Language Resources and Evaluation (2023) 57:1463–1486
https://doi.org/10.1007/s10579-023-09677-w

ORIGINAL PAPER

Fine-tuning language models to recognize semantic relations

Dmitri Roussinov[1] · Serge Sharoff[2] · Nadezhda Puchnina[3]

Accepted: 15 May 2023 / Published online: 23 July 2023

Abstract
Transformer-based pre-trained Language Models (PLMs) have emerged as the foundations for the current state-of-the-art algorithms in most natural language processing tasks, in particular when applied to context rich data such as sentences or paragraphs. However, their impact on the tasks defined in terms of abstract individual word properties, not necessary tied to their specific use in a particular sentence, has been inadequately explored, which is a notable research gap. Addressing this gap is crucial for advancing our understanding of natural language processing. To fill this void, we concentrate on classification of semantic relations: given a pair of concepts (words or word sequences) the aim is to identify the semantic label to describe their relationship. E.g. in the case of the pair *green/colour*, “is a” is a suitable relation while “part of”, “property of”, and “opposite of” are not suitable. This classification is independent of a particular sentence in which these concepts might have been used. We are first to incorporate a language model into both existing approaches to this task, namely path-based and distribution-based methods. Our transformer-based approaches exhibit significant improvements over the state-of-the-art and come remarkably close to achieving human-level performance on rigorous benchmarks. We are also first to provide evidence that the standard datasets over-state the performance due to the effect of “lexical memorisation.” We reduce this effect by applying lexical separation. On the new benchmark datasets, the algorithmic performance remains significantly below human-level, highlighting that the task of semantic relation classification is still unresolved, particularly for language models of the sizes commonly used at the time of our study. We also identify additional challenges that PLM-based approaches face and conduct extensive ablation studies and other experiments to investigate the sensitivity of our findings to specific modelling and implementation choices. Furthermore, we examine the specific relations that pose greater challenges and discuss the trade-offs between accuracy and processing time.

Keywords Pre-trained language models · Transformers · BERT · T5 · Semantic relations

Extended author information available on the last page of the article

1 Introduction

During the last couple of years, pre-trained Language Models (PLM) such as Bert, GPT or T5, have significantly advanced the state-of-art performance on most natural language processing (NLP) tasks (Devlin et al., 2018; Lample and Conneau, 2019; Raffel et al., 2020). The vast majority of those models have been successfully applied to modelling word representations when they are used in context (sentence, paragraph, etc.). At the same time, whether similar gains can be achieved in the tasks where the words are modelled in their abstract sense (outside of particular context) still remains to be seen. This includes classification of semantic relations: given a pair of concepts (words or word sequences) the aim is to identify the best semantic label to describe their relationship. The possible labels are "is a", "part-of", "property-of", "made-of", etc., depending on a particular benchmark set. Not being able to perform this task with the accuracy approaching human would signal that we are still far from Artificial General Intelligence. The task also has practical importance: manually curated dictionaries with their semantic relations exist only in well resourced languages such as English or German. They are often out-of-date and are not covering specialized domains, e.g. medical or legal. Despite the great effort invested in their creation and maintenance, even the largest ones (e.g., Yago, DBPedia or Wikidata) remain incomplete (Schlichtkrull et al., 2018). Therefore, there is still significant interest in methods for automated relation discovery, such as knowledge acquisition, taxonomy mining, ontology building, etc.

Prior 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 and Dagan (2016), essentially look for certain patterns in the joint occurrences of words (phrases, concepts, etc.) in a 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 modelling the occurrences of each word, *X* or *Y*, separately, not necessary in the proximity to each other.

One main goal here is to *improve, compare and combine those two classes of approaches*. While recently several works appeared that approached the problem in unsupervised (few-shot or zero-shot) settings, e.g. Petroni et al. (2019) , here we focused on the supervised approaches aiming to deliver state-of-the-art performance as in Shwartz and Dagan (2016), Wang et al. (2019). Thus we follow their empirical methods and involve the same datasets for training and testing to allow rigorous comparison. Our contributions are as follows:

1. We are the first to successfully incorporate a pre-trained language models (PLMs) into the task of semantic classification. Consequently, we have improved both approaches to the task, namely path-based and distribution-based methods. Through rigorous evaluation using standard benchmarks we demonstrate that our PLM-based approach significantly surpasses the state-of-the art methods.

We specifically investigate the use of two popular PLMs: BERT (Devlin et al., 2018) and T5 (Raffel et al., 2020). Despite its smaller size in terms of trainable parameters and pre-training corpus, BERT outperforms T5 on most tasks.

2. While the PLM-based approach is closely approaching human level performance on the currently standard datasets, we are first to point out that it still has certain limitations which may not be evident when relying solely on existing benchmarks. Following the procedure of lexical separation (Levy et al., 2015), we have created more challenging datasets that significantly widen the gap between algorithmic and human performance, highlighting that this task is far from being solved.
3. We have performed extensive ablation experiments to test the importance of the specific components and decisions within our models (such as the depth of the layer at which the pre-trained transformer is used), and the size of the training data needed. All of this is important for the practical applications when deciding on the trade-off between the accuracy and speed.
4. For the situations when involving a PLM is impossible or undesirable (e.g. when using limited computational resources or dealing with rare languages) We introduce our novel path-based model that combines useful properties of convolutional and recurrent 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.

We make our code and data publicly available.[1] The next section overviews the prior related work. It is followed by the description of the models, followed by our empirical results.[2] Finally, we present our conclusions and future directions.

2 Related studies

2.1 Without using PLMs

The approaches to automatically classifying semantic relations between words prior to the introduction of pre-trained language models (PLMs) can be divided into two major groups: (1) path-based and (2) distributional. *Path-based approaches* essentially look for certain patterns in the joint occurrences of words (phrases, concepts, etc.) in some validation text 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).

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 (Roussinov and Turetken, 2007; Nakashole et al., 2012; Riedel et al., 2013).

[1] https://github.com/dminus1/semantic.

[2] We presented much less detailed and weaker preliminary results at LREC 2020 and ECIR 2020 conferences.

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).

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), including "part of", "property of", "opposite of", "made of", "event", etc.

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.

Thus, depending on the size of the set of target pairs, the total number of paths included in the data is at the order of magnitude of millions. The paths include the sentence dependency information obtained by applying SpaCy parser.[3]

The Lexnet authors applied specially-crafted heuristic rules to remove certain words to simplify the paths. For example, a path *parrot is a bird*, is represented as *parrot/NOUN/subject be/verb/root bird/noun/attribute*. Each path consists of edges (words). Each edge is represented by an embedding vector obtained by concatenating the embedding vectors of the words and the associated dependence symbols. Thus, each word path p is represented as a sequence of vectors

$$\{\overrightarrow{v_{e1}}, \overrightarrow{v_{e2}}, \overrightarrow{v_{e3}}, ...\} = \{\overrightarrow{v_{et}}(x, y)\} \text{ for } t = 1, ..., l_p \tag{1}$$

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:

$$\overrightarrow{v_p}(x, y) = \text{RNN}(\overrightarrow{\{v_{et}(x, y)\}}) \tag{2}$$

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

$$\vec{v}_{xy} = \frac{\sum_p \overrightarrow{v_p}(x, y)}{\#Paths(x, y)} \tag{3}$$

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 and Dagan, 2016). Specifically, Shwartz et al. (2017) performed 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

[3] https://spacy.io/

corpus. Their approach applies to nouns only. 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 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 hyper-spherical 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 (SphereRE).

2.2 Using PLMs

Since pre-trained language models (PLMs) have recently achieved new state-of-the-art performance in many NLP applications (Devlin et al., 2018; Lample and Conneau, 2019; Raffel et al., 2020), it is not surprising that researchers have started to assess how much PLMs can help in various tasks involving the application of semantic knowledge: One of the key features of PLMs is the ability to compute a contextual representation for each word in a sentence. This is a major advancement from context-invariant embeddings where all instances of a polysemic word would correspond to the same vector. While we are not aware of any studies focusing on using PLMs for semantic relation classification, there have been studies on the related tasks, some of them carried out at the same time as our experiments reported here. The majority of those studies introduced their own datasets (called *probing classifiers*), manually or semi-automatically crafted to assess PLM's semantic and other types of knowledge, and sometimes without comparing PLM approaches against the state-of-art supervised non-PLM models (as we do in this study). In the remaining of this subsection, we briefly overview the related prior works utilising PLMs.

Tenney et al. (2019) showed that BERT embeddings encode information about parts of speech, syntactic chunks and roles. They also discovered that BERT excels in syntactic analysis, but provides only small improvements for tasks that require semantic knowledge.

Using their suggested probing classifiers called LAnguage Model Analysis (LAMA), Petroni et al. (2019), successfully demonstrated that PLMs can recall factual knowledge that is part of their training corpus. Their set of classifiers consists of knowledge base triples that are placed into templates with missing objects, e.g. "Obama was born in". The authors report that for some relations (particularly N-to-M relations, which are among those studied here) the performance is still far below human. Roberts et al. (2020) performed similar explorations using T5 model (Raffel et al., 2020).

McCoy et al. (2019) discovered that, in the case of natural language inference, a BERT-based model heavily relies on fallible syntactic heuristics rather than attaining a profound comprehension of the natural language input. Kassner et al. (2020) made modifications to the LAMA dataset by incorporating negation and distractors, illustrating that the direct application of PLMs can produce spurious outcomes (e.g., BERT showing equal propensity to generate 'Birds can fly' and 'Birds can't fly'). They conclude that "PLMs still have a long way to go to adequately learn human-like factual knowledge." According to their findings, pre-trained BERT does not effectively model *negation*, although fine-tuned BERT accurately classifies sentences as true or false. Similar observations were reported by Ettinger (2020), who also observed that BERT can generally discriminate between good and bad completions involving shared categories or role reversals, reliably retrieves noun hypernyms, but encounters difficulties in challenging inference and role-based event prediction.

Zhou et al. (2020) investigated the common sense ability of several PLMs, including GPT, BERT, XLNet, and RoBERTa. While their findings are similar to those mentioned above, they also observe that bi-directional context and larger training corpus typically give an advantage. They also discovered that current models perform poorly on tasks that require several inference steps. An example of this in the context of semantic relations would be inferring that *mosquito is a living organism* from the fact that *mosquito is an insect*. Kim et al. (2019) looked at word sense information contained in BERT embeddings. Mickus et al. (2020) noted that BERT's representations of the same word vary depending on the position of the sentence in which it occurs, possibly because that one of the two tasks on which it is pre-trained is a next-sentence prediction.

A number of studies looked at numeric values of self-attention, e.g. Clark et al. (2019). However, the methodological soundness of this approach has been also since debated, e.g. Brunner et al. (2020), since in a multi-layer model the attention is followed by non-linear transformations, so the patterns in individual heads do not necessarily provide a meaningful picture. For this reason, we have not included numeric analysis of the attention distribution in our study.

PLMs have also been successfully applied to a related but distinct task of *relation(-ship) extraction*, e.g. Shi and Lin (2020). Its goal is to identify instances of semantic relations (e.g. a drug affecting an illness, a protein affecting another protein, a head of state etc.) as expressed in specific contexts, typically text sentences. Semantic classification task we are focusing on here is classifying the abstract relations between concepts X and Y, *regardless of specific context*, e.g. "green" and "colour". Once big difference between the two tasks is that relation extraction primarily works with named entities and a set of pre-determined relations, while the words of interest in semantic-classification are often involve common nouns. The datasets and state-of-the-art approaches for these tasks differ. Another related but a different task where transformers and PLMs have been successfully applied is semantic frame parsing, e.g. Gupta et al. (2018), Aghajanyan et al. (2020).

3 Models for semantic relations

3.1 Our path-based neural model

3.1.1 Overview of the approach

Our proposed path-based neural model combines useful properties of convolutional and recurrent networks, while resolving several shortcomings of the current

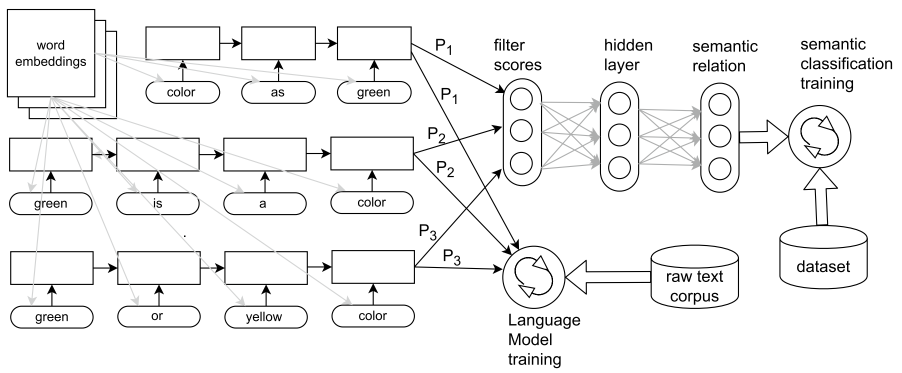

Fig. 1 Our path-based neural approach to semantic relationship classification. The trained language model implemented as a recurrent network (rectangles) applies to each "soft" (trainable) pattern consisting of the target words (*green* and *colour* in the example) and "pseudo" words represented by trainable word vectors. The probabilities obtained from the language model (P1, P2, P3 in the example) are fed into a semantic relation classifier. The language model, the semantic classifier and the "soft" pattern vectors are trained altogether. The example uses real words in the patterns (*is, as, or* etc.) only for illustration. They are represented by trainable vectors in the model, thus are not restricted to the actual words in the vocabulary

state-of-the-art model by Shwartz and Dagan (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. Unlike PLM-based models, rather than building an LM for the entire training corpus, our path-based model builds LM for its subset: only for the word paths connecting pairs of words (X,Y) of interest (target pairs) within the same sentence. In our experiments here, we used exactly the same word paths as in our baselines from Shwartz and Dagan (2016). They are publicly available and are already described above in our Sect. 2. Since Wikipedia serves as the source for our word paths, and also as the corpus to pre-train one of our transformer-based language models (BERT), the performance between the two approaches can not be simply attributed to the size of the training data.

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* as explained in the following. Inspired by the success of convolutional networks (CNNs) in computer vision and other applications, we use a fixed set of trainable filters (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 recurrent LM may suggest that there is a high probability of occurrence of the sequence *green is a colour* in raw text corpus. Combining those two facts suggests that *green* belongs to the category *colour* (true *is-a* relation between them). Figure 1 shows only three 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 colour* and *green is a relaxing colour.* 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 colour* also scores highly even if it never occurs in the corpus. Thus, as with CNNs used in computer vision, the limited number of filters does not necessary limit the number of patterns to which those filters may respond. This is because both the filters and the word vectors (embeddings) are trainable, thus even different patterns can produce high scores with the same filter.

Since the current state-of-the art path-based approach (Shwartz and Dagan, 2016) aggregates the word paths connecting each target pair by averaging the context vectors representing all the paths (formula 3), 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 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 and Dagan, 2016), which uniformly uses the labels "X" and "Y" for the path ends (e.g. "X is a type of Y" rather than "green is a type of colour") 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 learn to transfer the knowledge from similar words.

3.1.2 Formal definitions

Language Model (LM) is a probability distribution over sequences of words: $p(w_1, ..., w_m)$, where $w_1, ..., w_m$ is any arbitrary sequence of words in a language. We train LM jointly with our semantic relation classification task by minimizing cross-entropy costs, equally weighted for both tasks.

We use a recurrent neural network, specifically a GRU variation (Cho et al., 2014), which works as well as LSTM while being faster to train. In our preliminary experiments we also used a single-layer fully-connected network which resulted is weaker results so we are omitting them here. We also tried bi-directional versions, but that did not affect the results in a noticeable way).

Thus, the probability of a word w_m in the language to follow a sequence of words $w_1, ..., w_{m-1}$ is determined by using the RNN to map the sequence $w_1, ..., w_{m-1}$ into its context vector:

$$\vec{v}_{w_1,...,w_{m-1}} = \text{RNN}(w_1, ..., w_{m-1}) \quad (4)$$

and then applying a linear mapping and the softmax function:

$$p(w_m|w_1, ..., w_{m-1}) = \\ \text{softmax}\,(W \cdot \vec{v}_{w_1,...,w_{m-1}} + b) \tag{5}$$

where W is a trainable matrix, b is a trainable bias, and *softmax* is a standard function to scale any given vector of scores to probabilities.

As any typical neural LM, our LM also takes distributed representations of words as inputs: all the words are represented by their trainable *embedding* vectors $v_1, ..., v_m$.[4] This is important for our model and allows us to *treat LM as a function defined over arbitrary vectors* $p(v_m|v_1, ..., v_{m-1})$ *rather than over words.*

To classify semantic relations, we only look at the word paths that connect the target word pairs. Thus, we only make use of probabilities of the form $p(v_y|v_x, v_1, ..., v_k)$, where (x, y) is one of the *target pairs* of words - those in the dataset that are used in training or testing the semantic relations, (v_x, v_y) are their embedding vectors. The sequence of vectors $v_1, ..., v_k$ defines a trainable *filter*, and k is its size. While vectors $v_1, ..., v_k$ have the same dimensions as the word embeddings, they are additional parameters in the model that we introduce. They are trained with the other ones (word embeddings + RNN matrices + the decision layer) by back propagation. Due to the smoothness of a neural LM, the entire model is differentiable.

Thus, we formally define the score of each of our filters (kernels) the following way:

$$f_i = \log p(v_y|v_x, v^i_1, ..., v^i_k) \tag{6}$$

where $p()$ is determined by our language model as the probability of the word with the embedding vector v_y to follow the sequence of words with the vectors $v_x, v^i_1, ..., v^i_k$. We apply *log* in order to deal with high variation in the orders of magnitude of $p()$. Finally, we define the vector of filter scores by concatenating the individual scores: $\vec{f'} = [f_1,f_2,f_3, ..f_N]$, where N is the total number of filters (16 in our study here). They are concatenated with X and Y embedding vectors to produce the final vector of "semantic" features $\vec{f}$ which is mapped into a specific relation label by using a neural network with a single hidden layer. Thus, we define:

$$\overrightarrow{h_1} = \tanh(W_2 \cdot \vec{f} + b_2) \tag{7}$$

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:

$$c = \text{argmax}\,(W_3 \cdot \overrightarrow{h_1} + b_3) \tag{8}$$

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

$$cost = -\log((\text{softmax}\,(W_3 \cdot \overrightarrow{h_1} + b_3))[c_l]) \tag{9}$$

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

where c_l is the correct (expected) class label. We used stochastic gradient descent for cost minimization.

Earlier into our experiments, we had a theoretical concern that the conditional probabilities of target word occurrences are subject to frequency bias. However, we did not observe the impact of that that in practice, possibly due to the fact that their word embeddings are also fed into the classifier, thus it learns to adjust for more/less frequent words.

3.2 Attention-based transformer as PLM

Here, we briefly explain how the attention-based transformer (Vaswani et al., 2017) operates and how we use its pre-trained versions for a language model, specifically BERT (Devlin et al., 2018) and T5 (Raffel et al., 2020).

Instead of recurrent units with "memory gates" essential for RNNs, 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.

A full transformer consists of a decoder and encoder, and maps a sequence of vectors into another sequence of vectors, which in turn can be used in the downstream task, e.g. generating output words for machine translation or a sentence classification such as sentiment analysis.

The conversions from the inputs to the outputs are performed by several layers, which are identical in their architecture, possibly varying only in their hyperparameters and trained weights. 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, e.g. *tanh*:

$$\vec{v_i}' = \tanh(W \cdot \sum_{t=1}^{k} \alpha_t \vec{v_t}) \tag{10}$$

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 possibly 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.

(Vaswani et al., 2017) uses a scaled dot product of the vectors $\vec{v_i}$

and $\vec{v_t}$:

$$\alpha_t = \vec{v_i} \cdot W' \cdot \vec{v_t} \tag{11}$$

where W' is a trainable matrix (same regardless of i and t at the same layer but possibly 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 application such as text classification or generation. As it is commonly done with the transformers, we make our output classification decision based on the very first vector on the top level (called “classification token”). We do not use a hidden layer here, so we apply our formula 8 above to h_1 defined as the following:

$$\overrightarrow{h_1} = \overrightarrow{v_0^u} \tag{12}$$

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

Since the operations above are primarily matrix multiplications, they are algorithmically faster than back-propagation in time needed for recurrent neural networks. As a result, ready-to use attention-based models have been trained using large corpora such as Wikipedia.

We did not need to use BERT’s decoder part for this task since our output is not a sequence. As common in other applications and recommended by Devlin et al. (2018), we have also not used the decision making layer of BERT pre-trained on masked word and next sentence prediction. Thus, we fed the inputs to BERT in the standard two-sentence format: [CLS] X [SEP] Y [SEP], and set the required masks according to the documentation. Here, (X,Y) is the target pair of words. Based on transformers architecture, we conjectured that explicitly defining CNN-like filters defined by us in formula 6 of our path-based model is not necessary here, since a transformer should be able to “add" those words prior, between or after (X, Y) if needed on higher layers.

As we detail in our 4, indeed adding those filters explicitly did not affect the performance in any way. Thus, we argue and empirically verify that PLM approach can be viewed as both distributional and path-based. This is because the word path distribution is already contained in a PLM, so the latter takes over the role of the RNN showed on the left in our diagram in Fig. 1. The PLM also supplies the pre-trained word embeddings.

It is not possible to skip the decoder when using T5 since it is strictly a *sequence-to-sequence* model. Thus, as recommended by the documentation and used in the prior works, we assign the target text to the relation name, use the word "relation" as custom prefix (instead of the standard ones like “summarize", “predict sentiment", etc.) and put X and Y separated by empty space, without introducing any additional separators between them.

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. (*colour, 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 *colour* 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 generate 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 colour*, *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 and Dagan, 2016) above 95%. In order

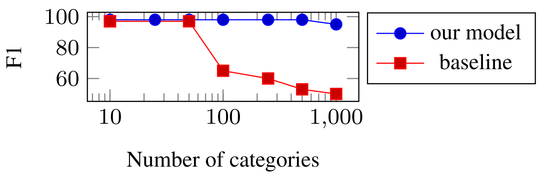

Fig. 2 F_1 scores of our model compared to the state-of-the-art baseline on synthetic data

Table 1 The relation types and statistics in each dataset

Dataset	Dataset relations	#Instances	#Unique X	#Unique Y
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
Hypenet Lexical	Is a	20335	16044	5148
Hypenet Random	Is a	49475	38020	12600

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 2 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, X-s is primarily a named entity (places, films, music albums and groups, people, companies, etc.). The important difference is that in order to create the partition 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". 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)*. It also has the smallest size. To make our comparison more direct, we used exactly the same partitions 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

We set the word embedding size, the RNN context vector size, and the hidden layer size 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.

For our PLMs, we used the base model of BERT (Devlin et al., 2018) (mono-lingual English uncased version) which has 12 layers and the output vector size of 768, with a total number of trainable parameters of 110 million. We also used "base" T5, which has 220 million parameters, and larger pre-training corpus than BERT.

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
Best word path models						
Concat	N/A	N/A	0.904	0.811	0.646	0.525
Lexnet	0.700	0.901	0.985	0.893	0.814	0.600
Our path language model	0.740	0.899	0.990	0.927	0.832	0.602
Best Distributional						
SphereRE	N/A	N/A	0.990	0.938	0.861	0.620
WBR	N/A	N/A	0.988	0.941	0.864	0.628
KEML	N/A	N/A	**0.993**	**0.944**	0.878	0.660
Our PLM-based						
Using BERT	0.832	**0.905**	0.987	0.942	**0.921**	**0.701**
Using T5	**0.843**	0.899	0.981	0.925	0.902	0.685
Human	**0.90**	**0.90**	**0.98**	**0.96**	**0.95**	**0.82**

4.4 Comparing against the baselines

Table 2 presents our results. We followed the same comparison methodology as in Shwartz and Dagan (2016), which provided our data and baselines. We performed pair-wise statistical significance tests as they are less affected by re-using the same data-points in different runs. Additionally, each reported result has been obtained by 4 different parameter initialisations and random presentation order of the training data. The maximum confidence interval observed was ±0.0031. For additional comparison, we include the "Concat" baseline result from Shwartz and Dagan (2016) obtained by a single layer fully connected model applied to the concatenation of the embeddings of the word pair. They also noticed that applying it to the difference of embeddings worked worse. Both approaches were explored prior to their work. While the results from KEML (Wang et al., 2021) were not available when we carried our experiments and presented preliminary results, we still have decided to include it here as an additional baseline. Similarly, we decided to include WBR from (Barkan et al., 2020). For HypeNet Random and Evalution datasets, we provided 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 within that type of approaches on three out of six datasets: Hypenet Lexical, Bless and Root09. The differences are statistically significant at the level of .01. On the remaining three 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 scores for HypeNet Random and K&H+N are already high due to "lexical memorization", as further tests below reveal. Since the models were evaluated on the same data, the obtained results clearly suggest

Table 3 F_1 scores of our tested PLM-based models on the lexically separated datasets showing larger gaps between human and algorithmic performance

Model	K&H+N	BLESS	ROOT09	EVALution
Lexical separation				
Path-Based	.721	.833	870	.587
BERT	.733	.855	.881	.594
T5	.719	.791	.853	.682
Undersampling to the same size:				
Path-based	.931	.886	.905	.601
BERT	.933	.889	.910	.607
T5	.963	.915	.898	.684
Human	0.98	0.96	0.95	0.82

that *our neural model is better than the current state-of-the-art word-path model* by Shwartz et al. (2016).

(2) Our PLM-based models have shown significant improvements *over state-of-the-art baselines*, including both path-based and distributional models, on four out of six datasets and worked as well on the remaining two (2). The differences are statistically significant at the level of .01. There are no statistically significant differences on Bless and K&H+N.

(3) On four out of six datasets, our *PLM models outperformed 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. It is noteworthy that the improvements due using *BERT and T5 were consistent* across all the datasets.

This suggests that *PLMs are highly effective in modeling semantic relations*. While they have demonstrated their effectiveness in various other applications, this is the first study that has used them for this particular task.

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.

4.5 Effect of lexical separation

The results in Table 2 make an impression that with the models that we are suggesting here, the algorithmic performance is closely approaching human level on 4 datasets out of 6, and only slightly below on the remaining 2. However, we argue that this impression is a result of somewhat over-estimated algorithmic performance due to the effect of "lexical memorization". Indeed, Levy et al. (2015) noted that supervised distributional methods instead of learning a relation between the two terms,

Table 4 F_1 scores for the specific semantic relations for the PLM-based models. The path-based results are included in parenthesis

Relation	Example	K &H+N	BLESS	ROOT09	Evalution
Hypernym	Cat-animal	0.95 (.93)	0.94 (.92)	0.98 (.91)	0.70 (.62)
Sister-term	Cat-dog	0.98 (.97)	0.97 (.95)	0.87 (.83)	
Meronym	Car-wheel	0.65 (.71)	0.93 (.92)		0.75 (.66)
Event	Axe-sharpen		0.90 (.89)		
Attribute	Apple-juicy		0.92 (.90)		0.87 (.76)
Random	Van-reservoir	0.98 (.98)	0.96 (.97)	0.92 (.84)	
Antonym	Join-quit				0.74 (.61)
Made of	Wind-air				0.67 (.58)
Synonym	Equal-even				0.43 (.37)

tend to 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*. Similar effect happens with respect to *Y*. The more times X and Y are repeated in the dataset (shown in Table 1), the higher the effect of "lexical memorization". This effect has been already alleviated by introducing negative examples into the ROOT09 dataset (Santus et al., 2016). In order to appraise this effect and to create more accurate benchmarks, we aslo applied the procedure of "lexical separation" from (Levy et al., 2015), so the training and testing parts don't have any words in common.

Table 3 presents the results of our PLM-based models on the lexically separated datasets. Since the separation decreases the training size by roughly a factor of 3, we also add the results on the same amount of training data without applying separation. We could not include the performance of Lexnet, SphereRE, WBR and KEML, on our lexically separated datasets since that would require re-implementing them. The following can be observed:

(1) The reduction in performance due to lexical separation is substantially larger than the drop due to reducing the training data size alone.
(2) On the lexically separated datasets, the algorithmic performance is still far from human. This suggests that modeling semantic relation task is still far from being solved.
(3) The performance drops are remarkably similar between BERT and T5 on all the datasets except Evalution (the most challenging out of 6 used here), in spite of their somewhat different models, number of trainable parameters and corpora used.

We make our lexically separated datasets available, hoping that in future they will be used by other researchers to assess how close the automated approaches are to the human level.

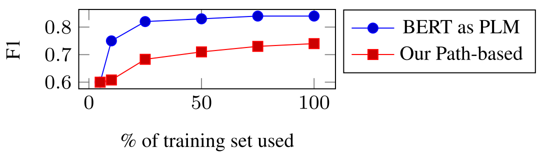

Fig. 3 Using only portion of HypeNet Lexical dataset for training

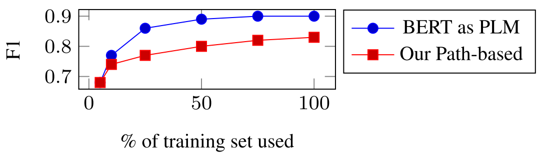

Fig. 4 Using only portion of Root dataset for training

4.6 Further analysis

Table 4 shows F1 scores for the specific relations from our BERT-based model followed by the results of our Path-based model in parenthesis. The blank entries are for the relations that are not present in that particular dataset.

The following can be observed: (1) The most challenging category is synonyms. Analysis of the confusion matrix suggests that it is often confused with antonyms and hypernyms, and also due to the fact that there are fewer paths that can link them which is consistent with the observation made by the prior research (Shwartz and Dagan, 2016) and with the gap in performance between our two types of models. (2) On *Root09* dataset, the hypernym relation works almost ideally, so the mistakes are almost exclusively because of confusion between the other two relations. 3) Meronyms are only challenging in *K&H+N* since there are only 768 training examples there out of 58,000 total.

We also informally explored the model's ability to handle adversarially created test pairs. We were motivated by a simple observation that the vast majority of positive examples in Hypenet dataset are named entities and that the vast majority of negative examples are not topically related ("oregon" and "piano", "asia" and "female", "russo" and "switzerland", etc.). We manually created 100 test cases consisting of made-up named entities on the topic of *aviation* by combining general words and the word "air" ("car air", "circle air", "new air", etc.). The model trained on HypeNet Lexical dataset erroneously classified all those cases as "airline." It also erroneously classified all the 30 correct airline names that we submitted as "airports" while still correctly classifying them as "airline" as well. The proportion of correct airline names classified as "recording label" was 62%, which is lower than for the correct category (hypernym), but still alarmingly high. We did not find a single false-positive within the entire original test-set that involved erroneous classification of a general word ("car", "book", "new", etc.) as something that expects a name entity. The above stated observations suggest that what the model designates all topically related named entities as positives and other candidates as negatives. Being able to check for those two conditions is sufficient to obtain a high score since very few categories in the dataset are topically related (e.g. "airport" and "airline"). We made similar observations using test-cases on other pairs of related topics,

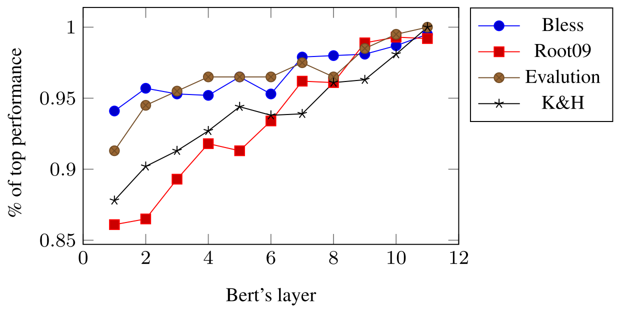

Fig. 5 Performance drop when using lower BERT layers

specifically *city/river, movie/book, and actor/director*."[5] Since our path-based experiments are limited by the available path data that we used from Shwartz and Dagan (2016), we were not able to carry out the same experiments with our path-based model, and this left it for future work.

4.7 Ablation studies

We have also tested the influence of training size on the model by comparing its performance with 5, 10, 25, 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 3 and 4 suggest the importance of the dataset size and the possibility of further improvements when more training data is available for the path-based approach. At the same time, out transformer-based model needs much less training to reach its top possible performance.

Table 5 shows the relative drop in performance (F1), when lower layers of BERT transformer were used. The following can be observed: (1) Lowering the level normally results in performance reduction. This is expected since less of the pre-trained information is used and the number of trainable parameters is reduced. However, the fact that the drop does not exceed 16 % suggests that the bulk of semantic knowledge is contained in the word embeddings rather than in the transformer's heads. (2) The relative decrease varies from 16% on Root to 6% on Bless. The difference can be explained by the larger size of Bless. (3) Depending on the preferences for the accuracy trade-off, on some datasets the lower layers can be used in practice in order to slash the processing time by a factor between 2 and 10, since according to the transformer algorithm, the processing time is roughly proportional to the number of layers used.

We also verified that using a hidden layer did not result in any increase in performance with either of our PLM-based transformer models. This is most likely due to the layered architecture of a transformer itself. Since out downstream task is

[5] The test sets are included in the released resource.

relatively simple (as compared e.g. with automated question answering), there likely to exist some over-parametrization at each of transformer's layer. The model can learn to use those under-utilized dimensions as a hidden layer since the transformations within a fully-connected network and an attention-based transformer are very similar.

We have also verified that the following did not have any noticeable effect on the performance: (1) Using any other position in the transformer's output instead of the recommended classification token ("[CLS]"). (2) Not adding the special tokens "[CLS]" and "[SEP]" at all. (3) Concatenating the entire layer instead of using the classification token, or several (up to 4 in our experiments) layers. The explanation for that probably lies again in the fact that the task is much less data intensive in comparison from the primary tasks for which the transformer is pre-trained (masked word or next sentence prediction), thus the model learns to delegate the role of special tokens to those under-utilized parameters.

We have experimented with adding reserved ("[unused*]") or real but rarely used words from BERT's vocabulary before, between and after X and Y, hoping that this may help the model to learn certain validation patterns (e.g. "X is a Y" for hyponymy), but we did not observe any effect of this on the performance. Similarly, no effect was observed when doing this in T5.

Using "small" version of T5 indeed resulted in reduced performance. We did not try run larger T5 versions than the base one. Using large BERT did not result in any noticeable improvements.

5 Conclusions

We have presented our original exploration on the successful utilisation of a pre-trained language model (PLM) in the identification of semantic relations. Given a pair of words (phrases, concepts, etc.) X and Y, the task is to automatically classify which relation label (e.g. "is a", "part of", "property of", "opposite of") describes the pair best irrespective of the specific context. We have made improvements to and compared two families of approaches derived from prior research: (1) those that model the existing word paths connecting the given pair in the corpus, and (2) those that focus on modelling the occurrences of either X or Y separately, regardless of their proximity to each other.

In order to introduce PLM into a path-based approach, we have come up with an original model that combines useful properties of convolutional and recurrent networks and avoids limitations of the prior models. Our model has improved the state-of-the-art performance within this family of approaches on 4 out of 6 standard datasets. An important feature of our approach is that it builds a language model not for the entire corpus, but only for the subset: the word paths that connect X and Y. This may result in significant computational savings.

Our implementation of a distributional approach uses a pre-trained language model (BERT or T5) pre-trained on the entire corpus, and thus digests larger amount of data and is more computationally demanding. Such an approach can be viewed as both path-based and distribution-approach at the same time. While fine-tuning classification step to the initial word representation is a standard architecture used for many current NLP tasks, at the time of our experiments and at present, our work stands out as the first to apply PLM to model global abstract properties of such small building blocks of human language as separate words, rather than their contextual usage in specific sentences or paragraphs.

Our PLM-based approach has further improved the state-of-the-art performance on 4 out of the same 6 datasets, approaching the human level performance. We have performed rigorous ablation studies on the importance of the specific parts of our models (such as the depth of the layer at which the pre-trained transformer is used), and the size of the training data needed. Furthermore, we demonstrate, that the current standard datasets tend to over-estimate the performance due to the effect of "lexical memorization". By applying the procedure of *lexical separation*, we have produced the datasets that don't have any common words between the testing and training partitions. When measured on those datasets, the best algorithmic performance is still significantly below that of humans. Thus, *automated semantic classification remains an unsolved task* and poses numerous challenges for future research.

The importance of this task extends to various practical applications and is considered fundamental in the development of Artificial General Intelligence. While manual dictionaries exist for semantic relations, they are primarily available for well-resourced languages like English or German, often lack updates, and may not cover specialized domains. Therefore, there is a strong demand for automated relation classification, particularly in domains such as medicine and law. To facilitate further research and practical implementations, we make our code and data publicly available.

There is an emerging topic of detecting what pre-trained models "know" about language and the world. While it has been shown that their weights can be used to predict some linguistic properties, such as semantic roles (Rogers et al., 2020) or word analogy (Vulic et al., 2020), our study is the first to show that the pre-trained models still have limitations, which are not always revealed by the standard datasets. For example, they can learn to reliably recognise names (countries, cities, companies, etc.) and their relatedness to a certain topic (e.g. "aviation"), but still they often fail to distinguish between closely related categories such as "airport" and "airline". Since these observations only scratch the surface, so we have left more methodological exploration for future studies.

Another promising research direction concerns the ability to transfer prediction models to less resourced languages via multilingual transfer, for example via multilingual embeddings (Sogaard et al., 2019), especially when we want to transfer our models between closely related languages, such as French and Italian (Sharoff, 2020). It is known that multilingual PLMs share enough information between languages to make this successful (Lample and Conneau, 2019). However, more specific experiments are needed to prove the possibility of successful fine-tuning on the lexical relations task across languages. The mechanisms inspired by meta-learning can be also applied, e.g. Roussinov and Puchnina (2019).

References

Aghajanyan, A., Maillard, J., Shrivastava, A., Diedrick, K., Haeger, M., Li, H., Mehdad, Y., Stoyanov, V., Kumar, A., Lewis, M., & Gupta, S. (2020). Conversational semantic parsing. https://arxiv.org/abs/2009.13655

Barkan, O., Caciularu, A., & Dagan, I. (2020). Within-between lexical relation classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*

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

Brunner, G., Liu, Y., Pascual, D., Richter, O., Ciaramita, M., & Wattenhofer, R. (2020). On identifiability in transformers. In *Inetrnational Conference on Learning Rerpresentations*

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., & Y Bengio, HS. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *55th Annual Meeting of the Association for Computational Linguistics*

Clark, K., Khandelwal, U., Levy, O., & Manning, CD. (2019). What does bert look at? an analysis of bert's attention. In *ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*

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

Ettinger, A. (2020). What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. In: Transactions of the Association for Computational Linguistics, Volume 8: 34–38

Gupta, S., Shah, R., Mohit, M., & Kumar, A. (2018). Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 volume 2: The 14th international conference on computational linguistics.*

Kassner, N., Krojer, B., & Schutze, H. (2020). Are pretrained language models symbolic reasoners over knowledge? In *24th Conference on Computational Natural Language Learning*

Kim, N., Patel, R., Poliak, A., Xia, P., Wang, A., McCoy, T., Tenney, I., Ross, A., Linzen, T., Durme, BV., Bowman, SR., & Pavlick, E. (2019). Probing what different nlp tasks teach machines about function word comprehension. In *Eighth Joint Conference on Lexical and Computational Semantics*

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

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

McCoy, RT., Pavlick, E., & Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In: ACL

Mickus, T., Paperno, D., Constant, M., & van Deemter, K. (2020). What do you mean, bert? assessing bert as a distributional semantics model. In *Proceedings of the Society for Computation in Linguistics* Vol. 3

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems, 26*.

Nakashole, N., Weikum, G., & 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., & Navigli, R. (2015). Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In M. Palmer, G. Boleda, P. Rosso (Eds.), *Proceedings of the fourth joint conference on lexical and computational semantics* (pp. 182–192). Denver, Colorado: Association for Computational Linguistics.

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., & Duchesnay, E. (2011). Machine learning in python. *The Journal of Machine Learning Research, 12*, 2825–2830.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Petroni, F., Rocktaschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., & Miller, A. (2019). Language models as knowledge bases? In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, PJ. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. In: Journal of Machine Learning Research (JMLR), 21: 5485–5551

Riedel, S., Yao, L., McCallum, A., & Marlin, M. B. (2013). NAACL-HLT: Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies* (pp. 74–84).

Roberts, A., Raffel, C., & Shazeer, N. (2020). How much knowledge can you pack into the parameters of a language model? In *Conference on Empirical Methods in Natural Language Processing*

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics, 8*, 842–866.

Roller, S., Kiela, D., & Nickel, M. (2018). Hearst patterns revisited: Automatic hypernym detection from large text corpora. arXiv preprint arXiv:1806.03191

Roussinov, D., & Puchnina, N. (2019). Combining neural networks and pattern matching for ontology mining-a meta learning inspired approach. In *Proceedings of the 13th IEEE International Conference on Semantic Computing (ICSC)*

Roussinov, D., & Turetken, O. (2007). Semantic verification in an online fact seeking environment. In *ACM Conference on Information and Knowledge Management*

Santus, E., Yung, F., Lenci, A., & Huang, CR. (2015). Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications, pp 64–69*

Santus, E., Lenci, A., Chiu, TS., Lu, Q., & Huang, CR. (2016). Nine features in a random forest to learn taxonomical semantic relations. arXiv preprint arXiv:1603.08702

Schlichtkrull, M., Kipf, T., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *The Semantic Web. ESWC 2018. Lecture Notes in Computer Science*

Sharoff, S. (2020). Finding next of kin: Cross-lingual embedding spaces for related languages. Journal of Natural Language Engineering 26

Shi, P., & Lin, J. (2020). Simple bert models for relation extraction and semantic role labeling. https://arxiv.org/pdf/1904.05255.pdf

Shwartz, V., & Dagan, I. (2016). Path-based vs. distributional information in recognizing lexical semantic relations. arXiv preprint arXiv:1608.05014

Shwartz, V., Goldberg, Y., & Dagan, I. (2016). Improving hypernymy detection with an integrated path-based and distributional method. arXiv preprint arXiv:1603.06076

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

Snow, R., Jurafsky, D., & Ng, A. Y. (2004). Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems, 17*.

Søgaard, A., Vulić, I., Ruder, S., & Faruqui, M. (2019). Cross-Lingual Word Embeddings. Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S. R., Das, D., & Pavlick, E. (2019). What do you learn from context? probing for sentence structure in contextualized word representations. arXiv preprint arXiv:1905.06316

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30.*

Vulić, I., Ponti, EM., Litschko, R., Glavaš, G., & Korhonen, A. (2020). Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics*, Online, (pp 7222–7240), https://doi.org/10.18653/v1/2020.emnlp-main.586,

Wang, C., He, X., & Zhou, A. (2019). Spherere: Distinguishing lexical relations with hyperspherical relation embeddings. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 1727–1737).

Wang, C., Qiu, M., Huang, J., & He, X. (2021). Keml: A knowledge-enriched meta-learning framework for lexical relation classification. In: AAAI

Washio, K., & Kato, T. (2018). Filling missing paths: Modeling co-occurrences of word pairs and dependency paths for recognizing lexical semantic relations. arXiv preprint arXiv:1809.03411

Zhou, X., Zhang, Y., Cui, L., & Huang, D. (2020). Evaluating commonsense in pre-trained language models. In *Proceedings of AAAI Conference*

Authors and Affiliations

Dmitri Roussinov[1] · **Serge Sharoff**[2] · **Nadezhda Puchnina**[3]

✉ Dmitri Roussinov
dmitri.roussinov@strath.ac.uk

Serge Sharoff
s.sharoff@leeds.ac.uk

Nadezhda Puchnina
np486061@tlu.ee

1 University of Strathclyde, Glasgow, Scotland, UK

2 University of Leeds, Leeds, England, UK

3 University of Tallinn, Tallinn, Estonia