Concurrent Learning of Semantic Relations

Mariann Burk
Affiliation / University of Zurich
mariann.burk@uzh.ch

Neeraj Kumar
Affiliation / University of Zurich
neeraj.kumar@uzh.ch

Abstract

Identifying semantic relation between words is of crucial importance and has been a long-term pursuit in the field of natural language processing (NLP). Understanding semantic relations like (Hypernymy, Co-Hypernymy, Synonymy) requires higher level understanding of language like context knowledge, common sense reasoning and inference, and not just some sort of vector and semantics things where cats and dogs are similar to each other in some latent space. In this paper, we explore the use of multitask deep neural networks to evaluate whether concurrent learning of two cognitively linked semantic relations can benefit from each other. In our experiments, we try Logistic Regression, CNN, Multitask MLP Neural Network, and Multitask CNN Network. Within this multi-task context, we also examined the benefits of self-learning where the training of a prediction function is performed over few labelled data jointly with many unlabelled ones.

1 Introduction

Automatic detection of semantic relations is one of the fundamental tasks for natural language processing (NLP). Discovering and precise identification of semantic relations is useful for numerous NLP applications like paraphrasing, query expansion, recognizing textual entailment, ontology building, metaphor detection etc. For example, consider a sentence containing a metaphor, like "I am drowning in a sea of grief". It may be extremely difficult for a NLP system to interpret this sentence if we go by the literal meaning. But if we replace 'drowning' by its co-hyponym 'overwhelmed' and 'sea' by its co-hyponym 'lot', it immediately provides an inference. It may be noted that, 'sea' and 'lot' are (co-)hyponyms for the concept 'large indefinite amount' whereas 'drown' and 'overwhelm' are (co-)hyponyms for the concept 'cover'.

Sematic relations like hyponyms, hypernyms, co-hyponyms, meronyms etc. can be classified into symmetric (co-hyponymy) and asymmetric (hypernymy, meronymy). Researchers have proposed various methodologies that either focus on a single lexical relation (e.g. hypernymy vs. random) exclusively or learn specific models capable of identifying multiple semantic relations (e.g. hypernymy vs. synonymy vs. random). First can be defined as to identify whether a given relation r holds between a pair of words (x,y) or not (i.e. two-class problem), whereas 2nd can be modelled as deciding which semantic relation ri (if any) holds between a pair of words (x; y) (i.e. multi-class problem).

This paper is based on the paper written by (Balikas et al) and uses an approach that applies multitask learning paradigm to the problem of identifying semantic relationships. We define a multitask learning framework using a hard parameter sharing deep neural network that takes as input a learning word pair (x; y) encoded as the concatenation of both word embeddings. The intuition behind our experiments is that on one hand multi-task learning profits from a regularization effect via alleviating overfitting to a specific task, thus making the learned representations universal across tasks whereas on the other hand if the tasks are correlated, the neural network should improve its generalization ability by considering the shared information.

In addition, we looked at a self-learning strategy to increase the supervised data set by using most confident predictions of the model(We iteratively take 10 unlabelled word pairs for which semantic relationship is identified with highest probability by the model, and add them to the labelled set) .

2 Related Work

The idea of representing words as vectors has been studied for about three decades. The public availability of word embedding training programs such as word2vec and GloVe allowed researchers to create models with different parameters and dimensionality sizes for different purposes. Although it is comparatively trivial to predict 'queen' as the answer to this query x = king man+woman, we cannot expect 'bike' as the answer to the query x = shoeboot + scooter with the same level of confidence if the relationship is expected to be either synonymy, antonymy, hyponymy, or hypernymy.

Researchers have identified two main approaches to capture the semantic links between two words pattern-based and distributional.Pattern Based has focused on creating a large data set of hypernymy pairs. To tell if the hypernymy relationship holds for a pair (X, Y), it simply checks if (X, Y) is in the data set. To create the dataset, it relies on information extraction over large corpra using some syntactic patterns that indicate the hypernymy relationship. This method is simple and efficient, but generally it has low precision and low recall because information extraction is error prone and the text corpus is always sparse.

Another approach for hypernymy identification is based on the Distributional Inclusion Hypothesis which assumes that hypernyms have broader contexts than hyponyms. The DIH states that if X is a hypernym of Y, then most context features of Y are also features of X. For example, if X is animal and Y is cat, most features of cat are also features of animal, but at least some features of animal do not apply to cat. For instance, the term rights co-occurrs with animal frequently, but not so much for cat. A criticism of these methods comes from the claims that show that supervised methods tend to learn the existence of prototypical hypernyms rather than the actual relations between the two terms. Firstly, one has to consider that the inclusion hypothesis is not always correct, that is, hypernyms do not always have broader context features. For example, American is a hypernym of Obama, but Obama have some features that are not strongly associated with American.

More recently, attention has been focusing on identifying semantic relations using neural language embeddings, as such semantic spaces encode linguistic regularities. In particular, CNN model has been applied in many multitask learning papers. For example (Zeng et al., 2014) use a convolutional deep neural network to extract lexical features learned from word embeddings and then fed into a softmax classifier to predict the relationship between words.

3 Data sets

To perform our experiments on synonyms, hypernyms and co-hyponyms, we are using two different datasets. Firstly, Rumen dataset, containing 18978 word pairs gathered from WordNet and equally organised amongst three classes – hypernymy, synonymy and random. Secondly, Roots dataset containing 9600 word pairs, randomly extracted from 3 different datasets, distributed between hypernymy, co-hyponymy and random relations. The word pairs emeddings are initialized using the 300 dimensional representations of GloVe. The datasets are split into test, train and validation subsets. In order to avoid lexical memorization and keep the model from overfitting, we follow the approaches from (Levy et al., 2015)(Levy et al., 2015) and (Balikas et al., 2018)(Balikas et al.,2018) and perform a lexical split –the split of the test and train sets such that each contains a distinct vocabulary. For our model, lexical repetition exists in the train, validation and unlabelled subsets, but the set is exclusive in terms of vocabulary.

4 Methodology

4.1 Multi-task Learning with Hard Parameter Sharing

In our experiments, we examined the use of a multi-task learning algorithm that relies on hard parameter Sharing. The intuition is that the shared parameters of the related tasks, for example word representations or weights of some hidden layers, can benefit the performance of all tasks learned concurrently. The NN architecture is illustrated in figure-1 (Balikas et al., 2018)

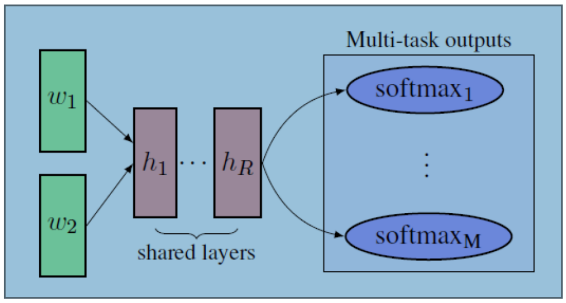

Figure 1: The Multitask Learning Architecture is defined on top of a feed forward neural network, where the layers h1. . . .hr are shared across tasks, while the output layers softmax1. . . softmaxM are task-dependent

The input of the network is the concatenation of the word embeddings of the word pairs followed by a series of non-linear hidden layers. Here, a softmax layer corresponds to a task, and concurrently learning M tasks requires M separate output softmax layers. As opposed to the soft parameter sharing, where each task has its own model with its own parameters, the hard parameter sharing, going back to (Caruana 1993), is efficient because the first layers that are shared, are tuned by back-propagating the classification errors of every task.This is where the multitasking happens when the architecture uses the datasets of all tasks, not just one at a time. The positive side of using this technique is that it reduces the risk of overfitting. In fact, (Baxter, 1997) showed that the risk of overfitting the shared parameters is an order N – where N is the number of tasks – smaller than overfitting the task-specific parameters, for example output layers. Figure 2 shows the details of the training protocol, we are applying the same algorithm as (Balikas et al.,2018).

Algorithm 1: Multi-task Training Process

Data: Labeled words pairs $\mathcal{L}^i$ for each of the M tasks, batch size b, epochs
epoch = 1 ;
while *epoch* $<$ *epochs* **do**
  for $i = 0;\ i < M;\ i = i + 1$ **do**
    Randomly select a batch of size b for task i ;
    Update the parameters of the neural network architecture according to the errors observed for the batch;
    Calculate the performance on the validation set of task i.
  end
end

Figure 2: The Multitask Learning Process

4.2 Semi-Supervision via Self-Learning

Semi-supervised learning approaches perform well in a variety of tasks such as text classification and text summarization. As in the supervised learning framework, we assume that we are given access to a set L that consists of K pairs of words labeled according to the relationship rel. Complementary to that, we also assume to have access to a set of K0 words pairs U distinct from those of L, and totally unlabeled. The challenge in this setting is to surpass the performance of classification models trained exclusively on L by using the available data in U. (Balikas et al., 2018)(Balikas et al). Following the algorithm provided by (Balikas et al., 2018), we are training a classifier C using word pairs L, and progressively expanding L, by pseudo-labeling 10 pairs within U, for which the current prediction function is the most confident and adding them to L.

Algorithm 2: Self-learning

Data: Word pairs: labeled $\mathcal{L}$, unlabeled $\mathcal{U}$, validation $\mathcal{V}$; integer N
$\mathcal{L}_0 = \mathcal{L}, \mathcal{U}_0 = \mathcal{U}$;
Train classifier C using $\mathcal{L}_0$;
V_0 : Performance of C on $\mathcal{V}$;
Set $t = 0$;
while *Size*$(\mathcal{U}_t) > 0$ *and* $V_t \geqslant V_0$ **do**
  Get probability scores p of C on $\mathcal{U}_t$;
  pseudo_labeled$(N) = \arg\max(p)$, **stratified wrt** $\mathcal{L}_0$;
  t = t + 1;
  $\mathcal{L}_t = \mathcal{L}_{t-1}$ + pseudo_labeled ;
  $\mathcal{U}_t = \mathcal{U}_{t-1}$ − pseudo_labeled;
  Retrain C using $\mathcal{L}_t$;
  V_t : Performance of C on $\mathcal{V}$;
end

Figure 3: Algorithm 2

5 Results

We have specified two tasks to compare our models namely coord-random task and hyper-random task. The earlier referring to a synonym detection and the latter a hypernym detection task.

In the first experiment, we studied the impact of concurrent learning of co-hyponymy (Red,Blue) and hypernymy (Color, Red). Results suggests that combination of CNN, multitasking and self-learning provides the best performance for the hypernymy relation. In this case, a 1.1% improvement is obtained over the best baseline (CNN) for hypernymy classification, indeed suggesting that there exists a learning link between hypernymy and cohyponymy.However, the results for

Table 1: Root Data Set Performance, First 2 columns correspond to Co-hypo. Vs Random Relation while last two belongs to Hypernymy Vs Random relation classification

Model	Acc	MaF1	Acc	MaF1
LR	0.912	0.879	0.822	0.776
NN Baseline	0.906	0.871	0.842	0.805
CNN Baseline	0.934	0.908	0.851	0.82
NN + SL	0.92	0.889	0.843	0.806
NN + MT	0.916	0.882	0.830	0.786
NN +MT + SL	0.917	0.885	0.830	0.782
CNN + SL	0.93	0.905	0.824	0.786
CNN + MT	0.919	0.890	0.850	0.815
CNN +MT + SL	0.907	0.862	0.856	0.821

Table 2: Rumen Data Set Performance, First 2 columns correspond to Synonymy Vs Random Relation while last two belongs to Hypernymy Vs Random relation classification

Model	Acc	MaF1	Acc	MaF1
LR	0.627	0.627	0.718	0.716
NN Baseline	0.687	0.687	0.743	0.74
CNN Baseline	0.73	0.73	0.753	0.744
NN + SL	0.691	0.691	0.754	0.75
NN + MT	0.693	0.693	0.757	0.753
NN +MT + SL	0.699	0.699	0.776	0.772
CNN + SL	0.731	0.731	0.744	0.737
CNN + MT	0.739	0.739	0.772	0.768
CNN +MT + SL	0.734	0.734	0.764	0.764

co-hyponymy classification can not compete with a classical supervised strategy using CNN. So, we can expect an improvement for hypernymy classification but not for co-hyponymy in a multi-task environment, suggesting a positive influence of co-hyponymy learning towards hypernymy but not the opposite.

In the second experiment, we examined the impact of concurrent learning of synonymy and hypernymy with the same models, used in the first experiments. For hypernymy classification, best configuration is combination of MLP NN, with multitask and self learning, where as for the synonymy task, best performance is achieved by combination of CNN, multitask and self learning model. The overall improvement for hypernymy prediction is 2.2% over the best base line(CNN), whereas for the synonymy task, improvement over the best base line(CNN) is 0.4%. So, these results tend to suggest that hypernymy identification may greatly be impacted by the concurrent learning of synonymy and vice versa (although to a less extent).

Overall, the results of RUMEN dataset are generally lower than that of the ROOT9 dataset, mainly because of the complexity of the datasets.

6 Conclusion

In the current paper, we examined the concurrent learning of semantic relations (co-hyponymy, hypernymy and synonymy) by using self-learning and multitask learning with hard parameter sharing. Our results show that concurrent learning can lead to improvements, thereby justifying our initial hypothesis.We saw that performance gain with multitask learning is much higher for simpler architectures like feed forward neural network than more complex architecture such as convolutional neural networks. Our results show that hypernymy classification can gain from concurrent learning of co-hyponymy. Additionally, synonymy and hypernymy classification gains from concurrent learning of each other. We also found out that adding the more complex CNN architecture improves the learning performance for similar tasks.

References

Georgios Balikas, Gaël Dias, Rumen Moraliyski, and Massih-Reza Amini. 2018. Concurrent learning of semantic relations. *CoRR*, abs/1807.10076.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.