1 Pre-trained Models for Natural Language Processing: A Survey

The authors of this paper list the two possible approaches to learning textual representations:

1. non-contextual word embeddings - Skip-Gram, GloVe, etc.

2. contextual word embeddings - CoVe, ELMo, OpenAI GPT, BERT, etc.

The non-contextual approach maps tokens (words or subwords) from a vocabulary to a matrix, where each of the vectors represent an embedding for the token, using a lookup table. This framework also contains a hyperparameter indicating the dimension of the token embeddings, allowing for more flexibility.

Limitations:

1. The embedding for a token is the same regardless of the context

2. Out of vocabulary tokens are randomly initialized

This contextual approach considers the context-dependent nature of words. Given a text, the representation of a token depends on all tokens.

Regarding the encoders used to acquire embeddings, there are two approaches. Sequence contextual encoders capture the local context of a word. The commonly used are convolutional models, which aggregate word embeddings by capturing local information, and recurrent models, which capture contextual representations using the nature of recurrent neural networks, often using a bidirectional approach. Non-sequence encoders learn representation using a tree or graph structure between words. The most important fully-connected graph model in practice is the fully-connected self-attention model, which uses a self-attention mechanism for computing the connections between words. Transformers fall exactly into this category.

Pretraining

Pretraining is the task of learning high-quality representation using a large-scale unlabeled corpora. This proved beneficial in several ways:

- It helps with downstream tasks
- It provides better model intialization and speeds up convergence
- It can be viewed as a form of regularization

The most important predecessor for BERT is the ULMFiT, which attempts to fine-tune a pretrained language model for text classification. Fine-tuning ULMFiT consists of: pretraining the language model on the domain, fine-tuning it on the target data, and finally, fine-tuning it on the target task. This approach paved the way towards the mainstream approach to adapting pretrained language models for downstream tasks.

The pretraining tasks can be divided into three categories:

1. Supervised learning - inputs and labels are known

2. Unsupervised learning - only inputs are known

3. Self-supervised learning - inputs are known, but labels are known for only part of the data

The most important and widely used approach is self-supervised learning.

Some approaches to pretraining are:

- Language modeling - A common unsupervised task, where given a text sequence we estimate the join probability. It can be trained using maximum likelihood estimation. The biggest drawback is it uses only leftward context tokens, but a bidirectional approach is also possible

- Masked language modeling - Masks some tokens in the inputs sentences and predicts them using all other tokens. A problem that occurs is that the mask token does not appear during fine-tuning, which is why the method was modified. To tackle this, a mask token is used 80% of the time, a random token 10% of the time and the original token 10% of the time.

- Sequence-to-sequence MLM - Uses an encoder-decoder architecture, where the encoder takes in the masked sequence, and the decoder produces the masked tokens. It proved to be beneficial for Sequence-to-sequence downstream tasks.

- Enhanced masked language modeling

  - RoBERTa - uses dynamic masking where the masked token in each sentence changes for every epoch
  - UniLM - uses unidirectional, bidirectional and sequence-to-sequence language modeling tasks
  - StructBERT - extends BERT's original MLM pretraining objective by pre-shuffling trigrams after masking, while the model tries to predict the original tokens from this input and extends NSP by predicting the previous sentence as well

- Permutated language modeling - permutes the input sequence, chooses and masks some tokens (usually the last last few) as targets, while the model predicts masked tokens in an autoregressive fashion
- Denoising autoencoder - partially corrupts the text and uses a sequence-to-sequence model to reconstruct the original text
  - Token masking - randomly samples tokens and replaces them with [MASK] tokens
  - Token deletion - randomly deletes tokens from the text, while the model predicts the positions of missing inputs
  - Text infilling - randomly chooses a span of tokens and replaces it with a [MASK] token, while the model predicts how many tokens were replaced
  - Sentence permutation - divides the document in sentences and randomly shuffles them
  - Document rotation - randomly selects a token and rotates the document so it becomes the starting token, while the model predicts the original starting token position
- Constrastive learning - minimizes a score function represented by a neural encoder that incorporates three pieces of text: a text x, a similar text y^+ and a dissimilar text y^-
  - Deep InfoMax - assumes a span of the sequence is more similar to the same sequence with this span masked than to a random n-gram from the corpus
  - Replaced token detection - given a surrounding context, the model predicts whether a token is replaced (e.g. ELECTRA extends this approach)
  - Next sentence prediction - trains the model to distinguish does a sentence appear after the previous sentence
  - Sentence order prediction - uses consecutive sentences as positive examples, and consecutive sentences in reversed order as negative examples

Contextual embeddings in models such as BERT contain linguistic knowledge and world knowledge. This mean the latter might prove useful in the task of claim detection, especially combined with task adaptive pretraining to possibly obtain even more world knowledge.

Tuning strategies

Most commonly used tuning strategies are feature extraction (pretrained parameters are frozen) and fine-tuning (pretrained parameters are not frozen). Some other non-exclusive fine-tuning strategies include:

- Two-stage fine-tuning - uses fine-tuning on an intermediate task before fine-tuning on the target task
- Multi-task fine-tuning - simultaneously fine-tunes on intermediate and target tasks
- Fine-tuning with adaption modules - adds new modules to an existing model and only fine-tunes those parameters
- Gradual unfreezing - gradually unfreezes layers of the model starting from the top layer
- Sequential unfreezing - unfreezes the classification/regression head and fine-tunes the model, then does the same for the hidden layers, and the same for the embeddings as well

2 Don't Stop Pretraining

The paper asks two interesting questions:

- Is the distribution over language characterizing a given topic or genre even relevant?
- Is it it useful to use separate pretrained models for specific domains?

To determine this, they approach the problem using four domain-specific texts: computer science publications, biomedical science publications, news and reviews. They also use eight different classification tasks (two per domain). The considered approaches were *domain-adaptive pretraining* (DAPT) where continued pretraining is done on the textual domain, and *task-adaptive pretraining* (TAPT) where the authors use the unlabeled task dataset. In low resource cases, they use *manually selected* unlabeled data for pretraining and propose ways to automate this procedure.

As a baseline, they fine-tune the RoBERTa model. The results show DAPT outperforms the baseline when using a relevant textual domain, even in low-resource settings. This occurrence suggests the importance of pretraining on domain-relevant data. However, pretraining on domain irrelevant data shows inconclusive results, helping in some cases, while being detrimental in other cases. TAPT is less expensive to run than DAPT because of using smaller corpora. In comparison the baseline, TAPT gives much better results, sometimes even as good as DAPT. The best approach is using both DAPT and TAPT. A potential problem with this paper is they only employ RoBERTa, which is pretrained exclusively using masked language modeling.
Important links:

1. code - `https://github.com/allenai/dont-stop-pretraining`

3 Frustratingly Simple Pretraining Alternatives to Masked Language Modeling

The authors of this paper propose modifications of pretraining tasks for large language models.

- Shuffled word detection is the first of five modifications. Its' idea is to shuffle 15% of tokens, while the model tries to predict which tokens are shuffled. This is simpler than the ELECTRA approach because it doesn't require a generator network.
- Random word detection is the second proposed approach. It replaces 15% of tokens in the input sequence with random words.
- The third combines the previous two tasks by creating a multi-class (shuffled, random, original) token-level classification task. The percentages of altered tokens are also lowered to 10% for each of the modifications.
- The fourth approach is masked token type classification. This is also a multi-class token-level objective, where the possible classes are stop words, digits, punctuation marks and content words. As in aforementioned tasks, 15% of the tokens is masked.
- The final method is masked first character prediction. This is a simplified version of the masked language modeling task, where the model has to distinguish the first character of the masked tokens. This is computationally less intensive because it is a 29-way classification task. The same percentage of tokens is masked as in the previously mentioned task.

For around half of the tasks the best performing technique is vanilla masked language modeling, while for the other half the third approach (shuffle + random) achieves the best results. Interestingly, a big improvement occurs in semantic textual similarity, paraphrase detection and duplicate question detection which are similarity tasks and require fine-grained understanding, as does claim-checkworthiness detection.

4 Claim Check-Worthiness Detection as Positive Unlabelled Learning

The paper takes into account the abundance and ease of acquiring unlabeled data for learning claim-checkworthiness detection. The authors propose positive unlabeled learning given just positive labeled and unlabeled examples. Their approach consists of a positive-unlabelled classifier which creates sample weights. The sample weights for positively labeled examples are constant, while the weights for the unlabeled examples are first fed into a classifier f which produces sample weights as its' output. The most positive sample weights for unlabeled examples are treated as if the example truly is positive. A second classifier g is then trained using these examples with generated and original labels.

Besides employing positive unlabeled learning, the authors use positive unlabeled conversion (PUC) which relabels unambiguously positive examples. They use a smart heuristic to acquire data in a cheap manner by training on statements from Wikipedia citation needed featured articles labeled as containing a citation (positive) and unlabeled. After training a classifier on this data, they enforce PUC to obtain a second classifier.

They train a final classifier for claim-checkworthiness detection using the weights of the second classifier. They repeat the same PUC procedure with the target dataset and use these examples in the final classifier for claim-checkworthiness detection.

As a baseline they use an off-the-shelf BERT classifier. The results are inconclusive, since PU and PUC result in increases for recall and F1 score while being detrimental to precision. Overall, the models that utilize PU or PUC have lower variance. They also experiment with transferring knowledge to a dataset consisting of political speeches, but the PU and PUC approaches don't show any benefits in comparison to a vanilla BERT trained on the aforementioned target dataset.

Important links:

1. code - `https://github.com/copenlu/check-worthiness-pu-learning`

5 Language Models as Fact Checkers?

The authors propose a new pipeline for fact-checking. They use masking, a language model (BERT) and a verification classifier. This approach is very different compared to the traditional one. The authors mask the last named entity in the claim using spaCy's model, because the knowledge about named entities may give the classifier an edge in comparison to other classifiers. Otherwise, the masked word may be a stop word which does not tell the model much. They use the original claim and outputs of the LM which is fed the masked claim, after which they predict the entailment.

As baselines, they introduce a BERT model with the encoder frozen, and a BERT with no frozen parameters. The results show their model outperforms the frozen baseline, but performs worse than a fine-tuned BERT.

6 A New Approach to Claim Check-Worthiness Prediction and Claim Verification

This paper uses a more traditional machine learning approach for the task of claim-checkworthiness. The authors try out two different approaches: supervised and unsupervised. For the supervised approach, they employ a modified TF-IDF model to establish a relationship of statements and background articles, combined with a gradient boosting classifier. The unsupervised approach consists of extracting check-worthy claims without labels, which would help in creating new datasets for claim-checkworthiness detection from a corpus. The features they extract for this approach answer the following questions:

- Does a claim have a verb in continuous tense and a proper noun?
- Does a claim use someone else's statement in indirect speech with a connective word (e.g. "which")?

They also engage in explanation extraction, where they use POS tagging on claims and POS tagging on the articles from the dataset. Then, they compute the cosine similarity between the POS tags for the claim and the POS tags for the possible explanations and use the maximum as the true explanation.

Finally, to determine truthfulness detection a sentiment score is employed on each sentence, and the *compound score* is calculated using these sentiment scores.

The achieved results for claim-checkworthiness arise suspicion due to each metric being above 98%. The claim verification results are more sound, but the authors didn't compare any model to a baseline.

7 IMHO Fine-Tuning Improves Claim Detection

This paper presents a clever way of collecting data for the task of claim detection, which is similar to claim-checkworthiness detection. They mine comments from Reddit containing the acronyms IMO (in my opinion) and IMHO (in my humble opinion). Besides this dataset, they consider 4 other labeled datasets. The problem that arises with this dataset is that it does not contain any non-claims, which they solve by using a language model fine-tuning approach. They implement ULMFiT model, but modify the step of fine-tuning the LM on task-specific data by using the IMO/IMHO claims.

The achieved results show that IMHO LM fine-tuning is superior to task-specific LM fine-tuning and to a CNN they used for comparison on every metric and each of the four datasets. They used a 10-fold cross-validation setting and rerun each model 10 times which makes the results even more convincing.

8 It Takes Nine to Smell a Rat: Neural Multi-Task Learning for Check-Worthiness Prediction

The authors of this paper use a political debates dataset and a multi-task learning approach. The way they approach multi-task learning is by taking into account different sources of annotation over the same training dataset and treating them as different tasks. Their proposed model has ten task-specific layers and each of them corresponds to one fact-checking organization and their annotations.

They use different features as inputs:

- language features
- discourse features
- length, size, position
- named entities
- embeddings
- similarity to checked claims
- metadata features

The setup used is updating the parameters of the whole model. Four fold cross-validation was employed where one of the four debates was left out in each fold and each experiment was repeated three times with random seeds to handle variance.

The results imply that multi-task learning benefits performance and outperforms the single-task baseline model. Even for all nine fact-checking sources evaluation results show improvements, but vary from organization to organization.