbia_Montenegro	Serbian
	Morocco	Moroccan
	Slovakia	Slovak
	-	Slovakian
	:	:
:	:	
extradites_Noriega	-	
<b>Glove</b>	<b>Belgium</b>	<b>Belgian</b>
	Belgique	-
	<b>Austria</b>	<b>Austrian</b>
	<b>Sweden</b>	<b>Swedish</b>
	<b>Finland</b>	<b>Finnish</b>
	Helsinki	-
	Denmark	Danish
	Europe	European
	Iceland	Icelandic
	U.K.	-
	:	:
	:	:
	Sweden	Swedish
	-	Sweedish

\* Bold text are bootstrapped entities. The rest are expanded

Table 3: Ground truth sets statistics

Set	Size
country	204
demonym	241
capitals	325
airport-codes	373
country→demonym	245
city→NFL	32
country→capital	204

of information. For the following experiment we created sets for all 'countries', 'demonyms', 'capitals' and 'IATA airport codes' that exist in Wikidata to be compared to constructed sets. These sets were chosen since in our experiments we have found these four categories are possible to extract both on Skip-Gram and GloVe using the same seed entity.

Since we wanted to compare our model to some baseline algorithm, we compared our bootstrapping method to two clustering algorithms; the naive approach, k-nearest neighbors, and the more robust spectral clustering which uses eigenvalues in order to reduce dimensionality before performing clustering on the k-nearest neighbors of our seed entity. In order to choose the best cluster coming out of spectral clustering we manually print and count which cluster has the most relevant entities and use that cluster in our comparison. Both clustering algorithms were compared on each of the two vector-embedded spaces used in our experiments.

### Skip-Gram

In this section all results were produced using the vector space that was produced by the Skip-Gram model, was pre-trained by as described in *Distributed Representations of Words and Phrases and their Compositionality* (Tomas Mikolov et al. -2013), and is available for download [here](#).

### GloVe

In this section all results were produced using the vector space that was produced by the GloVe model, was pre-trained by as described in *GloVe: Global Vectors for Word Representation* (Jeffrey Pennington et al. -2014), and is available for download [here](#).

### 5.3 Comparing Triplets to Ground Truth

Another important measure of the relational representation we are able to extract, is the comparison between our triplets  $(A, r, B)$  and some ground truth of the specific relation  $r$  we are describing. As our ground truth, we extracted all data available on Wikidata of a specific relation. In our case, we are evaluating the  $country \rightarrow demonym$  and  $country \rightarrow capitals$  triplets as well as the  $City \rightarrow NFL\ team$ .

Once we find our  $(A, r, B)$  representation, we wish to evaluate the degree in which our set corresponds with the Wikidata representation of the same relation. Using our

Table 4: Area under curve of precision recall line. Individual sets compared with WikiData ground truth

AUC-PR - Individual sets				
	demonym set	country set	capitals set	airport-code set
<b><i>Skip-Gram</i></b>				
Nearest Neighbors	0.04	0.018	0.001	-
Spectral Clustering	0.038	0.014	0.001	-
Our Model	0.21	0.135	0.058	-
Our Model + Hopping	0.29	0.174	0.075	-
<b><i>GloVe</i></b>				
Nearest Neighbors	0.08	0.133	0.02	0.088
Spectral Clustering	0.043	0.08	0.03	0.08
Our Model	0.25	0.395	0.138	0.127
Our Model + Hopping	0.25	0.589	0.147	0.2

Table 5: Area under curve of precision recall line. Relations extracted compared with WikiData ground truth

AUC-PR - Triplets ( $A, r, B$ )			
	country $\rightarrow$ demonym	city $\rightarrow$ NFL team	country $\rightarrow$ capital
<b><i>Skip-Gram</i></b>			
Nearest Neighbors	0	0	0
Spectral Clustering	0	0	0
Our Model	0.1312	0.1921	0.0543
Our Model + Hopping	0.1312	0.1990	0.066
<b><i>GloVe</i></b>			
Nearest Neighbors	0	0	0
Spectral Clustering	0	0	0
Our Model	0.1704	0.1705	0.1875
Our Model + Hopping	0.1708	0.1798	0.248



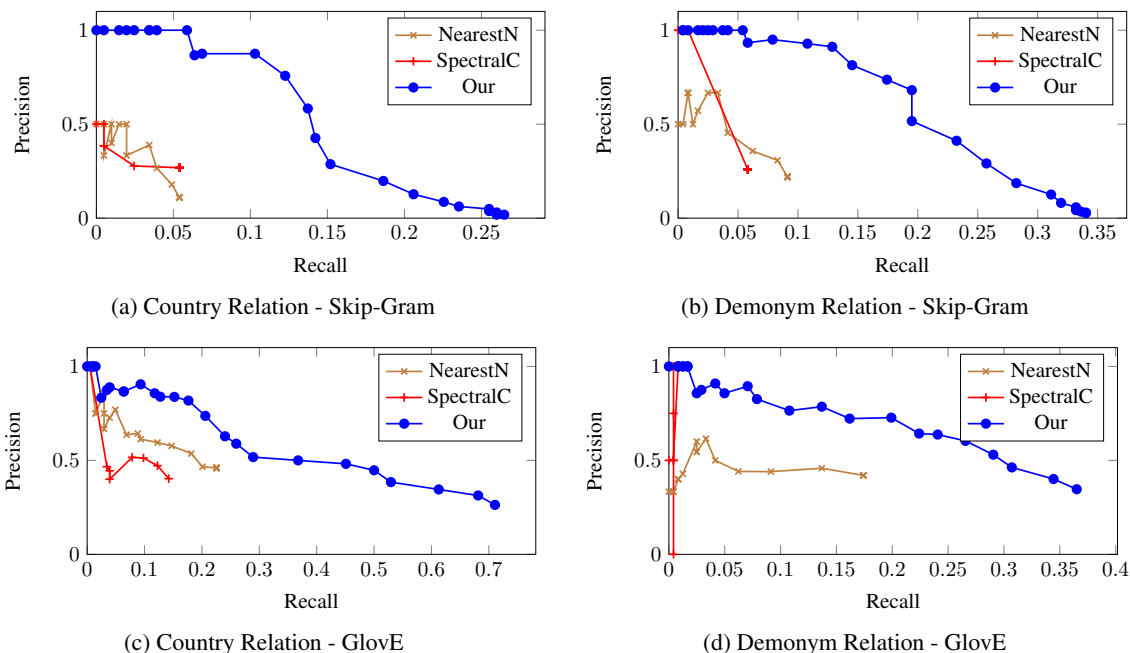


Figure 3: Precision-Recall Curves: each of the extraction models on the Skip-Gram or GloVe embedded space compares with the Wikidata ground truth relationship. Our method surpasses both compared methods and is able to produce much better recall in addition to holding to high precision longer.

expanded Head set  $A$ , we perform a check for each head entity and verify a corresponding tail entity in Tail set  $B$ . For all triplets that adhere to this symmetry, we calculate the precision and recall compared to our ground truth.

In table 4 we can see results for our AUC-PR when comparing with Wikidata. Both relations examined give very good AUC-PR results. In the case of  $City \rightarrow NFL\ team$  our results are really impressive. We are able to hold very high precision as we move up in recall. More specifically, our precision is around 70% accuracy all the way to 25% recall.

When comparing to baseline algorithms: Nearest Neighbors and Spectral Clustering, we have a way of constructing sets of entities which are the result of both clustering algorithms, but we do not have an instance of a relationship vector collected as part of these algorithms. In order to be able to compare the results of these to our ground truth, we used the same relationship vector that our model used in each of the examples to try and find a corresponding tail entity for each of the entities in the head set.

## 5.4 Information sharing

Once we have some established relational representation that encompasses a few relations and sets, we are interested in finding whether it would be possible to "hop around" these sets and improve our results even more using information from a distant set that might have no immediate connection to the improved set. Our assumption in this experiment is that given two relations  $r_1, r_2$ , each with head and tail set  $H_1, H_2, T_1, T_2$  where  $T_1$  is a set

with similar entities to  $H_2$ . In other words: the tail set of one relation is of the same category as the head set of another relation. Using these sets will allow us to do two things; use the transitive relation to expand our relational representation. i.e: if we have a relation  $a \rightarrow b$  and  $b \rightarrow c$  then transitively  $a \rightarrow c$ . In addition, since the duplicate category sets were constructed using different seed values, they therefore might benefit from the information that exists in the mirror set.

### Expansion by hopping

For the purpose of this experiment we will use our benchmarked sets (*demonyms* set and *countries* set), and find a new relation that will have either *countries* or *demonyms* as a head or tail set, and use the new set to run our experiment.

Using the GloVe embedded space and passing *Amsterdam* as a seed, we are able to find the relation "Capital of", where our head set consists of capitals and our tail set consist of countries. By identifying this kind of situation where in our case we currently hold sets of categories:  $capitals \rightarrow countries$  and  $countries \rightarrow demonyms$ , we are able to use the join of the bootstrapped entities on the duplicate set and expand the joined set of countries to receive a more robust final set as can be seen in figure 6. By using the entities in the two similar sets, we expect to get a more inclusive result set that have been expanded using the unique features that the two separate sets are likely to carry, and therefore aid in successfully enhance our extraction process.

In order to validate the generality of this experiment we repeated the steps outlined above on the Skip-Grap

embedded space. Again using *Amsterdam* as seed, this time we have found a *capitals*  $\rightarrow$  *demonyms* set and used that in the same manner of the previous example, only this time expanding the *demonym* set. The improvement in results can be seen in figure 7.

As we can see from the two examples, as well as from further investigation as can be seen in table 3, in most cases hopping successfully further enhances the preliminary expansion of sets and provides better AUC results.

### Expansion by transitivity

Another information we wish to make use of, once we have an initial relational representation, is the ability to establish a transitive relation between sets. Using the same example we enhanced by hopping between sets, we can show that using the two bootstrapped sets: *capitals*  $\rightarrow$  *countries* and *countries*  $\rightarrow$  *demonyms*, we can construct a third representation solely by using exiting information. Using the transitive rule where if  $a \rightarrow b$  and  $b \rightarrow c$  then transitively  $a \rightarrow c$ , we can connect entities from our *capitals* to entities in our *demonyms* set. Let *rel1*, *rel2* represent the relation between *capitals* and *countries*, *countries* and *demonyms* respectively. and *vec* be the vector representing the word *paris*. Using simple arithmetic, we are able to create a bridge between *vec* and its corresponding transitive relation in the *demonyms* set. i.e: by calculating  $target = vec + rel1 + rel2$ , and looking at the  $\operatorname{argmax}_{v \in X}(\cos(target))$  we can expect to get the vector representing the word *French*. Therefore, given that we are able to identify sets of similar entities, we could use previously established sets to uncover these transitive relations between any two sets that comply with the transitive relation rule.

## 5.5 1-N and N-1 Relations

While our model is geared towards constructing a 1-1 relational representation, a simple change in the bootstrapping process will allow us to focus collection on many to one or one to many relationships. If in the original model, our *c* and *c'* tail entities were extracted separately to encourage the bootstrapping of discrete entities, when looking for many-one relations we will enforce  $c = c'$ , while still producing head entities such that  $w \neq w'$ . Otherwise using the same algorithm to bootstrap our one-many sets, we will get a many set on *A* and a one set on our *B*.

### Example

Using *Lyon* as a seed on our modified algorithm we get  $A = \{Grenoble, Lille, Lyon, Marseille, Montpellier, Nantes, Rennes, Toulouse\}$  and  $B = \{France\}$ .

Normally our next step would be to expand both sets. However as this is a one-many relationship, it is enough in this case to continue and expand only *A* and treat the expanded set as the many set relating to  $B = \{France\}$ .

## 5.6 Non textual vector space

In this paper we have evaluated our model solely on spaces embedding textual information. We are interested however to find out whether our approach could generalize to extract relations from spaces encoding information

other than text. Using the the embedded space used in [Kiros et al. \(2014\)](#), we run our model on the vector space trained by using the Flickr8K image dataset that comes with 8,000 images, with each image annotated using 5 sentences composed by independent annotators. In table 6 and 7, we can see 2 separate bootstrapped triplets. Although some visual relationship in our triplets might be obvious to us, and guessing could be done as to what the actual relation vector might be between the head and tail sets, the relation will largely stay unnamed. Since the multimodal embedded space encodes a combination of different types of data as a single vector encoding, it will be hard for us to determine exactly what the relation between the head and the tail might be. However, since convolutional neural net is used to encode the images, and a LSTM machine is used for the textual portion of the encoding, we can try and infer some overall relationship between the sets that spans both visual and textual relationships. For example: in table 6, we can see the results of seeding an image of a dog running. In the resulting representation both our head and tail entities depict the same breed of dog (or two that are extremely similar). While in this case the relationship appears more obvious, In table 7, seeding an image of a biker on a dirt trail, we get a bootstrapped set of bike riding in various situations. Similarly to the first case, our model captures similarity, this time in the scenes in head and tail sets, and in addition to the similarity in the settings, it appears that our model captured a relationship where the rider and bicycle are mirrored between the head and tail sets.

Table 6: Multimodal vector space relation extraction I



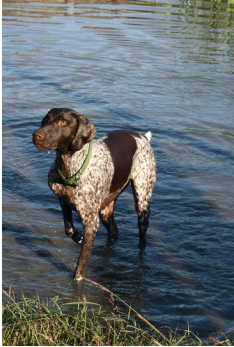
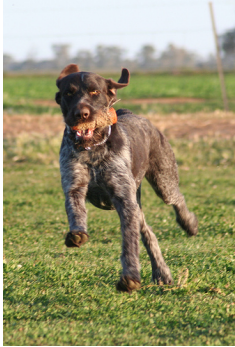








Head Set	Tail Set
	
	
	

Table 7: Multimodal vector space relation extraction II

Head Set	Tail Set
	
	
	

## 6 Conclusion and Future Work

In our this paper, we have suggested a zero shot relation extraction method, that allows us to generate a relational representation using vector-embedded spaces. Allowing us to potentially use our method in order to generate the maximum amount of relational information from a generated vector space by iterating over all entities in the space. We have shown that even given no prior information our model shows great results when compared to naive baseline and to current online relational knowledge bases.

As next steps, the ability for our model to identify duplicate entities such as misspelling of a word or different names for the same entity, will greatly increase our accuracy. In addition, enhancing the effectiveness of information sharing between established relational sets. Using more shared information will allow us to identify entities that have been missed in our expansion stage and add them retroactively. Another aspect we would like to address is our ability to filter entities that were collected during expansion but are considered noise and are irrelevant to the set. This could be done by removing entities with no relevant corresponding entity that is valid in the opposite set, or by removing entities that are not the first match to an entity in the opposite set. Filtering out noise from our sets will greatly increase robustness of our model.

## References

- Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014). Unifying visual-semantic embeddings with multi-modal neural language models. *CoRR*, abs/1411.2539.
- Lai, S., Liu, K., He, S., and Zhao, J. (2016). How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.
- Levy, O. and Goldberg, Y. (2014). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308.
- Levy, O., Seo, M., Choi, E., and Zettlemoyer, L. (2017). Zero-shot relation extraction via reading comprehension. *CoRR*, abs/1706.04115.
- Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI’15*, pages 2181–2187. AAAI Press.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.
- Min, B., Grishman, R., Wan, L., Wang, C., and Gondek, D. (2013). Distant supervision for relation extraction with an incomplete knowledge base. In Vanderwende, L., III, H. D., and Kirchhoff, K., editors, *HLT-NAACL*, pages 777–782. The Association for Computational Linguistics.
- Min, B., Shi, S., Grishman, R., and Lin, C.-Y. (2012). Ensemble semantics for large-scale unsupervised relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL ’12*, pages 1027–1037, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2, ACL ’09*, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Riedel, S., Yao, L., Marlin, B. M., and McCallum, A. (2013). Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL ’13)*.
- Socher, R., Chen, D., Manning, C. D., and Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc.
- Speer, R., Chin, J., and Havasi, C. (2016). Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975.
- van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.
- Yan, Y., Okazaki, N., Matsuo, Y., Yang, Z., and Ishizuka, M. (2009). Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2, ACL ’09*, pages 1021–1029, Stroudsburg, PA, USA. Association for Computational Linguistics.