Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5838–5845
Marseille, 11–16 May 2020

Recognizing Semantic Relations by Combining Transformers and Fully Connected Models

Dmitri Roussinov, Serge Sharoff, Nadezhda Puchnina
University of Strathclyde, University of Leeds, University of Tallinn
dmitri.roussinov@strath.ac.uk, s.sharoff@leeds.ac.uk, np486061@tlu.ee

Abstract

Automatically recognizing an existing semantic relation (e.g. "is a", "part of", "property of", "opposite of" etc.) between two words (phrases, concepts, etc.) is an important task affecting many NLP applications and has been subject of extensive experimentation and modeling. Current approaches to automatically telling if a relation exists between two given concepts X and Y can be grouped into two types: 1) those modeling word-paths connecting X and Y in text and 2) those modeling distributional properties of X and Y separately, not necessary in the proximity to each other. Here, we investigate how both types can be improved and combined. We suggest a distributional approach that is based on an attention-based transformer. We have also developed a novel word path model that combines useful properties of a convolutional network with a fully connected language model. While our transformer-based approach works better, both our models significantly outperform the state-of-the-art within their classes of approaches. We also demonstrate that combining the two approaches results in additional gains since they use somewhat different data sources.

1. Introduction

During the last few years, Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) have resulted in major breakthroughs and are behind the current state-of-the-art algorithms in language processing, computer vision, and speech recognition (LeCun et al., 2015). Meanwhile, modeling higher level abstract knowledge still remains a challenging problem even for them. This includes classification of semantic relations: given a pair of concepts (words or word sequences) identify the best semantic label to describe their relationship. The possible labels are "is a", "part-of", "property-of", "made-of", etc. This information is useful in many applications. For example, knowing that *London* is a *city* can help a Question Answering system answer the question *What cities does the River Thames go through?* Information retrieval benefits from query expansion with more specific words, e.g. *transportation disasters* $\rightarrow$ *railroad disasters*. For the task of database federation, an attribute in one database (e.g. with values *France, Germany*, and *UK*) often needs to be automatically matched with an attribute called *country* in another database. The connection between word meanings and their usage is prominent in the theories of human cognition (Mikoajczak-Matyja, 2015) and human language acquisition (Bybee and Beckner, 2015). While manually curated dictionaries exist, they are often out-of-date, not covering specialized domains, designed to be used by people, and exist for only a few well resourced languages (English, German, etc.). Therefore, here we are interested in methods for automated discovery (knowledge acquisition, taxonomy mining, etc.) .

The automated approaches to detecting semantic relations between concepts (words or phrases) can be divided into two major groups: 1) path-based and 2) distributional methods. *Path-based approaches* (e.g. Shwartz et al. (2016)) essentially look for certain patterns in the joint occurrences of words (phrases, concepts, etc.) in the corpus. Thus, every word pair of interest *(x,y)* is represented by the set of *word paths* that connect *x* and *y* in a raw text corpus (e.g. Wikipedia). *Distributional approaches* (e.g. Wang et al. (2019)) are based on modeling the occurrences of each word, *x* or *y*, separately, not necessary in the proximity to each other. Our goal here is to *improve, compare and combine those two classes of approaches*.

Attention-based transformers (e.g. Vaswani et al. (2017)) have been recently shown more effective than convolutional and recurrent neural models for several natural text applications, leading to new state-of-the-art results on several benchmarks including GLUE, MultiNLI, and SQuAD (Devlin et al., 2018; Lample and Conneau, 2019). At the same time, we are not aware of any applications of attention-based transformers to the task of recognizing semantic relations between words. Our contributions are as follows: 1) We develop a novel path-based model that combines useful properties of convolutional and fully connected networks. Our approach resolves several shortcomings of the prior models within that type. As a result, it outperforms the state-of-the art path-based approaches. 2) We suggest a distributional approach that is based on an attention-based transformer, and show that it exceeds the state-of-the art performance on several standard datasets. 3) While our distributional approach worked significantly better than our neural path-based model, the combination of our two approaches demonstrates additional gains, since the two approaches use somewhat different data sources. 5) We illustrate that even our best transformer model still has certain limitations which are not always revealed by the standard datasets.

We make our code and data publicly available. The next section overviews the prior related work. It is followed by the description of the models, followed by our empirical results.

2. Prior Work

While earlier *path-based* approaches used small sets of manually crafted templates to detect patterns (Hearst, 1992; Snow et al., 2004), later works successfully involved trainable templates (Nakashole et al., 2012; Riedel et al., 2013). Successful models using trainable *distributional representation* of words (their *embedding* vectors) (Mikolov et al., 2013; Pennington et al., 2014) were developed and for some time surpassed the path-based methods in performance (Santus et al., 2016; Necsulescu et al., 2015). Levy et al. (2015) noted that supervised distributional methods tend to perform lexical memorization: instead of learning a relation between the two terms, they learn an independent property of a single term in the pair. For example, if the training set contains pairs such as *(dog, animal), (cat, animal)*, and *(cow, animal)*, the algorithm learns to classify any new *(x, animal)* pair as true, regardless of *x*.

Shwartz et al. (2016) successfully combined both distributional and path-based approaches into a single model that uses a recurrent neural network (RNN) and exceeded the best results at the time for the hypernymy detection ("is-a" relation). Their model was later extended to multiple relations in Shwartz and Dagan (2016) and became the state-of-the-art after winning a competition-format workshop on recognizing semantic relations. We include a mathematical description of their approach here since 1) We are using it as one of our baselines and 2) in the immediately following section, we elaborate how we overcome its shortcomings. Their approach (HypeNet, later called Lexnet) proceeds as following. The data consists of the targeted pairs of words along with their relationship label and all the word paths connecting the target pairs in the corpus. Each path consists of edges (words). Each edge is represented by a word embedding vector. Thus, each word path p is represented as a sequence of edge vectors $\{\overrightarrow{v_{e1}}, \overrightarrow{v_{e2}}, \overrightarrow{v_{e3}}, ...\} = \{\overrightarrow{v_{et}}(x,y)\}$ for $t = 1, ..., l_p$, where l_p is the path length. This sequence is mapped by an RNN into a *context vector* $\overrightarrow{v_p}(x,y)$ defined for each path:

$$\overrightarrow{v_p}(x,y) = \mathrm{RNN}(\overrightarrow{\{v_{et}(x,y)\}}) \quad (1)$$

The context vector for the pair (x,y) is defined as the *average* context vector for its paths:

$$\overrightarrow{v}_{xy} = \frac{\sum_p \overrightarrow{v_p}(x,y)}{\#Paths(x,y)} \quad (2)$$

where $\#Paths(x,y)$ is the number of word paths connecting the pair (x,y). This vector, in turn, is used to make a classification decision, with an optional hidden layer.

There have been several related studies following Shwartz et al. (2016): Shwartz et al. (2017) did extensive comparison of supervised vs. unsupervised approaches to detecting "is-a" relation. Washio and Kato (2018) looked at how additional word paths can be predicted even if they are not in the corpus. Roller et al. (2018) also looked at "is-a" relation and confirmed the importance of modeling word paths in addition to purely distributional methods. Still, the models from Shwartz et al. (2016) and Shwartz and Dagan (2016) remain unsurpassed within the class of word-path models. We are using them as one of our baselines, along with the *same datasets and same corpus data for a direct comparison of our models*. Among distributional approaches, Wang et al. (2019) suggested using hyperspherical relation embeddings and improved over the results of Shwartz and Dagan (2016) on 3 out of 4 datasets. We use the model from Wang et al. (2019) as another baseline.

A cognate approach concerns a stacked architecture combining CNN and LSTM, which for example was used in (Yuan and Sharoff, 2020) for classification of translation errors.

3. Combined Models for Semantic Relations

3.1. Path-Based

3.1.1. Informal Description

Since our proposed path-based model does not use a recurrent network, it is simpler to describe and faster to train. It also resolves several shortcomings of the current state-of-the-art model by Shwartz et al. ((Shwartz et al., 2016)) as we explain below. Figure 1 presents an informal intuitive illustration. We jointly train our semantic classification along with an unsupervised language modeling (LM) task which captures the probability distribution over sequences of words in the language. In our experiments here, we used exactly the same word paths as in our baselines (Shwartz et al., 2016) and (Shwartz and Dagan, 2016) since the paths are publicly available. 95% of the paths are no longer than 4 words. This allowed us to simplify our language model even further: instead of currently popular recurrent neural networks, we simply used a fully connected feed-forward neural network with a single hidden layer. We limited the number of input words to 4. The shorter paths are padded with special symbols ("<E>").

The output of LM is the probability of occurrence of any input word sequence. *We use some of those probabilities as features for our relation classification model.* Inspired by the success of convolutional networks (CNNs)(e.g. (Ma et al., 2019; Zeng and Ji, 2015), we use a fixed set of trainable filters (also called *kernels*), which learn to respond highly to certain patterns that are indicative of specific semantic relations. For example, a specific filter f_i can learn to respond highly to *is a* (and similar) patterns. At the same time, our LM may suggest that there is a high probability of occurrence of the sequence *green is a color* in text. Combining those two facts suggests that *green* belongs to the category *color* (true *is-a* relation between them). Figure 1 shows only three such filters (and the probabilities of the sequences P_1, P_2, P_3), while in our current study we used up to 16.

Thus, the LM probabilities act as approximate ("soft") pattern matching scores: 1) similar patterns receive similar scores with the same filter and 2) similar filters produce similar scores for the same pattern. LM also reduces the need for using many filters as explained by the following intuitive example: While training, LM can encounter many examples of sequences like *green is a popular color* and *green is a relaxing color.* By modeling the properties of a language, LM learns that removing an adjective in front of a noun does not normally result in a large drop of the probability of occurrence, so the sequence *green is a color* also scores highly with LM even if it never occurs in text.

Since the current state-of-the art path-based approach (Shwartz et al., 2016) aggregates the word paths connecting each target pair by averaging the context vectors representing all the paths (formula 2), we believe their approach has two specific drawbacks that our approach does not: 1) when averaging is applied, the different occurrences of word patterns are forced to compete against each other, so the more rare occurrences can be dominated by more common ones and their impact on classification decision neglected as a

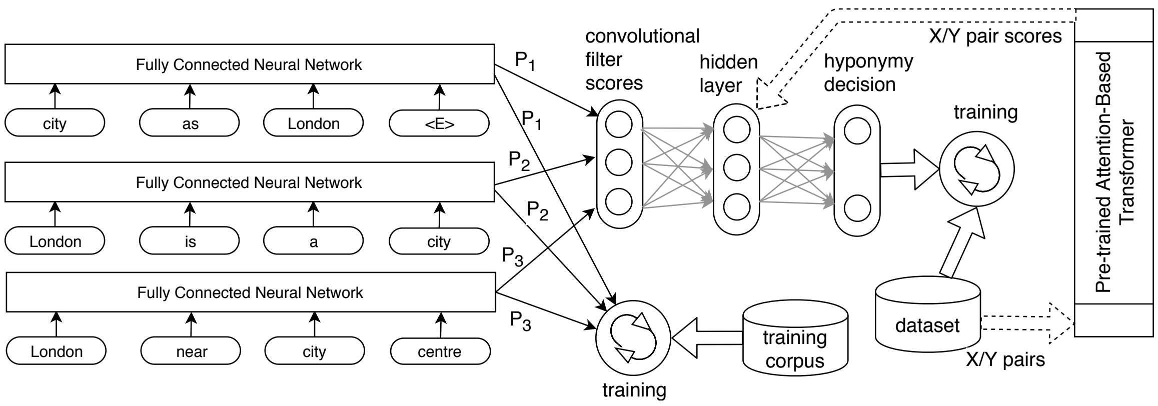
Fully Connected Neural Network
city
as
London
<E>
Fully Connected Neural Network
London
is
a
city
Fully Connected Neural Network
London
near
city
centre
P1
P1
P2
P2
P3
P3
convolutional filter scores
hidden layer
hyponymy decision
X/Y pair scores
training
training
training corpus
dataset
X/Y pairs
Pre-trained Attention-Based Transformer

Figure 1: Our path-based neural approach to semantic relationship classification.

result. By using LM we avoid facing the question how to aggregate the context vectors representing each path existing in the corpus. 2) The other relative strength of our approach over the baseline comes from the fact that our model does not "anonymize" the word paths unlike Shwartz et al. (2016), which uniformly uses "x" and "y" for the path ends regardless of which words the target pair (x,y) actually represents. Without the use of LM, this anonymizing is unavoidable to generalize to the previously unseen (x,y) pairs, but it also misses the opportunity for the model to transfer knowledge from similar words.

3.1.2. Formal Description

Language model (LM) is a probability distribution over sequences of words: $p(w_1, ..., w_m)$. We are using currently common *distributed* word representations: all the words are represented by vectors $v_1, v_2, ..., v_m$ which are trainable paramaters in the model, typically called *word embedding vectors*. For our LM, we use a fully connected feedforward neural network with a single hidden layer and a softmax layer, and limit the number of input words m (word path length) to 4. Thus, the probability of word w_4 to follow a sequence of words $w_1, ..., w_3$ is determined as follows. First, the *hidden layer* vector is defined as:

$$\overrightarrow{h} = \tanh(W \cdot [v_1; v_2; v_3] + b) \tag{3}$$

where v_1, v_2, v_3 are the embedding vectors for the words w_1, w_2, w_3 accordingly[1], [] stands for concatenation, W is a trainable matrix (parameter) and b is a trainable vector (bias). The hidden layer is used as the input to the softmax layer, so the output probability is defined as:

$$p(w_4|w_1, w_2, w_3) = \text{softmax}\,(W_1 \cdot \overrightarrow{h} + b_1)[w4] \tag{4}$$

where W_1 is a trainable matrix, b_1 is a trainable vector (bias), and *softmax* is a standard function to scale any given vector of scores to probabilities. The operator [] here shows that we take the value of the *softmax* function corresponding to the word w_4.

Next, we extend our LM to be defined not only over word vectors, but over any arbitrary 4 vectors: $p_{LM} = p(w_4|v_1, ..., v_3)$. Since we are interested in what word patterns are typically connecting the target pairs of words (x,y), we only make use of conditional probabilities of the form $p(v_y|v_x, v_1, v_2)$, where (x, y) is one of the *target pairs* of words, (v_x, v_y) are their embedding vectors and the vectors v_1, v_2 define a trainable *filter*. Our filters are designed to capture which paths commonly connect a given pair of target words (x, y). While informally v_1, v_2 can be interpreted as defining certain patterns like "is a", they don't have to correspond to any real words in the vocabulary. Formally, they are just model parameters and are trained along with the other parameters by back-propagation. Thus, we define the score of each of our filters the following way:

$$f_i = p(v_y|v_x, v_1^i, v_2^i) \tag{5}$$

We define the vector of filter scores by concatenating the individual scores: $\overrightarrow{f} = [f_1, f_2, f_3, ..f_N]$, where N is the total number of filters (16 in our study here).

Filter scores $\overrightarrow{f}$ are mapped into a relation classification decision by using a neural network with a single hidden layer. Thus, we define:

$$\overrightarrow{h_1} = \tanh(W_2 \cdot \overrightarrow{f} + b_2) \tag{6}$$

where W_2 is a trainable matrix and b_2 is a trainable "bias" vector. The classification decision is made based on the output activations:

$$c = \text{argmax}\,(W_3 \cdot \overrightarrow{h_1} + b_3) \tag{7}$$

where W_3 and b_3 are also trainable parameters. As traditional with neural networks, we train (optimize our model parameters) to minimize the cross-entropy cost:

$$\textit{cost} = -\log((\text{softmax}\,(W_3 \cdot \overrightarrow{h_1} + b_3))[c_l]) \tag{8}$$

where c_l is the correct (expected) class label. Operator [] indicates here that we take the value of the vector returned by the softmax function that corresponds to the correct class label c_l. We used stochastic gradient descent for cost minimization.

3.2. Pre-Trained Attention-Based Transformer

The diagram on Figure 2 illustrates how attention-based transformer (Vaswani et al., 2017) operates. Instead of

[1] We deliberately do not use the arrow over the word vectors to simplify the notation.

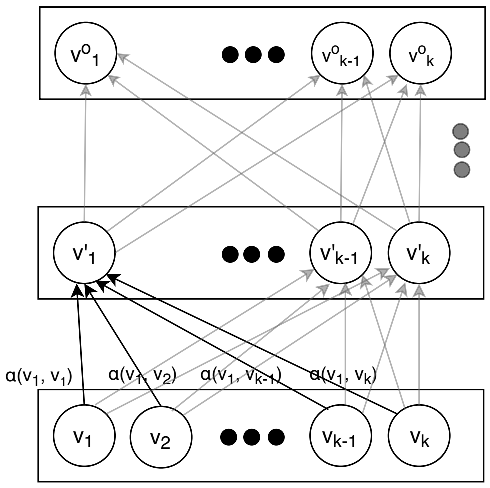

Figure 2: Attention-based transformer used in our distributional approach to semantic relationship classification.

recurrent units with "memory gates" essential for RNN-s, attention-based transformers use additional word positional embeddings which allows them to be more flexible and parallelizable than recurrent mechanisms which have to process a sequence in a certain direction. The conversions from the inputs to the outputs are performed by several layers, which are identical in their architecture, varying only in their trained parameters. In order to obtain the vectors on the layer above, the vectors from the layer immediately below are simply weighted and added together. After that, they are transformed by a standard nonlinearity function. We use *tanh*:

$$\vec{v_i}' = \tanh(W \cdot \sum_{t=1}^{k} \alpha_t \vec{v_t}) \qquad (9)$$

here, $\vec{v_i}'$ is the vector in the i-th position on the upper layer, $\vec{v_t}$ is the vector in the t-th position on the lower layer, W is a trainable matrix (same regardless of i but different at different layers), and α_t is a trainable function of vectors $\vec{v_i}$ and $\vec{v_t}$, such as the weights for all $\vec{v_t}$ add up to 1. We use a scaled dot product of the vectors $\vec{v_i}$ and $\vec{v_t}$:

$$\alpha_t = \vec{v_i} \cdot W' \cdot \vec{v_t} \qquad (10)$$

where W' is a trainable matrix (also same regardless of i and t at the same layer but different at different layers). The normalization to 1 is accomplished by using a *softmax* function.

This mechanism allows rich vector representations to be formed at the highest layers that can capture the entire content of a word sequence (e.g. a sentence or a word pair) so it can be effectively used for any AI applications such as text classification or generation. As it is commonly done with the transformers, we make our output classification decision based on the first vector on the top level. We do not use a hidden layer here, so we apply our formula **??** above to h_1 defined as the following:

$$\vec{h_1} = \vec{v_0^u} \qquad (11)$$

where $\{\vec{v_t^u}\}$ is the vector sequence produced by the transformer for the top level.

3.3. Combining our models

To combine our models, we first used 75% of the training set to train them separately. Then, we froze both models and used the remaining 25% to learn how to combine them. The combination was performed by a single *softmax* decision layer that takes the following two concatenated inputs: 1) the activations at the hidden layer of our path-based model as defined by formula 6 and 2) the classification vector $\vec{v_0^u}$ of our transformer as defined by formula 11. Thus, the combining model acted as a "mixture of experts", while still having access to word properties (embeddings), so it learns to customize combination decisions, e.g. using different preferences between the two models for the pairs of nouns from those for the pairs of verbs.

4. Empirical Evaluation

4.1. Illustration on Synthetic Datasets

In order to gain additional insight into which model works best, we also experimented with a synthetic dataset. Using synthetic data is commonly used to illustrate the limitations of a specific approach. For example, inability to learn a *XOR* function by a single layer neural network illustrates the need for a hidden layer. Thus, we sought to generate as simple and easily replicable text as possible, while still having the properties that are present in real datasets and may present a challenge to a particular model. For the ease of interpretation, we limited the number of semantic relations in the simulation to 2 (true/false): an example of such situation is the problem of verifying membership in a semantic category, e.g. (*color, green, true*), (*coffee, drink, true*) but (*coffee, green, false*). Rather than generating the text and then extracting the paths from it, we directly generate the paths, so they all start/end with x/y where (x, y) is one of our target pairs of words. Without loss of generality, we call y a *category*, and we call x a *candidate*. We generate three non-overlapping types of words:
1) Category labels: {*c1, c2,..., cC*}, where *C* is the total number of categories. 2) Candidates: {*w1,...,wW*}, where *W* is the total number of them. These are the words that may happen to belong (true-pair) or not to belong (false-pair) to the specific categories. 3) Connectors: {*is1, is2, ..., isM*}, where M is the total number. They represent typical word patterns that connect true category-candidate pairs. For example, for the path *water is liquid*, the connector is *is*. For simplicity, all our connectors consist of a single-word, and x always precedes y.
Every category uses only a randomly-assigned subset of m connectors for positive (correct) candidates, which mirrors the real text where, for example, category *color* uses "is a" but does not use "is" , while category *actor* uses "is" but does no use "is a". Each category also uses $m - 1$ connectors for negative pairs. Thus, *the only difference between true and false candidates is that the true candidates occur with larger number of connectors*. Since we wanted to stress our algorithms, we deliberately generated positive and negative pairs that are challenging to tell apart.
We generate all the paths by the following: for each category *c*, each true candidate *x* and each connector *o* we gen-

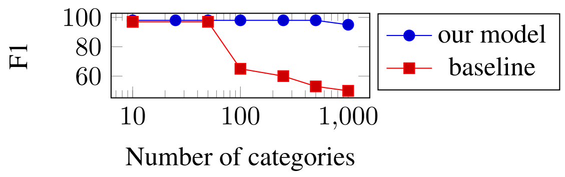

Figure 3: F_1 scores of our model compared to the state-of-the-art baseline on synthetic data.

erate the path *x* + *o* + *c* if *x* is a true instance of *c* and connector *o* is used by *c*. Otherwise, we generate the path *x* + *o* + *w*, where *w* is sampled uniformly from $\{$*w1,...,wW*$\}$. So basically, if the connector is not used by category *c* or if *x* is not a true instance of *c*, then the combinations of *x* and connector *c* can be followed by any arbitrary words. E.g. since *red* is not a *drink*, we will not have paths like *red is drink*, but may have instead paths like *red is color*, *red is stop*, etc.

In our first round of experiments we used $W = 10000, M = 6, m = 4$. We separately generated training and testing subsets of equal size, without overlapping categories. Guided by the sizes of our real datasets listed in Table 1, we tested the numbers of categories on the logarithmic scale: $\{10, 50, 100, 500, 1000\}$. Both the baseline and our model were able to achieve the F_1 score (the metric used by our baseline papers (Shwartz et al., 2016; Shwartz et al., 2017) above 95% on all of them.

In order to further strain both methods and to make our data more realistic, we imposed additional noise by repeating each path *r* number of times, where *r* was randomly selected between 1 and 10. We have also added the paths consisting entirely of "noise" words, which were randomly selected from the $\{n1, ..., nN_n\}$ set. We set $N_n = C$ to avoid introduction of an additional simulation parameter but still reflecting the scale of the generated set.

Figure 3 presents the results for several numbers of categories *C*. We report the maximum score on the test set. It can be seen that for the small number of categories both approaches still work well, but for 100+ categories the state of the art method, which is based on averaging the context vector, starts getting disoriented. One possible explanation may be that it happens because the occurrences of some connectors lose their impact on the classification decision when averaging occurs. Since our approach is based on a language model rather than on averaging, it is not affected by the introduction of additional noise (the filter values remain the same).

We have run the similar experiments with the word embeddings and network sizes in the $\{$*25, 50, 100, 200*$\}$ set. While the specific F_1 scores were different, the overall comparison remained the same: only our approach was able to handle the imposed noise well. Only after we reduced the number of filters to below 6 (the number of unique possible connectors in the simulation), its performance also dropped below 90%.

4.2. The Datasets

Table 1 summarizes general statistics of the datasets. We used the same datasets as our baselines: the first two are from Shwartz et al. (2016) and were built using a similar methodology: the relations used in them have been primarily taken from various sources including WordNet, DBPedia, Wikidata and Yago. Thus, their x-s are primarily named entities (places, films, music albums and groups, people, companies, etc.). The important difference is that in order to create the split between training, testing and validation sets for HypeNet Lexical, the lexical separation procedure was followed (Levy et al., 2015), so that there is no overlap in words (neither x nor y) between them. This reduces "lexical memorization" effect mentioned above. The last four datasets are from Shwartz and Dagan (2016), which originate from various preceding studies: K&H+N (Necsulescu et al., 2015), BLESS (Baroni and Lenci, 2011), ROOT09 (Santus et al., 2016), EVALution (Santus et al., 2015). Most of the relations for them were also taken WordNet. BLESS dataset also contains *event* and *attribute* relations, connecting a concept with a typical activity/property, e.g. *(alligator, swim)* and *(alligator, aquatic)*. EVALution dataset contains the largest number of semantic relations including antonyms, e.g. *(good,bad)*. To make our comparison more direct, we used exactly the same splits into training, development (validation) and testing subsets as in the baselines. We also used exactly the same word paths data, as it is made publicly available by the authors.

4.3. Experimental setups

Since we sought to keep the number of hyper-parameters to the minimum, we set the word embedding size, the RNN context vector size, and the hidden layer size to be the same within all our path-based models. We tested their values in the range of $\{$*50-1000*$\}$. This size is the only hyper-parameter that was varied in our experiments. We used the static learning rate of 0.01. As it is commonly done, we report the results computed on the test sets with the hyper-parameter and the number of training iterations that maximize the F_1 scores on the validation sets, thus using exactly the same metrics and procedures as were used to obtained the baseline results: scikit-learn (Pedregosa et al., 2011) with the "weighted" set-up, which computes the metrics for each relation, and reports their average, weighted by support (the number of true instances for each relation). For HypeNet datasets, that was accordingly set to "binary". We also verified through personal communications with the authors of Shwartz and Dagan (2016) that our metrics are numerically identical for the same sets of predicted labels.

For our path-based models, all the trainable parameters were initialized by a normal distribution around 0 average and standard deviation of 1. We used the same transformer architecture and hyper-parameters as in Devlin et al. (2018) (BERT mono-lingual English uncased version) which has 12 layers and the output vector size of 768, resulting in the total number of trainable parameters of 110 million. As it is commonly done when using a pre-trained transformer, we initialize our weights to those that were already trained by Devlin et al. (2018) for a language model and next sentence prediction tasks on a copy of English Wikipedia text and the BookCorpus. For consistency with the data used during pre-training, we add the same special markers before, between and after our input word sequences x and y.

dataset	dataset relations	#instances	#unique X	#unique Y
Hypenet Lexical	is a	20335	16044	5148
Hypenet Random	is a	49475	38020	12600
K&H+N	is a, part of	57509	1551	16379
BLESS	is a, part of, event, attribute	26546	201	8089
ROOT09	is a	12762	1218	3436
EVALution	is a, part of, attribute, opposite, made of	7378	1631	1497

Table 1: The relation types and statistics in each dataset.

4.4. Comparing against the Baselines

Table 2 presents our results. For additional comparison, we also include "Before Baseline" row, which lists the baselines used in Shwartz et al. (2016) and Shwartz and Dagan (2016). For HypeNet Random and Evalution datasets, we put the larger values that we obtained in our re-implementation of the distributional methods that they used rather than their reported values. The following can be observed:

1) Our **neural word path model** has been able to improve the state-of-the-art on three (3) out of six (6) datasets: Hypenet L, Bless and Root09. The differences are statistically significant at the level of .01. On the remaining three (3) datasets (HypeNet Random, K&H+N and Evalution), our results are the same as with the baseline performance (no statistically significant difference at the level .05). The baseline did not improve on those datasets over the prior work either. The scores for HypeNet Random and K&H+N are already high due to "lexical memorization" mentioned above. Since the compared models used exactly the same data, the obtained results clearly suggest that *our neural model is better* than the current state-of-the-art word-path model by Shwartz et al. (2016).

2) Our transformer-based model has also demonstrated tangible gains over both state-of-the-art baselines (**path-based and distributional**) on four (4) out of six (6) datasets and worked as well on the remaining two (2). Those differences are statistically significant at the level of .01. There are no statistically significant differences on Bless and K&H+N.

3) On four (4) out of six (6) datasets, our **distributional model worked better** than our neural word path model. The differences are statistically significant at the level of .01. There are no statistically significant differences on the remaining two. This suggests that *an attention-based transformer is a very powerful mechanism for modeling semantic relations*. Although they have been shown to be very effective in many other applications, this is the first study that has used them for semantic relations.

4) **Combining** our models results in further error reduction on HypeNet Lexical, Bless and Root. While smaller, those reductions are still statistically significant at the level of .01. The reductions are not surprising since the models use somewhat different data sources. We left more powerful combination models for future research.

We estimated the human performance on our datasets by giving 100 randomly selected word pairs to 3 independent graders, who were allowed to look up the meanings online (last row). It can be seen that the state-of-the-art approaches have already achieved the human level on the datasets where no improvement was detected (HypeNet Random and K&H+N), so this may explain why our approaches did not substantially improve them any further.

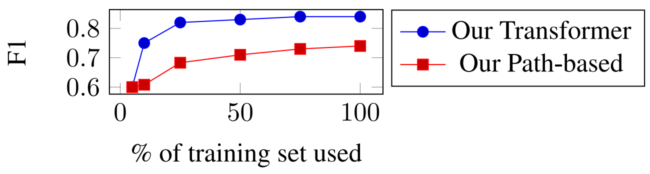

Figure 4: Using only portion of HypeNet Lexical dataset for training.

4.5. Additional Comparisons

4.6. Observations

We also tried to play an adversarial role and fed more challenging pairs to the trained models to see when they are starting to fail. Our attention-based transformer model trained for HypeNet Lexical dataset (named entities mostly) erroneously classified all the 100 examples created by combining random general words and the word "air" (e.g. "car air", "circle air", "new air") as "airline." It also erroneously classified all the 30 correct airline names that we tried as "airports" in addition to correctly classifying them as "airline." The proportion of correct airline names classified as "recording label" was 60%, which is lower than for the correct category, but still alarmingly high. Meanwhile, general words (like "car", "book", "new", etc.) are very rarely classified as members of any categories in this dataset since the model correctly sees that they are not named entities. Those observations suggest that what the transformer actually learns for this dataset is to use the combined properties of a word sequence (n-gram) to check if it can possibly be a named entity, and then if it *topically* fits the category (e.g. "aviation" in general). Those two conditions are sufficient to make a positive classification and to obtain high scores since very few test categories in the dataset are closely related (e.g. "airport" and "airline"). While our neural path models don't make such mistakes, their mistakes are primarily due to no word paths existing between the candidates in the training corpus, which was already noted in the related prior work. This suggests that a combination of those two approaches may provide additional gains over each. We have left more formal exploration of those observations for future studies.

4.7. Ablation Studies

We have also tested the influence of training size on the model by comparing its performance with 5%, 10%, 25%,

Model	Hypenet L	Hypenet R	K&H+N	BLESS	ROOT09	EVALution
Prior Shwartz et al. (2016)	0.660	0.890	0.983	0.889	0.788	0.595
Word path models:						
Shwartz et al. (2016)	0.700	0.901	0.985	0.893	0.814	0.600
Our neural model	0.739	0.897	**0.991**	0.925	0.831	0.600
Distributional:						
Wang et al. (2019)	N/A	N/A	**0.990**	0.938	0.861	0.620
Our transformer-based	**0.832**	**0.905**	0.987	**0.942**	**0.891**	**0.701**
Our combination	*0.841*	*0.905*	*0.986*	*0.949*	*0.903*	*0.700*
Human	0.90	0.90	0.98	0.96	0.95	0.82

Table 2: F_1 scores of our tested models compared to the state-of-the-art baselines.

Model	Hypenet L	Hypenet R	K&H+N	BLESS	ROOT09	EVALution
No hidden layer	-11 %	-9%	-.1%	-31%	-12%	-.4%
8 filters only	-5%	-4%	-.2%	-5%	-4%	-1%
4 filters only	-10%	-8%	-1%	-10%	-9%	-5%
Half of word paths used	-8%	-9%	-.5%	-8%	-8%	-5%
Quarter of paths used	-15%	-21%	-1%	-10%	-10%	-9%

Table 3: Ablation results for our neural word path model: relative loss of F_1.

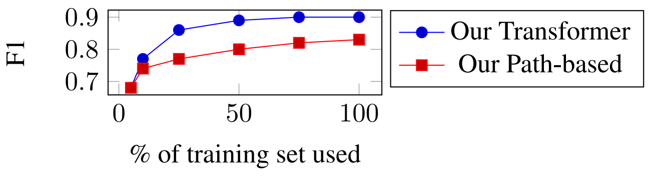

Figure 5: Using only portion of Root dataset for training.

50% and 75% of randomly selected training subsets of the two datasets on which both our models provided the biggest gains over the baselines: HypeNet Lexical and Root09. The results shown in figures 4 and 5 suggest the importance of the dataset size and the possibility of further improvements when more training data is available for the path-based. At the same time, out transformer-based model needs much less training to reach its top possible performance.

We also verified that all the components of our path-based model here are essential to exceed the baselines, specifically: using a hidden layer, using all the available word paths, using all 16 filters. Larger number of filters did not result in any gains, but increased the training time. Table 3 presents the relative loss in F_1 due to a particular ablated configuration.

5. Conclusions

We have considered the task of automatically recognizing semantic relations between words (phrases, concepts, etc.) such as "is a", "part of", "property of", "opposite of" etc., which is an important task affecting many data applications. While manually curated dictionaries exist in well resourced languages such as English or German, they are often out-of-date and not covering specialized domains, so the strong need for automated classification remains. Using six standard datasets, we have demonstrated that both distributional and word path state-of-the-art approaches can be tangibly improved. Out of those two approaches that we suggested, the transformer-based distributional approach worked significantly better. It has decreased the gap between the current strong baselines and human performance by roughly 50% for those datasets that still had room for improvement. We are not aware of any other work applying an attention-based transformer for this task. The combination of our two approaches demonstrates additional gains, since the two approaches use somewhat different data sources. We have also illustrated that the transformer-based model still has its own limitations, which are not always revealed by the standard datasets, e.g. learning to reliably recognize names (countries, cities, companies, etc.) and their relatedness to a certain topic (e.g. "aviation"), but still sometimes failing to distinguish between closely related categories such as "airport" and "airline". More sensitive datasets and models will be required in future to overcome those limitations, which we are currently planning. We plan a number of heuristic improvements, such as integrating training of the transformer with the semantic classification task on a deeper level. Another research direction concerns the lack of resources for training, which can be overcome by using cross-lingual models, especially between closely related languages (Doval et al., 2019; Sharoff, 2019; Lample and Conneau, 2019).

6. Bibliographical References

Baroni, M. and Lenci, A. (2011). How we blessed distributional semantic evaluation. In *Proceedings of 2011 workshop on geometrical models of natural language semantics*.

Bybee, J. L. and Beckner, C. (2015). Usage-based theory. In *The Oxford Handbook of Linguistic Analysis*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv preprint arXiv:1810.04805*.

Doval, Y., Camacho-Collados, J., Espinosa-Anke, L., and Schockaert, S. (2019). Meemi: Finding the middle ground in cross-lingual word embeddings. *arXiv preprint arXiv:1910.07221*.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *ACL 1992*.

Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining. In *arXiv preprint arXiv:1901.07291*.

LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. In *Nature 521(7553)*.

Levy, O., Remus, S., Biemann, C., and Dagan, I. (2015). Do supervised distributional methods really learn lexical inference relations? In *Proceedings of NAACL-HLT*.

Ma, Y., Wang, S., Aggarwal, C. C., Yin, D., and Tang, J. (2019). Multi-dimensional graph convolutional networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*.

Mikoajczak-Matyja, N. (2015). The associative structure of the mental lexicon: Hierarchical semantic relations in the minds of blind and sighted language users. In *Psychology of Language and Communication (19)*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *NIPS 2013*.

Nakashole, N., Weikum, G., and Suchanek, F. (2012). Patty: A taxonomy of relational patterns with semantic types. In *2012 Joint Conference EMNLP and CoNLL*.

Necsulescu, S., Mendes, S., Jurgens, D., Bel, N., and Navigli, R. (2015). Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *SEM 2015*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Machine learning in python. In *Journal of Machine Learning Research (12)*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *EMNLP 2014*.

Riedel, S., Yao, L., McCallum, A., and Marlin, M. B. (2013). Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT 2013*.

Roller, S., Kiela, D., and Nickel, M. (2018). Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *ACL 2018 (short paper)*.

Santus, E., Yung, F., Lenci, A., and Huang, C.-R. (2015). Proceedings of the 4th workshop on linked data in linguistics: Resources and applications.

Santus, E., Lenci, A., Chiu, T.-S., Lu, Q., and Huang, C.-R. (2016). Nine features in a random forest to learn taxonomical semantic relations. In *LREC 2016*.

Sharoff, S. (2019). Finding next of kin: Cross-lingual embedding spaces for related languages. *Journal of Natural Language Engineering*, 25.

Shwartz, V. and Dagan, I. (2016). Path-based vs. distributional information in recognizing lexical semantic relations. In *COLING 2016*.

Shwartz, V., Goldberg, Y., and Dagan, I. (2016). Improving hypernymy detection with an integrated path-based and distributional method. In *ACL 2016*.

Shwartz, V., Santus, E., and Schlechtweg, D. (2017). Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *EACL 2017*.

Snow, R., Jurafsky, D., and Ng, A. Y. (2004). Learning syntactic patterns for automatic hypernym discovery. In *NIPS 2004*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *NIPS 2017*.

Wang, C., He, X., and Zhou, A. (2019). Spherere: Distinguishing lexical relations with hyperspherical relation embeddings. In *ACL 2019*.

Washio, K. and Kato, T. (2018). Filling missing paths: Modeling co-occurrences of word pairs and dependency paths for recognizing lexical semantic relations. In *NAACL-HLT 2018*.

Yuan, Y. and Sharoff, S. (2020). Sentence level human translation quality estimation with attention-based neural networks. In *Proc LREC*, Marseilles, May.

Zeng, T. and Ji, S. (2015). Multi-dimensional graph convolutional networks. In *IEEE International Conference on Data Mining*.