

Final project
Computational semantics 2019/2020

Hypernym Discovery: SemEval2018 - task 9

**Hypernym discovery with K-nearest:
a comparison of results in English and Spanish**

Ana Mestre
Nengjing Ren
Jonatan Koren

07/01/2020

Table of contents

1.	Introduction.....	3
2.	Methodology.....	4
2.1.	Training the semantic model.....	4
2.2.	Task algorithm.....	5
3.	Evaluation.....	7
3.1.	Evaluating metrics.....	7
3.2.	Results of our submission.....	8
3.3.	An output analysis.....	9
4.	Conclusion.....	12
5.	References.....	13

1. Introduction

In order to make improvements in question-answering applications (Prager et al. 2008; Yahya et al. 2013) and textual entailment or semantic search systems (Hoffart et al. 2014; Roller and Erk 2016), efforts have been made in the past two decades, to investigate and improve the identification of semantic connections, and hypernyms of course, is an essential field.

Different from traditional tasks in which word pairs were given to find out whether exists hypernym relationship between them, the present task, inspired by Espinosa-Anke et al. (2016), is designed to find and extract the suitable hypernym or hypernyms for an input term from a target corpus and to obtain a ranked list of candidate hypernyms, as shown in the table below:

	Input term	Hypernym(s)
1A: English	sorrow	sadness, unhappiness
1C: Spanish	guacamole	salsa para mojar, salsa, alimento

Table 1: Examples of input terms and expected outputs

A valid input term could be a word like ‘sorrow’, ‘guacamole’ or multi-word expressions like ‘City of Whitehorse’, ‘asesino en serie’. Five different subtasks covering three languages and two specific domains of knowledge in English (Medical and Music) was proposed. In this paper we are focusing on the sub-task of General Purpose Hypernym Discovery as well as comparing between English (subtask 1A) and Spanish (subtask 1C).

The source corpus for the English is the 3 billion word UMBC corpus, which contains paragraphs extracted from the web as part of the Stanford WebBase Project. And the source corpus for the Spanish is 1.8-billion-word Spanish corpus. About input terms: there’re 3000 terms for English subtask and 2000 for the Spanish one. For each dataset, training and test sets are evenly split. In order to show how systems tend to perform differently on two kinds of hyponymy relation, also to reduce lexical ambiguity, every input term is labeled by hand as either a concept or an entity. For example, the term ‘apple’ is labeled as concept when it refers to a fruit, it’s labeled as entity when it refers to a company.

Each word is given a gold hypernyms, which are extracted from multiple resources and manually validated.

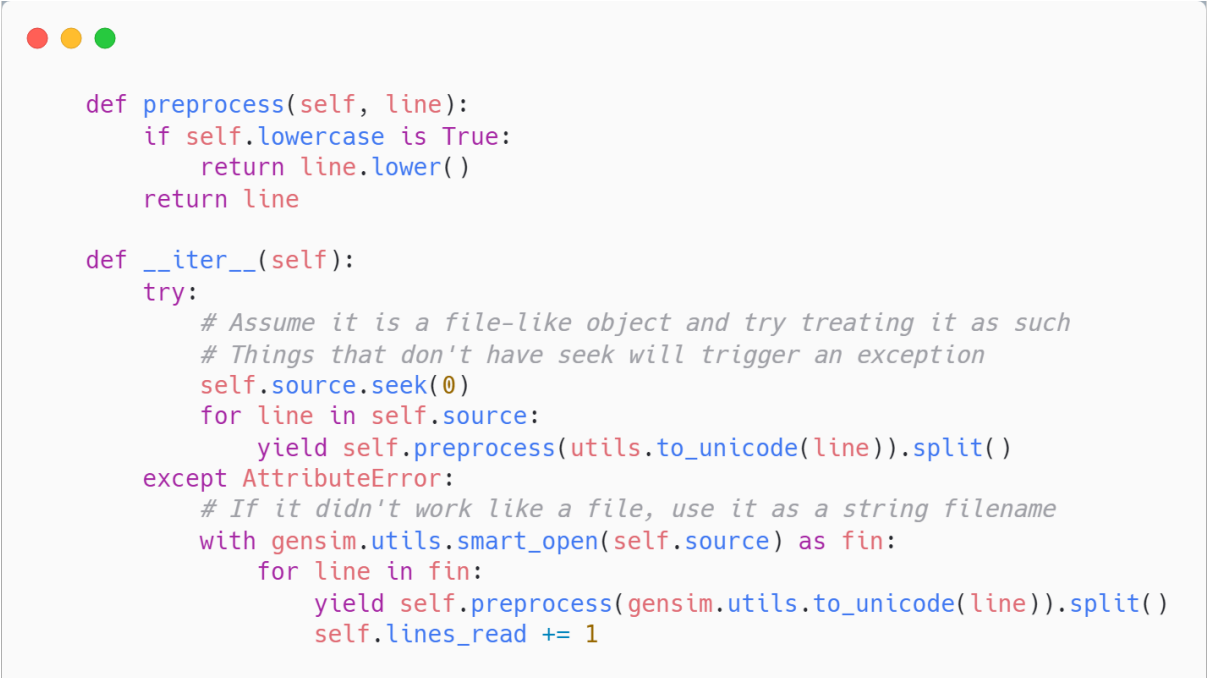
Apart from comparing performance of English subtask and Spanish subtask. Once obtained output of lists of candidate hypernyms, a qualitative analysis is carried out to find which kind of input terms get better results: concepts or entities, words or expressions, abstract terms or concrete terms?

2. Methodology

2.1 *Training the semantic model*

For this project, as explained before, we had two large corpus, one for the English task and another one for the Spanish task. With these corpus we could do some preprocessing of the data in order to improve our chances of getting better results. At the end the only preprocessing that has been done was to lowercase the words contained in the corpus. These have given us the results that will be explained further below.

In the picture below there is a snippet of the preprocessing part of the code.



```
def preprocess(self, line):
    if self.lowercase is True:
        return line.lower()
    return line

def __iter__(self):
    try:
        # Assume it is a file-like object and try treating it as such
        # Things that don't have seek will trigger an exception
        self.source.seek(0)
        for line in self.source:
            yield self.preprocess(utils.to_unicode(line)).split()
    except AttributeError:
        # If it didn't work like a file, use it as a string filename
        with gensim.utils.smart_open(self.source) as fin:
            for line in fin:
                yield self.preprocess(gensim.utils.to_unicode(line)).split()
            self.lines_read += 1
```

However, there is another preprocessing task that could have taken place but we didn't consider at the beginning, which was to write an underscore between words in a phrase. The only reason we didn't add this feature was due to the huge amount of time it took every model to train (22 hours for the English model and 13 hours for the Spanish one) and we did not have that much time left to try more preprocessing tasks. However, we are sure that this would have improved a lot our results.

2.2 Task algorithm

In order to meet the mission objectives, we chose to use the K-nearest-neighbor algorithm. The reason we chose this approach was that our data is already labeled so we can try a supervised learning algorithm and also because of the simplicity of it.

The algorithm works so that it calculates the distance of a new data point to all other training data points and selects the K-nearest data points. Finally, It assigns the data point to the class to which the majority of the K data points belong.

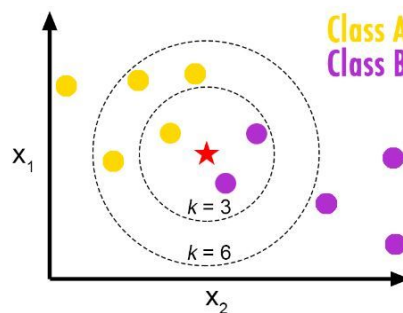


Figure 1: knn illustration

Advantages

Easy implementation, fast algorithm that requires no training prior to making real time predictions. Moreover, new data can be added seamlessly.

Disadvantages

The algorithm finds it difficult to calculate distances in high dimensions and has a high prediction cost for large datasets because in large datasets the cost of calculating distance between new point and each existing point becomes higher. Moreover, it does not work well with categorical features.

On the basis of the KNN algorithm, we adapted it to our problem. As we have observed on class and also as Qiu et al. (2018 June) suggested, hyponyms and hypernyms do not tend to have similar vectors. However, all hyponyms of a specific hypernym do tend to be close between each other. For this reason, we are going to apply KNN to find the K nearest hyponyms to a target hyponym. From these new hyponyms, we are going to obtain their hypernyms and assign them to the target hypernym sorted by the cosine similarity between the target hyponym and the near hyponym. A pseudo-code of the algorithm we are proposing looks like this:

For target-hyponym in test:

- Look for K nearest hyponyms in training.
- For hypo in hyponyms:**
 - Get hypernyms of hypo.
- Sort hypernyms by similarity between the target-hyponym and the hypernym's hyponym.
- Get Top N.

Hence, find below our final code for processing the most important part of our algorithm, where we apply the K-nearest neighbours approach.

```
# Retrieves a string of sorted hypernyms for "hyponym"
def get_hypernyms_by_hypo(sem_model, k, hyponym, hypo_hyper, hypo_vectors):

    if hyponym in sem_model:
        # This is how we get the K nearest hyponyms of "hyponym"
        most_similar = get_most_similar(sem_model, hyponym, k, hypo_vectors)
        top_hypernyms = {}
        similarity = {}

        for (hypo, sim) in most_similar:
            hypernyms = hypo_hyper[hypo] # Getting the list of hypernyms of every hypo.

            for hyper in hypernyms:
                if hyper in top_hypernyms.keys():
                    prev_value = top_hypernyms[hyper]
                    if sim > prev_value:
                        top_hypernyms[hyper] = sim
                        similarity[hyper] = sim
                else:
                    top_hypernyms[hyper] = sim
                    similarity[hyper] = sim

        res = sorted(top_hypernyms, key = lambda key: top_hypernyms[key], reverse = True)
        return res[:15], similarity # We just take 15 hypernyms

    else:
        return [], {}
```

As we used the K-nearest neighbours algorithm, we had to find the best K value to work with in this project. We tried with large and small sizes of K and we came to the conclusion that there was not much need to use larger K since we are taking all hypernyms of every K nearest hyponym. All this sum of hypernyms was usually higher than 15, which is how many we needed. Therefore, we decided to stay with $K = 10$.

3. Evaluation

3.1. Evaluating metrics

The evaluation was carried out with automatic Python code file called ‘scorer.py’. The file measured participants by three parameters: MRR, Precision@k, MAP@k.

$$\mathbf{MRR} = \mathbf{Mean\ Reciprocal\ Rank} = \mathbf{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\mathbf{rank}_i}.$$

This statistical function returns ordered by probability correctness, a list of possible responses to a sample of queries, in our case hypernyms and hyponyms.

$$\mathbf{Reciprocal\ Rank\ (RR)\ score} = \frac{1}{k}$$

The MRR is the mean RR across multiple queries

Precision@k = The number of relevant results among the top K documents. It has a shortcoming that it does not take into account the position of those relevant documents, so they evaluate our code with Precision@1,3,5,15.

$$\mathbf{MAP@k} = \mathbf{Mean\ Average\ MAP} = \frac{\sum_{q=1}^Q \mathbf{AveP}(q)}{Q} \quad \mathbf{Precision@K} =$$

A complementary metric to MRR that takes into account whether hypernyms were retrieved within the k first positions in all Precision@k measures ($1 \leq k \leq 15$)

3.2. Results of our submission

According to *Information Retrieval* course of Stanford University, we assume that highly relevant documents are more useful than marginally relevant documents. The lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined.

Table 2 shows the results evaluated on the test data. The MRR Parameter for Spanish is larger than the English, while the MAP Parameter for English and Spanish are quite close. When checking P@K Parameter we notice: for English, the value increases moderately from P@1 (0.0613) to P@5 (0.075) while increases dramatically when measuring P@15 (0.1264). For Spanish, the value decreases from P@1 (0.0780) to P@3 (0.0713), then increases slightly for P@5 (0.076). However, the value increases dramatically in P@15 researching 0.1205, while happens to English in similar way.

Language	MRR	MAP	P@1	P@3	P@5	P@15
English	11.39	9.33	6.13	6.70	7.58	12.64
Spanish	12.27	9.31	7.80	7.13	7.62	12.05

Table 2: Results on the test data for our submissions (%)

3.3. *An output analysis*

According to the results, we can see our method showed a better performance in Spanish than in English. As Qiu et al. (2018 June) suggested, it was caused by the properties of provided data: the hypernyms in the test set are similar to those in the training set for Spanish, but dissimilar for English. When we take a deeper look at the hypernyms output, we notice that our method can only discover the hypernyms that have occurred in the training set. In fact, it extracts hypernyms for a hyponym by finding its close hyponyms in the training set and building up its list of hypernyms with others' hypernyms. This might account for those abundant yet not really relative words in the lists of hypernyms. There are actually for the English case, 1323 hypernyms out of 3076 (43 %) that appear in the testing dataset that do not appear in the training one. For the case of Spanish, there are 907 hypernyms out of 2143 (42 %) that appear in the testing dataset that do not appear in the training one.

Our model did much better in finding hypernyms for concepts than for entities, since it failed to find hypernyms for any of the entities. We assume it might be because our model is not capable to find similarities with other entities, thus it can not extract hypernyms for them. Similar failure happened when the input is an expression, the model failed to assign them hypernyms. The reason this is happening is because, as explained in the methodology section, we didn't preprocess our corpus data enough, we should have underscored the phrases in order to work with the average of the similarity vectors of the words contained in those phrases. Hypernyms extracted for concrete terms turned out to be more appropriate than those for abstract terms due to the inherent difficulty to assign hypernyms to abstract nouns.

Here are some examples of pretty good predictions:

- **For English:** For the hyponym *warship*.

Gold:

military vehicle passenger transport naval ship traveling
 ship transport transport means of transportation watercraft
 travel land vehicle mechanism travelling movement

Prediction:

transportation passenger transport sailing ship ship type mode of
 transport ship transport means of transport sailboat
 boat sailing boat means of transportation mechanical assembly
 travel move

- **For Spanish:** for the hyponym *hidroavión*.

Gold:

aeroplano avión lancha motora motonave lancha de motor
 barco bote embarcación aeronave vehículo nave transporte
 naval transporte fluvial máquina medio de transporte transporte
 transporte de pasajeros

Prediction:

hirundinidae golondrina aeronave nave vehículo medio de
 transporte transporte transporte de pasajeros medio máquina
 transporte naval transporte fluvial marina mercante
 transporte por camión transporte terrestre

As we can see, for both examples there are some hypernyms that are right but still about half of them are mistakenly assigned to that hyponym. Which means that there is still room for improvement. Some of these improvements could be done by just applying some of the features explained before.

And here, we can find some examples of really bad predictions:

- For **English**: for the hyponym incorporation

Gold:

polity compounding

Prediction:

mathematical operation thought process problem solving
 mathematical process announcement declaration annunciation
 proclamation person convenience evolution course of instruction
 course of study unicameral legislature national assembly

There's not even one single correct hypernym prediction in the case of the English example.

- For **Spanish**: for the hyponym *maní*

Gold:

legumbre planta fruto semilla

Prediction:

remedio terapéutico curativo cura remisión medicina
 fármaco medicamento sustancia fruto duende monstruo
 animal mítico animal legendario criatura mitológica

One more observation in these cases is that we should have paid more attention to the similarity value, which we are using to sort our hypernyms. We could have sorted the hypernyms in a different way, for example by creating a function that generates a value between a hyponym and the predicted hypernym based on a weighted measure that depends on both the cosine similarity between hyponym and hypernym and the now used cosine similarity between the target hyponym and the found similar-hyponym that provided the corresponding hypernyms. In this case we could have made some experiments in order to find the best balance between those weights.

Another mistake that could have been fixed is that we provide for every hyponym, 15 hypernoms (if we can find them) instead of just providing the best found hypernoms. For this scenario we could have tried with different thresholds and just show the ones that are above that value.

4. Conclusion

In this paper, we present our project for the subtask 1A and subtask 1C of the Hypernym Discovery task in SemEval 2018. We turned to the K-nearest approach and obtained an output of a better performance in the Spanish subtask than the English one, which revealed a main drawback of our method: it can not discover the hypernoms that have never occurred in the training set. This shortcoming makes us realize that our model might be improved by combining syntactic patterns to extract hyponym-hypernym. During the process, a factor that held us back from running the model multiple times with changes is the excessive model running time: 22 hours for English, 13 hours for Spanish.

5. References

- Camacho-Collados, J., Delli Bovi, C., Espinosa-Anke, L., Oramas, S., Pasini, T., Santus, E., ... & Saggion, H. (2018). SemEval-2018 task 9: Hypernym discovery. In Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018); 2018 Jun 5-6; New Orleans, LA. Stroudsburg (PA): ACL; 2018. p. 712–24.. ACL (Association for Computational Linguistics).
- Qiu, W., Chen, M., Li, L., & Si, L. (2018, June). NLP_HZ at SemEval-2018 Task 9: a Nearest Neighbor Approach. In Proceedings of The 12th International Workshop on Semantic Evaluation (pp. 909-913).
- Prager, J., Chu-Carroll, J., Brown, E. W., & Czuba, K. (2008). Question answering by predictive annotation. In Advances in Open Domain Question Answering (pp. 307-347). Springer, Dordrecht.
- Yahya, M., Berberich, K., Elbassuoni, S., & Weikum, G. (2013, October). Robust question answering over the web of linked data. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management (pp. 1107-1116). ACM.
- Hoffart, J., Milchevski, D., & Weikum, G. (2014, July). STICS: searching with strings, things, and cats. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval (pp. 1247-1248). ACM.
- Roller, S., & Erk, K. (2016). Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. arXiv preprint arXiv:1605.05433.
- Espinosa-Anke, L., Camacho-Collados, J., Delli Bovi, C., & Saggion, H. (2016). Supervised distributional hypernym discovery via domain adaptation. In Conference on Empirical Methods in Natural Language Processing; 2016 Nov 1-5; Austin, TX. Red Hook (NY): ACL; 2016. p. 424-35.. ACL (Association for Computational Linguistics)

[Information Retrieval course of Stanford University:](#)

<https://web.stanford.edu/class/cs276/handouts/EvaluationNew-handout-1-per.pdf>