ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Bachelor Thesis
in
Computer Science

Reformulating Named Entity Recognition

Vasilis Vythoulkas

Supervisors: Ion Androutsopoulos
Prodromos Malakasiotis

March 2022

Vasilis Vythoulkas

Reformulating Named Entity Recognition

March 2022

Supervisors: Ion Androutsopoulos, Prodromos Malakasiotis

Athens University of Economics and Business

School of Information Sciences and Technology

Department of Informatics

Natural Language Processing Group

Athens, Greece

Abstract

In this thesis, we study the task of Named Entity Recognition (NER) which has multiple applications in practice. First, we reformulate Named Entity Recognition as a Question Answering problem (QA). We apply it in few-shot learning which diminishes the necessity of data labelling, making it very appealing in practice. In many cases, in which we applied few-shot learning, we achieve performance comparable to state-of-the-art methods. Nevertheless, NER as QA performs worse compared to other methods in the literature when trained on a lot of training data. Secondly, another issue we focus on is robustness of NER. Existing methods for NER are biased and not robust to adversarial attacks. To address these issues, we propose using an auxiliary task which aims to predict the named entity types based only on the context. The model is trained in the auxiliary task concurrently with the standard NER task in RockNER, an adversarial dataset. Nevertheless, no substantial improvements were observed in the standard NER task, whereas performance in the auxiliary task was reasonably high.

Περίληψη

Η αναγνώριση ονομάτων οντοτήτων (Named Entity Recognition) αποτελεί ένα σημαντικό πρόβλημα στον τομέα της επεξεργασίας φυσικής γλώσσας (ΕΦΓ, Natural Language Processing, NLP), καθώς διαθέτει πολλές εφαρμογές στην πράξη. Αρχικά, μετασχηματίζουμε το πρόβλημα της αναγνώρισης ονομάτων οντοτήτων σε πρόβλημα απάντησης ερωτήσεων (Question Answering). Εφαρμόσαμε την μέθοδό μας έχοντας διαθέσιμα λίγα παραδείγματα εκπαίδευσης (few shot learning). Η μάθηση με λίγα παραδείγματα εκπαίδευσης παρουσιάζει ιδιαίτερο ενδιαφέρον, διότι δεν απαιτεί το κόστος της συλλογής πολλών δεδομένων και της ανάθεσης ετικετών σε αυτά. Σε πολλές περιπτώσεις όταν είχαμε λίγα παραδείγματα εκπαίδευσης, η μέθοδός μας ήταν εξίσου επιτυχημένη με τις καλύτερες μεθόδους που υπάρχουν στη βιβλιογραφία. Ωστόσο, η μέθοδός μας όταν διαθέτει πολλά παραδείγματα εκπαίδευσης δεν είναι εξίσου επιτυχημένη με άλλες μεθόδους που υπάρχουν στη βιβλιογραφία. Στη συνέχεια, ασχολούμαστε με το ζήτημα της ανθεκτικότητας (robustness) στην αναγνώριση ονομάτων οντοτήτων. Η παλαιότερη βιβλιογραφία δείχνει ότι τα μοντέλα δεν είναι ανθεκτικά κατά την αναγνώριση ονομάτων οντοτήτων, με αποτέλεσμα να έχουν και προκαταλήψεις (bias). Για να περιορίσουμε αυτά τα προβλήματα, προτείνουμε τη χρήση ενός βοηθητικού προβλήματος (task), που στοχεύει στο να προβλέπει τις κατηγορίες των ονομάτων οντοτήτων αποκλειστικά από τα συμφραζόμενα. Το μοντέλο εκπαιδεύεται ταυτόχρονα στο κυρίως πρόβλημα (αναγνώριση ονομάτων οντοτήτων) και στο βοηθητικό πρόβλημα (πρόβλεψη κατηγοριών οντοτήτων από τα συμφραζόμενα). Ωστόσο, παρατηρούμε ότι η προσθήκη του βοηθητικού προβλήματος δεν βελτιώνει τις επιδόσεις στο κυρίως πρόβλημα, παρ' όλο που οι επιδόσεις στο βοηθητικό πρόβλημα είναι σχετικά υψηλές.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Ion Androutsopoulos for his continuous support and the opportunity he gave me to work in a very interesting research area. His constructive comments helped me throughout this thesis. Secondly, I would like to thank my co-supervisor postdoctoral researcher Prodromos Malakasiotis for the valuable guidance he provided me.

Contents

1 Introduction

1.1 What are the problems we solve?

In this thesis we worked on Named Entity Recognition (NER). NER is a subfield of information extraction that aims to find and classify named entities present in some input text [44].

First, we give an overview of a new paradigm in NLP, prompting, which reformulates NLP tasks into language modeling (LM) tasks.

Secondly, we reformulate NER as a Question Answering (QA) problem by prompting QA models. We apply it mainly in few-shot scenarios where limited training data are available.

Finally, we propose an auxiliary task hoping that it will help NER models make better decisions using context. We apply the auxiliary task concurrently with the NER task in a fully supervised setting on adversarial datasets. The aim of the adversarial datasets is to check the robustness of NER models against changes in the context and against replacements of the named entities.

1.2 Why are they important?

NER has been used in multiple applications, from improving customer support[1] to extracting Protein–Protein Interactions [8].

Prompting has gained much importance recently due to its success in few-shot learning [5, 47]. In many real-world problems we observe a scarcity of available examples and labelling more data requires significant time and resources or might even be unattainable. Therefore, there is an urgent need to learn exclusively from few labelled samples. The use of few-shot learning is really appealing in this respect, because it can diminish the necessity of data collection and labelling and, thus, save time and resources, which are necessary in order to create efficient machine learning models.

[1] https://www.allerin.com/blog/exploring-named-entity-recognition-use-cases-across-industries

Previous work has shown that the main factor influencing NER decisions is the named entities themselves, while context is used in a much smaller degree [2]. This leads to a lack of robustness and bias problems [1, 36]. By making better use of context, these problems may be significantly reduced.

1.3 What has been done until now and what's our contribution?

Reformulating NER as QA already exists as a concept in the literature. Previous work first fine-tuned NER as QA in a fully supervised setting and then tried zero-shot [24] or few-shot evaluation [61] on a different dataset. Apart from fine-tuning in a fully supervised setting and trying few-shot on a different dataset, we apply downsampling for some specific or all named entity types of a dataset.

Regarding the auxiliary task, previous work did an interpretability analysis for NER and tried to predict the types of the named entities using only the context exclusively during inference [2]. In contrast, we train the model to predict the types of the named entities using only the context.

The main findings of this thesis are:

- Reformulating NER as QA performs comparably to state-of-the-art methods for few-shot learning.
- Reformulating NER as QA performs worse than other results in the literature when trained in a fully supervised setting.
- Regarding adversarial datasets, no substantial improvements were observed in the NER task, whereas performance in the auxiliary task was reasonably high.

1.4 Thesis Structure

The remaining of the thesis is organized as follows:

Chapter 2

In chapter 2, we survey prominent prompting approaches.

Chapter 3

In chapter 3, we describe our NER as QA method, the datasets which were used and our experimental results.

Chapter 4

In chapter 4, we describe our auxiliary task, the dataset we used and the results of this method.

Chapter 5

Chapter 5 concludes and proposes directions for future work.

2 Background

2.1 Background

Recently, much attention has been drawn to prompting. Below, we give an overview of this new paradigm in NLP. Prompting converts downstream tasks to LM tasks. The prompt is created using a template T that adds text including a [MASK] token to an input text x. We also need to create a verbalizer that maps labels to words from the model's vocabulary.

For instance, suppose a downstream task is binary sentiment analysis and an input sentence is x = 'This song is my favourite music of all time.'. We can define a template like '[X] It is [MASK]'. Then after filling slot [X] with input text x, the prompt x_{prompt} turns into "This song is my favourite music of all time. It is [MASK]". What is fed into the language model is the prompt x_{prompt}. We can use label words such as "good" and "bad" which correspond to the labels positive and negative respectively. The model must predict label words, as the most probable filler of the [MASK] slot.

Prompting has seen a great success in few-shot learning. The most famous prompting papers are GPT3 [5] and PET [47]. GPT-3 is an autoregressive language model with 175 billion parameters, which achieved remarkable results without tuning the LM parameters. The PET paper used a much smaller LM (millions of parameters) and suggested to fine-tune (further train on end-task data) the LM while simultaneously using ensemble learning with many dinstict prompts and semi-supervised learning. For a detailed survey you can also read [28].

2.2 Manual Prompt

The simplest way to define a prompt is to manually write templates based on human intuition about the task that we want to solve. However, it is non-intuitive for some tasks and the models are highly sensitive to the context making hand-written templates sub-optimal [17, 46]. As a result, we measure a lower bound of what language models "know".

In order to deal with these issues, a variety of methods have been recommended to automate the template creation process.

2.3 Discrete Prompt

Some authors suggest searching in the discrete space for templates. Below, we provide an overview of some basic methods that were proposed recently.

A simple approach is to use a thesaurus to find synonyms of template words [58]. Round trip translation (also known as back-translation) can also be used to translate an initial manual template to another language and then translate back to the original language in order to generate a paraphrased template with probability $P_{forward}(b \mid a) * P_{backward}(c \mid b)$, where a is the initial template, b is the translated template in another language, c is the paraphrased template in the original language [17].

Other work trains a masked language model (rewriter) which uses a hand-crafted prompt as an input and then replaces the entire input sequence with the closest tokens in the embedding space. Then, a different frozen (not fine-tuned) BERT model takes these new tokens as an input and makes predictions [13]. This method is called "BERTese", you can see Figure 2.1 for an example. A difference from the previous approaches is that "BERTese" first applies the template to the input and then it paraphrases producing a separate paraphrase for each downstream task sentence.

Some authors use the sequence to sequence T5 pre-trained model [40] to generate template words [10]. They add a single sentinel token (e.g. <X>) for the prompt to the input of the model and let T5 fill it. Thus, they just define the label words and the place where the additional text will be positioned in the input to the model. They don't have to pre-define the length of the additional tokens which is required for masked language models like BERT. This is because T5 has been pre-trained to replace each sentinel token by one or more tokens. An example is given in Figure 2.2.

Other work [4] uses a domain adaptation algorithm that has a two-step multi-task mechanism. First, they train T5 to generate the domain name and some predefined domain related words for each input and then the task label is predicted. For any unseen domain during training, T5 generates a training domain name and some words from training domains. Then, these are used to create a prompt for a classification task. An example is given in Figure 2.3.

Factual probing task aims to measure factual information present in LMs. Prior research scrapes Wikipedia to find templates given subjects and objects of a factual probing task

(LAMA [38]). They use frequent middle words or dependency paths between the subject and the object as a template, because they often indicate the relationship between the subject and the object [17].

Some researchers insert "trigger" tokens in the input and initialize them as [MASK] tokens. Then at each step, they use the equation $\mathcal{V}_{\text{cand}} = \underset{w \in \mathcal{V}}{\text{top-}k} \left[{w_{\text{in}}}^T \nabla \log p(y|x_{\text{prompt}}) \right]$, where the gradient is taken with respect to the input embedding of $x_{\text{trig}}^{(j)}$ and w_{in} is the input embedding of vocabulary word w. With this equation they find a candidate set of words that are estimated to give the highest increase in probability. Then, the current trigger token is replaced by the word of the candidate set that maximizes the probability $p(y|x_{\text{prompt}})$. A drawback of this method [49] is that the "trigger tokens" are not always interpretable.

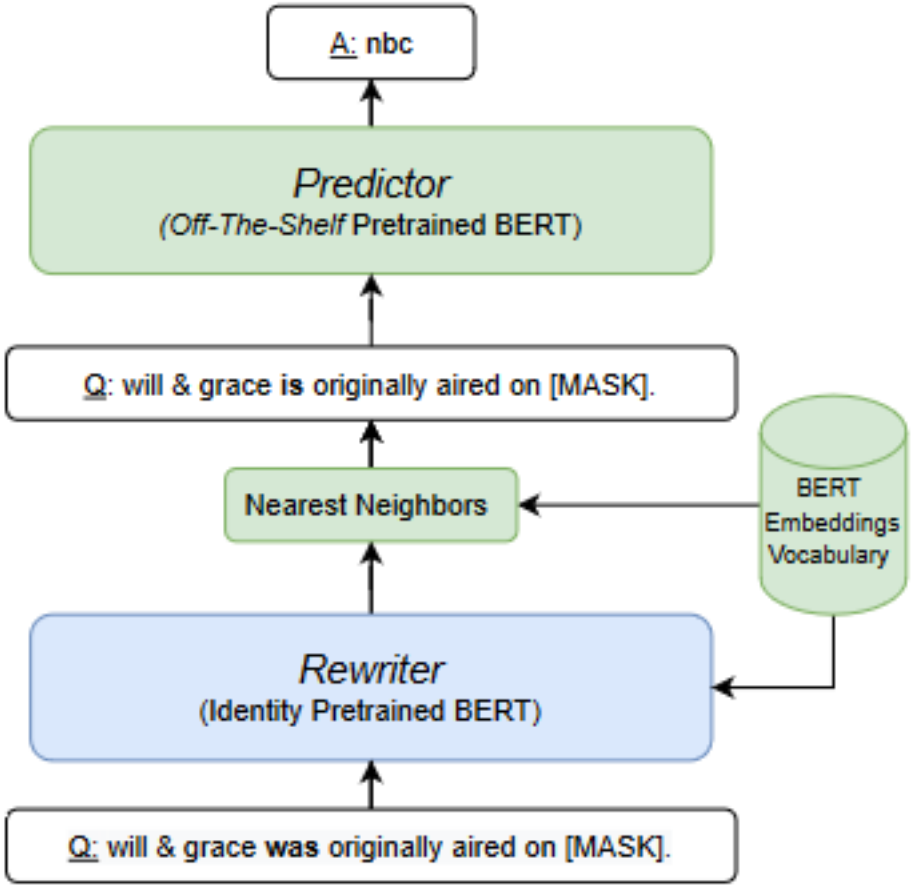

Fig. 2.1: Image from the BERTese paper [13].

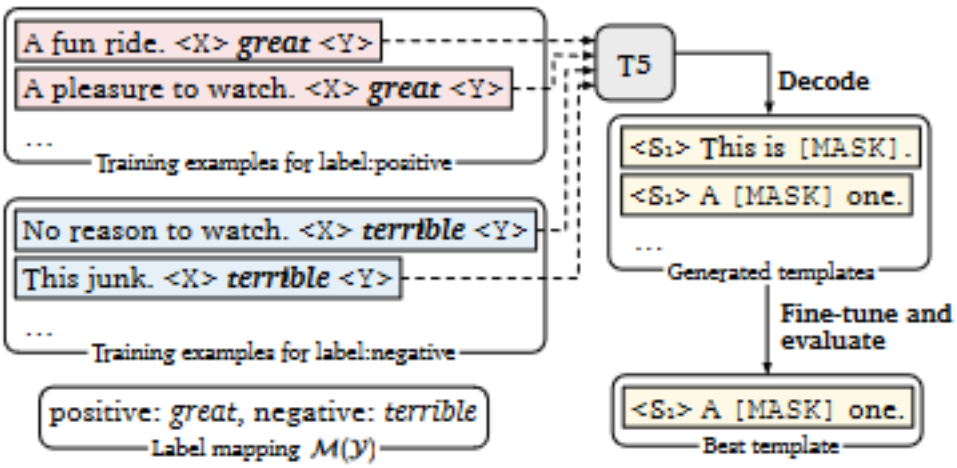

Fig. 2.2: Image from the Making Pre-trained Language Models Better Few-shot Learners paper [10].

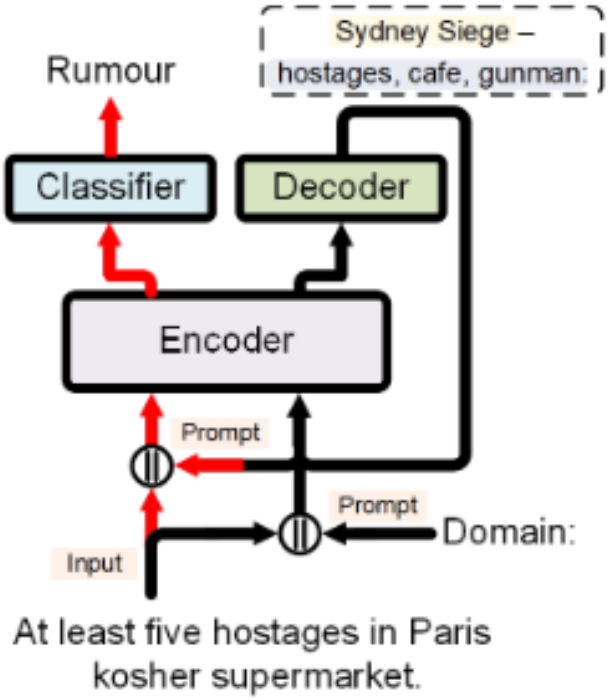

Fig. 2.3: Image from the PADA paper [4]. Sydney-siege is the domain name and hostages, cafe, gunman are domain related words.

2.4 Continuous Prompt

The previous methods constrain the template words to be vocabulary words. Next, we describe some methods that don't have this constraint and search in the continuous embeddings space for the proper template words. One approach inserts some virtual tokens into the input of LM and learns the word embeddings of them while keeping the LM parameters frozen [12, 21, 60]. These embeddings can be initialized randomly, with words from vocabulary or for classification tasks with the embeddings of labels words (a.k.a. verbalizer). Some authors have found that, if models exceed billions of parameters, continuous prompt can perform equally well with updating all LM parameters [21]. This success may be attributed to the large depth of the large language models. Previous works have shown that the deeper the network is, the more expressive the first layers are [32, 50]. Regarding the interpretability of continuous prompts, prior research has found that even when we add an auxiliary loss term that maps continuous prompts to arbitrary or irrelevant natural language text, the results are similarly successful to the best continuous prompts [19]. As a result, they suggest not to interpret continuous prompts with nearest neighbour projection.

Prefix-tuning adds a prefix to the input in order to augment the context. This method optimizes all layers of the prefix while keeping the rest of the LM frozen [23]. Overall, a difference of continuous prompts compared to Adapters [15] is that the latter are added between layers. In summary, continuous prompt is both a template construction method and a light-weight parameter update method.

2.5 Tuning free prompting

Another parameter update method is sometimes called "Tuning free prompting". It keeps all LM parameters frozen and predicts using only the help of additional text, which is added in the input. This approach seems very appealing, because it is extremely memory efficient, as we can use the same model for multiple tasks. Moreover, zero-shot learning can be used and catastrophic forgetting [43] cannot occur. Nevertheless, models may be very sensitive to the wordings of the template text which is added in the input. One application of this approach is factual probing which is also considered one of the first applications of prompting in the literature [38].

Frequently, it is used for few-shot learning by adding at the beginning of the sentence some priming examples (a.k.a demonstrations) as context to the evaluation example. One example is shown in Figure 2.4. "In-context learning" or "priming" refers to this combination of demonstrations and tuning free prompting. "In-context learning" became known with the GPT-3 paper [5] whose largest model performed well on a wide variety of tasks. By

employing "in-context learning" in a natural language interface even inexperienced users could create NLP systems. A drawback of "in-context learning" is that it is mostly successful when applied to very large scale models (billions of parameters).

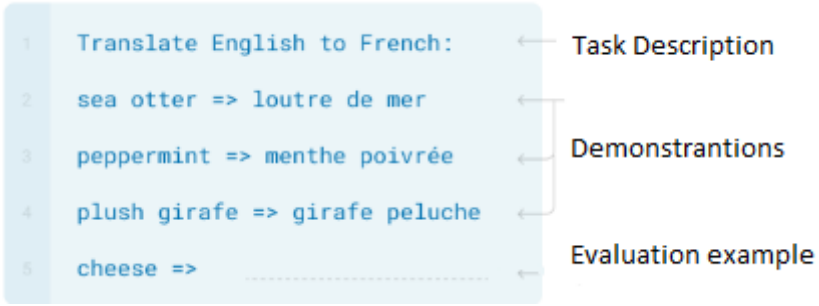

Fig. 2.4: Image from GPT-3 paper.

There are several ways to select which demonstrations are added in the input. The simplest approach is to follow the GPT-3 paper and randomly sample examples from the training set in order to concatenate them with the input. However, this approach has several disadvantages. Some scholars suggest that "in-context learning" in GPT-3 is susceptible to majority label bias, i.e., a type of bias which favours label words that are common in the demonstrations [59]. For example, when performing the binary classification task and providing only demonstrations with positive label, GPT-3 is extremely biased towards predicting that class. They also found that GPT-3 suffers from recency bias which means that it usually predicts answers which are identical to the labels of the most recent demonstrations. Furthermore, recency bias prevails over majority label bias. For instance, in the binary classification task, when performing 4-shot learning where only the last demonstration has a negative label, GPT-3 is biased towards predicting the negative class. Moreover, GPT-3 is prone to common token bias, i.e., it favours outputting tokens which can also be identified in its pre-training distribution [59]. To mitigate these problems it is proposed to first calculate probabilities for a context-free input (e.g. Input: Subpar acting. Sentiment: Negative Input: Beautiful film. Sentiment: Positive Input: N/A. Sentiment:). Then they use the real input (e.g. the prompt would be "Input: Subpar acting. Sentiment: Negative Input: Beautiful film. Sentiment: Positive Input: Hilarious. Sentiment:") to get the probability distribution P. To make final predictions they multiply element-wise P with the inverse of context-free probabilities. Entropy-based methods can also be used to rank each prompt ordering and find the best order of demonstrations [31]. Another approach for choosing which demonstrations to add in the input, is to use sentence embeddings to find training examples that are similar (in terms of sentence embedding similarity) to the test example [10, 26] .

Some recent works suggest, before applying "in-context learning" for some task, to concurrently train the model on a collection of other tasks, in which they have added demonstrations [35, 54]. A similar approach is to fine-tune on a large collection of tasks using prompting and then do zero-shot learning on different tasks [45, 54]. Using multiple

prompts per dataset has been found to generalize better to unseen tasks [45]. This approach matched GPT-3 “in-context learning” performance for several tasks despite being 16 times smaller.

2.6 Prompt-based fine-tuning

Prompt-based fine-tuning uses prompts, but also updates the parameters of the LM. This approach is primarily applied for LM which have only millions of parameters in few-shot scenarios, because it is efficient to train such language models in contrast to larger ones. Prompt-based fine-tuning has been particularly successful for classification tasks that are easily reformulated in a LM format [10, 46, 47]. This approach leads to large improvements mainly in low resource settings compared to the standard fine-tuning paradigm without prompts, that was used by papers like RoBERTa [29] and BERT [9] which use a fine-tune head. Nevertheless, a disadvantage of prompt-based fine-tuning compared to tuning-free prompting is that we usually need to keep a different model for each task increasing memory requirements, except for the case where multi-task learning is applied.

2.7 Why prompting works?

According to the dominant view, the effectiveness of prompts can be attributed to their ability to act as instructions for downstream tasks [5, 54]. Prior work found that when using prompt-based fine-tuning, just adding a [MASK] token to the input (a.k.a NULL prompt) performs comparably to manually written templates [30]. They believe that the success of prompting is partly because of predicting on a [MASK] token with a pre-trained MLM head. Another work [53] does prompt-based fine-tuning for the task of natural language inference and finds no statistically significant difference between relevant and irrelevant templates at any number of few-shot learning training instances. For instance, a relevant template they tried is, ‘[premise] Should we assume that “[hypothesis]” is true? [MASK]’ and an irrelevant template they tried is, ‘[premise] Single-family zoning is bad for American cities.“[hypothesis]”? [MASK]’. Nevertheless, they empirically show that misleading templates, such as ‘[premise] Does that have the same meaning as "[hypothesis]"? [MASK]’, learn slower than instructive templates. They claim that gains from prompting are a result of better pattern matching and not of learning from instructions.

3 NER as QA

3.1 Introduction

In this chapter we reformulate NER as a Question Answering (QA) problem. We apply NER as QA in both fully supervised setting and few-shot scenarios.

3.2 Related Work

The application of prompting for NER has already drawn the attention of several researchers. Some researchers use BART [22] and enumerate all possible n-grams which are applied to pre-determined templates [7]. They use templates like '[candidate_span] is a [entity_type] entity'.

Another work proposes "LightNER" [6] which uses the idea of soft-prompting by adding key-value pairs at every layer l of Transformer while updating only prompt parameters. Formally, they use the following equation:

$$\text{Attention}^l = \text{softmax}(\frac{\boldsymbol{Q}^l[\boldsymbol{K}^l; \boldsymbol{\phi}_k^l]^T}{\sqrt{d}})[\boldsymbol{V}; \boldsymbol{\phi}\_v^l]. \tag{3.1}$$

"LightNER" also learns weights to leverage different label words of each named entity. Template free-prompt tuning [22] uses BERT [9] and creates a verbalizer by using wikidata Knowledge-base. They explored mapping labels to frequent words in the knowledge base, to words based on the LM output distribution or to virtual label words (continuous vectors).

The last years, we have observed a trend of transforming NLP tasks to question answering (QA). Some work transformed multiple NLP tasks into question answering [34]. For instance, for summarization they asked the question "What is the summary?". Regarding reformulating NER as QA, some researchers first fine-tuned in a fully supervised setting and then they tried zero-shot [24] or few-shot evaluation [61] on a different dataset.

3.3 Datasets

- The CoNLL03 [51] dataset is one of the most widely used datasets for the task of Named Entity Recognition. We use the English version which contains news stories from Reuters in 1996. It contains 4 named entity types as shown in Figure 3.1. Miscellaneous entity refers to an entity that is not a person, an organization or a location.

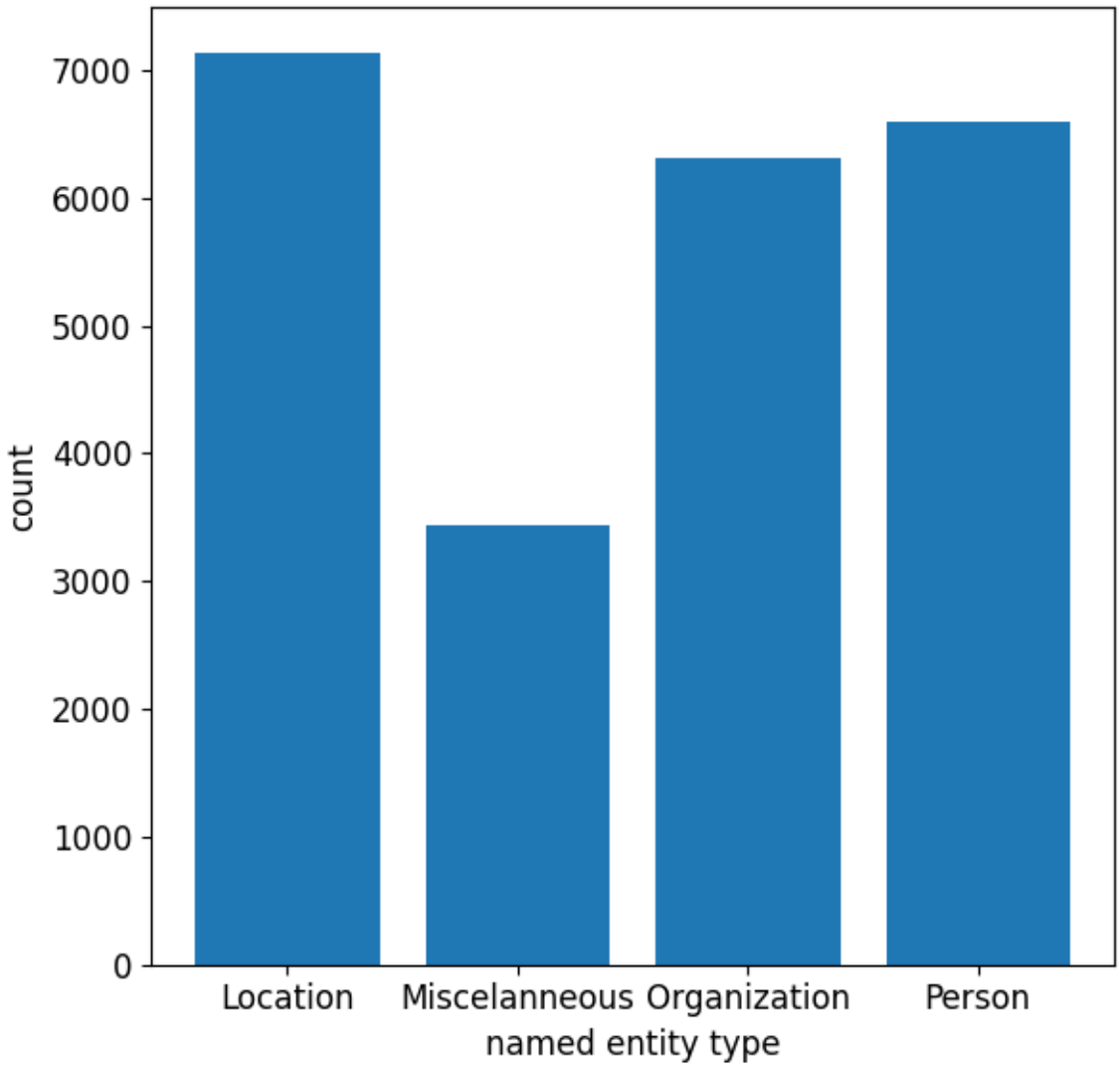

Fig. 3.1: CoNLL03 training distribution.

- MIT-Restaurant [27] dataset consists of spoken queries on restaurant searching and booking. It contains named entity types such as location, price and restaurant names. The test distribution of named entity types is shown in Figure 3.2.

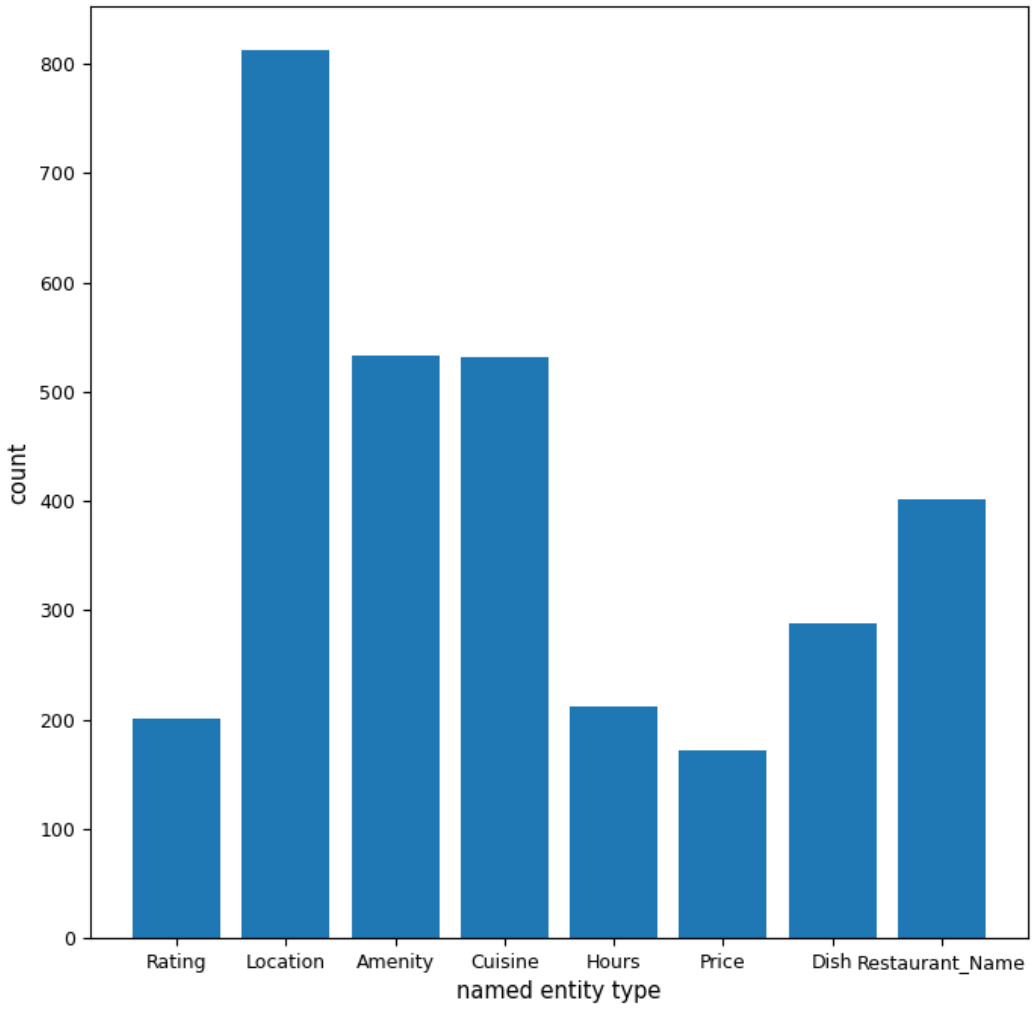

Fig. 3.2: MIT test distribution.

3.4 Model

Our method reformulates Named Entity Recognition as a question answering problem. We create templates such as "Which [E] is present in the text? \n [X]", where [E] is a slot for named entity types like person, location etc. and [X] is an input slot for the input. Symbol "\n" helps to distinguish the question from the input. As an example consider the sentence x="Rare Hendrix song sells for almost $17000" (Figure 3.3). For the person entity type we have a template of the form "Which Person is present in the text? \n [X]", where [X] is the input slot. After applying the template to x we get the x_{prompt} = "Which Person is present in the text? \n Rare Hendrix song sells for almost $17000". The named entities generated are enclosed in brackets, because we use a text-to-text model and we need to distinguish different named entities of the same named entity type present in a sentence.

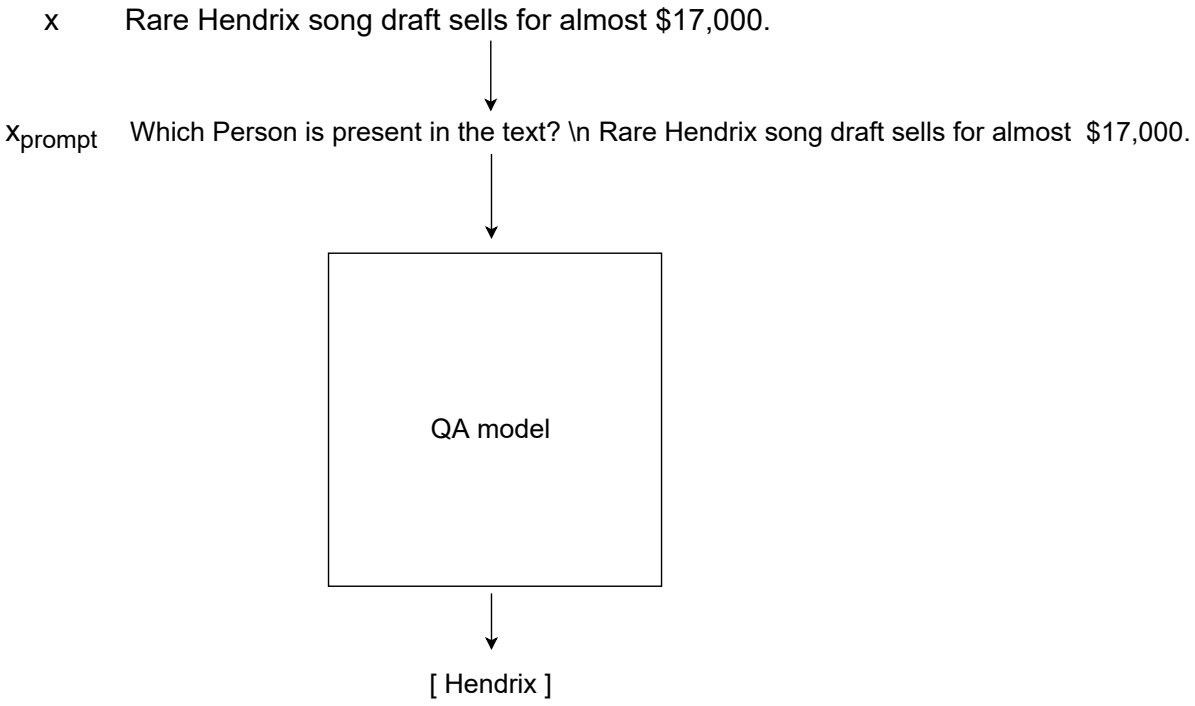

Fig. 3.3: Our NER as QA model.

The model we used is Unified-QA [18] which is an instance of T5 already trained on a variety of QA tasks, including extractive QA datasets like SQuAD [42] and SQuAD2.0 [41]. For this reason, we expect good few-shot NER performance. We note that the Unified-QA-v2-1363200 base variant of Unified-QA is used. A drawback of utilizing Unified-QA as our question answering model is that we need to learn the beginning and the end identifier of an entity in order to distinguish multiple different named entities present in a sentence. We enclosed named entities inside brackets. During inference, we ask an input sentence a different question for every named entity type, which is not scalable for big datasets with many named entity types.

Briefly, we describe below a high level formulation of the forward pass. Given an input sentence x we apply a template and we get x_{prompt} which is fed into Unified-QA encoder producing the following:

$$\mathbf{h}^{enc} = \textsc{Encoder}(x_{prompt}) \tag{3.2}$$

where $\mathbf{h}^{enc} \in \mathcal{R}^{n \times d}$ and d is the hidden state dimension.

At each timestep t of the decoder, we get a vector

$$\mathbf{h}_t^{dec} = \textsc{Decoder}(\mathbf{h}^{enc}, o_{1:t-1}) \tag{3.3}$$

where $\mathbf{h}_t^{dec} \in \mathcal{R}^d$ and d is the hidden state dimension and $o_{1:t-1}$ are the previous output words of the model (at inference time) or the ground truth previous output words (during training).

The conditional probability of the correct output o_t at timestep t is:

$$p(o_t|o_{1:t-1}, x_{prompt}) = \textsc{softmax}(\mathbf{h}_t^{dec}\mathbf{W} + \mathbf{b}) \tag{3.4}$$

where $\mathbf{W} \in R^{d \times |\mathcal{V}|}$, $\mathbf{b} \in R^{|\mathcal{V}|}$, d is the size (dimensions) of the decoder's hidden state and $|\mathcal{V}|$ is the vocabulary size of the pretrained model. We then apply the negative log likelihood loss between the desired output and decoder output.

$$\mathcal{L} = -\sum_{t=1}^{m} \log p(o_t|o_{1,t-1}, x_{prompt}) \tag{3.5}$$

3.5 Experimental Results

3.5.1 Training details

In all experiments we used Unified-QA-v2-1363200 base model and the Adafactor optimizer [48]. We tried learning rates 0.0001, 0.001, 0.0002. Usually, smaller learning rates performed better. The batch size is set to 12. The dropout rate is set to 0.1 . Early stopping is also used on F1 to avoid overfitting.

When using CoNLL03 dataset, we trained for 7 epochs. When using MIT-Restaurant dataset, i.e., for few-shot learning, we trained Unified-QA for 30 epochs.

During inference, for every sentence we need to ask a question for every possible named entity type. Nevertheless, a sentence may not have a particular named entity type and

the model should learn to generate the proper text for unanswerable questions. For this purpose, during training we add some unanswerable questions. If a question doesn't have an answer, we consider "<no answer>" as the correct (ground truth) answer. Usually, for most named entity types we use an equal amount of questions that have answers and that are unanswerable.

For few shot learning we run all the experiments four times and we report average scores over the four repetitions.

3.5.2 Evaluation

We evaluate all the different models based on their performance over the test datasets. In the following equations TP, FP, and FN represent the True Positives, False Positives, and False Negatives respectively. Precision calculates what percentage of named entities found by the model is correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.6}$$

Recall calculates what percentage of named entities existing in the dataset is found by the model.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.7}$$

The evaluation metric we report is F1-score. F1-score is the harmonic mean of the precision and recall.

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.8}$$

We removed BIO-tags and consider that a named entity is correct only when it is an exact match of the corresponding entity in the dataset sentence. We do entity-based (as opposed to token-based) evaluation, i.e., all the tokens of an entity name (and only those) need to be identified correctly, for the entity name to be counted as a true positive. When our model considers as a named entity words that are not consecutive in the input sentence, we regard it as an extra False Positive that is included when calculating the results.

During inference, when two or more text spans are given distinct labels, despite having text overlap, we only keep the text span with the maximum score to avoid prediction inconsistencies. This approach is also adopted by [7].

3.5.3 Results

We compare our approach with other methods existing in the literature. LSTM-CNNs-CRF [33] utilizes a CNN [20] for characters representations that are combined with word-level representations. Both are fed into an LSTM [14] and followed by a CRF layer. LUKE [56] is an MLM that augments the attention mechanism by adding an entity-aware query mechanism, which considers the type of the tokens (words or entities) when calculating self-attention. Label Refinement for Sequence Labeling [11] uses ideas from Bayesian Neural Networks and encodes word-to-label interactions in the attention mechanism of the Transformer Architecture [52]. BERT [9] uses a bidirectional Transformer encoder with a fine-tune head on top. MRC for Named Entity Recognition [24] and Example-NER [61] use BERT and reformulate NER as a QA problem. NER as Dependency Parsing [57] uses the Biaffine model on top of a multi-layer LSTM. Sequence Labeling BART [7] uses BART encoder with a fine-tune head on top. Label-Agnostic sequence labelling [55] uses a nearest-neighbor approach which copies labels from retrieved neighbors. TemplateBART and LightNER were explained in Section 3.2. MP-NSP [16] is a prototype based method that also performs self-training on external web data.

Supervised learning

First, we evaluate our method in a standard fully supervised setting. As shown in Table 3.1, our approach does not perform as well as the other methods. We observe that NERasQA has the lowest recall and f1-score. The comparison between our results and the results of MRC for Named Entity Recognition indicates that perhaps UnifiedQA is not the most suitable model for the task. Note that, when comparing different methods, they do not always use the same tokenizers.

Traditional Models	**Precision**	**Recall**	**F1**
LSTM-CNNs-CRF[33]	-	-	91.2
LUKE [56]	-	-	92.4
Label Refinement for Sequence Labeling[11]	-	-	92.0
MRC for Named Entity Recognition[24]	92.4	93.2	92.8
NER as Dependency Parsing [57]	92.8	92.1	92.5
Sequence Labeling BERT [9]	91.9	91.5	91.7
Sequence Labeling BART [7]	89.6	91.6	90.6
Few-shot Friendly Models	**Precision**	**Recall**	**F1**
Label-Agnostic Sequence Labelling [55]	-	-	89.9
TemplateBart [7]	90.5	93.3	91.9
LightNER [6]	92.3	93.4	92.9
NERasQA (Ours)	91.3	84.7	87.9

Tab. 3.1: Fully supervised setting performance on the CoNLL03 dataset.

Few-shot learning

First, we create a few-shot learning scenario on CoNLL-2003 by downsampling, which reduces the training examples for some specific named entity types. Specifically, we downsample the CoNLL-2003 training set and generate 4,001 training sentences which include 2,496 Person named entities, 3,763 Organization named entities, 100 Miscellaneous named entities and 100 Location entities. For more information about the training distribution,

Models	PER	ORG	LOC*	MISC*	Overall
Sequence Labeling BART [7]	75.7	73.5	58.7	57.3	66.8
BERT[9]	76.2	75.3	61.5	59.3	68.1
TemplateNER [7]	84.4	72.6	71.9	73.3	75.5
LightNER [6]	90.9	76.8	81.5	52.0	78.9
NERasQA(ours)	86.7	74.2	78.2	56.3	76.3

Tab. 3.2: In-domain few-shot F1 performance on the CoNLL03 dataset. * indicates a few-shot entity type.

Source	Methods	10	20	50	100	200	500
None	BERT[9]	21.8	39.4	52.7	53.5	57.4	61.3
	Sequence Labeling BART[7]	6.3	8.5	51.3	52.2	56.3	60.2
	TemplateNER[7]	46.0	57.1	58.7	60.1	62.8	65.0
	LightNER[6]	48.5	58.0	62.0	70.8	75.5	80.2
	NERasQA(ours)	47.3	57.8	65.8	70.2	73.7	76.2
CoNLL	Label-Agnostic[55]	4.1	3.6	4.0	4.6	5.5	8.1
	Example-NER[61]	25.2	26.1	26.8	26.2	25.7	25.1
	BERT[9]	27.2	40.9	56.3	57.4	58.6	75.3
	MP-NSP[16]	46.1	48.2	49.6	49.6	50.0	50.1
	Sequence Labeling-BART[7]	8.8	11.1	42.7	45.3	47.8	58.2
	TemplateNER[7]	53.1	60.3	64.1	67.3	72.2	75.7
	LightNER[6]	58.1	67.4	69.5	73.7	78.4	80.1
	NERasQA(ours)	47.7	60.3	66.9	69.7	74.2	77.0

Tab. 3.3: Few-shot F1 on the MIT Restaurant dataset when using 10, 20, 50, 100, 200, and 500 training instances per entity type. The "Source" column indicates whether the models where trained from scratch (None) or pre-trained on CoNLL.

we refer readers to Section 3.3. As shown in Table 3.2, NERasQA performs better than most other methods in this setting. In general, prompt-based methods outperform Sequence Labeling BART and BERT with a large margin. This demonstrates the success of prompting on few-shot scenario. Surprisingly, the two highest performing methods overall, i.e., LightNER and NERasQA, perform better on low-resource type Location compared to high-resource entity type Organization. This may be explained by the hypothesis that these methods better leverage the knowledge of LMs for this entity type.

Next, we used MIT-Restaurant dataset where we randomly sample a fixed number of training instances per entity type (10, 20, 50, 100, 200, 500 instances). First, we train on MIT Restaurant without having trained on CoNLL data beforehand. We observe that NERasQA performs better than all other methods in all k-shot settings except for the LightNER method. However, for k= 50 NERasQA outperforms LightNER. High performance of NERasQA can be explained by the fact that it was trained beforehand on several QA datasets.

In another few shot scenario, we utilized the already fine-tuned on CONLL model and then we further trained the model on MIT-Restaurant. We only used again a limited number of training examples from MIT-Restaurant dataset (10, 20, 50, 100, 200, 500 instances). We observe no gains compared to training from scratch on MIT-Restaurant. Our method is

closer to Example-NER [61] that reformulates NER as QA problem. Example-NER [61] performs much worse than our method, because it applies fine-tuning only on the CoNLL dataset not on the MIT-Restaurant dataset.

Note that we tried also other templates apart from "Which [E] is present in the text? \n [X]" but the performance was worse.

Briefly based on the above, NERasQA performed comparably to the best methods when overall we only trained on limited data. Nevertheless, when the model was trained on the full CoNLL dataset and then was evaluated on it or further trained on MIT-Restaurant, it did not perform well.

4 Auxiliary Task for NER

4.1 Introduction

Apart from work on few shot learning, we have also developed an auxiliary task for Named Entity Recognition (NER), which the model is trained to perform concurrently with NER, in an attempt to improve performance in the NER task. The auxiliary task requires the model to predict the types (categories) of masked named entities, based only on their context.

4.2 Related Work

Previous work found that the primary element that influences NER results is learning the named entities, while context affects predictions at a smaller degree [2]. Specifically, during testing, they masked the entire dataset one word each time and made NER predictions depending only on the context resulting in a large drop in performance. Each word was masked exclusively during the testing process, while masking was not applied while fine-tuning. Another finding of the study is BERT's ability to recognize successfully the entity type from the context, despite the fact that humans are often unsuccessful in this respect.

Other researchers created adversarial attacks and observed significant drop in NER performance when replacing named entities with others of the same named entity category [25]. BERT has been found not to be robust to named entity replacements on Natural Language Inference, Coreference Resolution, and Grammar Error Correction tasks [3]. Other work substituted named entities with other similar entities of a different nationality and found that the performance of NER models highly depends on the entities of each country [1]. Similarly, other work empirically showed that models have better performance when person named entities are white names [36].

4.3 Dataset

The dataset we used is OntoRock [25]. The dataset is created by applying two different types of attacks to the development set and the test set of OntoNotes [39]. Firstly, the original entities were substituted from Wikidata, which are out of the training distribution, in order to create entity-level attacks. Secondly, by using BERT [9], the context words were replaced by others of similar semantics but out of the training distribution, in order to create context-level attacks. It is important to note that while the context changed, the syntax remains correct. Both adversarial attacks were also applied concurrently. OntoRock contains 11 named entity types. More information about OntoRock Dataset can be found in Table 4.1.

	Train	**Dev**	**Test**
# Sentences	59,924	8,528	8,262
# Tokens	1.1M	148k	153k
# Entities	55,008	7,482	7,433
# Attacked Entities	N/A	6,962	6,939
% Attacked Entities	N/A	93.05	93.35
# Attacked Context Words	N/A	16,155	15,664
% Attacked Sentences	N/A	98.03	97.53

Tab. 4.1: Statistics of the OntoRock dataset [25].

4.4 Model

We create an auxiliary task that tries to predict the types of the named entities using only the context. We mask the named entities present in the sentence and feed the result into the T5 model [40]. The desired output of T5 is a sequence containing the types of named entities present in the initial NER sentence. The auxiliary task is trained concurrently with the NER task.

Figure 4.1 illustrates the following example. Consider the sentence 'John broke his leg'. We mask the named entities present in the sentence and get '`<extra_id_0>` broke his leg' which is fed into T5 model. The desired output is a sequence containing the types of named entities in the input sentence, i.e., in the specific example '`<extra_id_0>` Person `<extra_id_1>`'. We have a distinct sentinel token (`<extra_id_>`) for every named entity in the input sentence. Intuitively, with the auxiliary task we hope that the model learns to predict named entities using the context exclusively.

For the main task we feed into the T5 model the initial NER sentence and expect T5 to generate for every named entity their named entity type followed by the respective named entity itself. Figure 4.2 shows the following example. Consider the previous sentence 'John

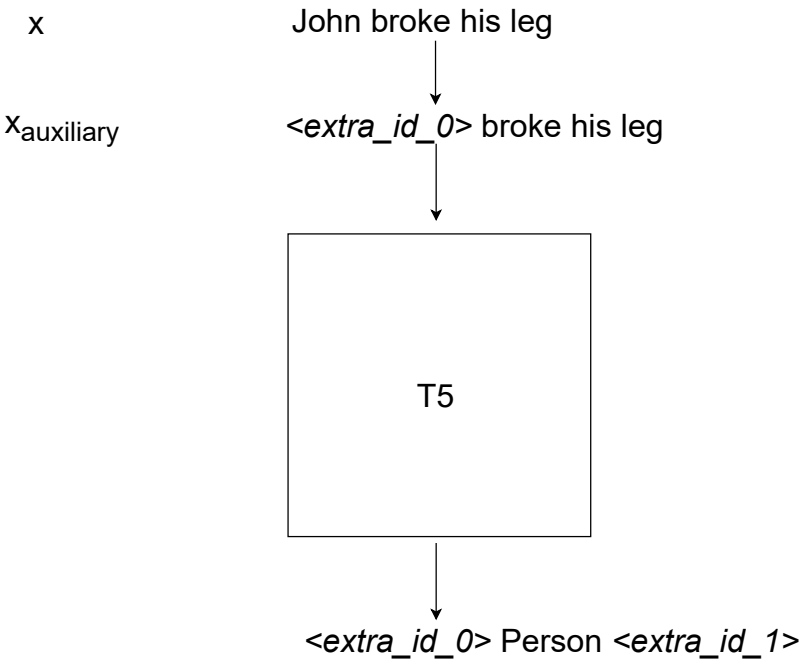

Fig. 4.1: Auxiliary task using the T5 model. The model is expected to generate the named entity category of every named entity present in the sentence.

broke his leg' which is fed into T5 model. The desired output is a sequence containing the types of named entities in the input sentence followed by the named entities themselves, i.e., in the specific example 'Person John'.

The training procedure uses the same equations (Eq. 3.2 - 3.5) as mentioned in Section 3.2 above. We use use the base variant of T5, which has 220 million parameters.

4.5 Experimental Results

4.5.1 Training details

As stated previously we trained concurently the main and the auxiliary task. We concatenate the two datasets and randomly sample examples from the combined dataset. Regarding the auxiliary task, we removed the following sentences, as there was nothing the model should learn from them:

- all the sentences without any named entities,
- all sentences whose tokens are all part of a named entity,

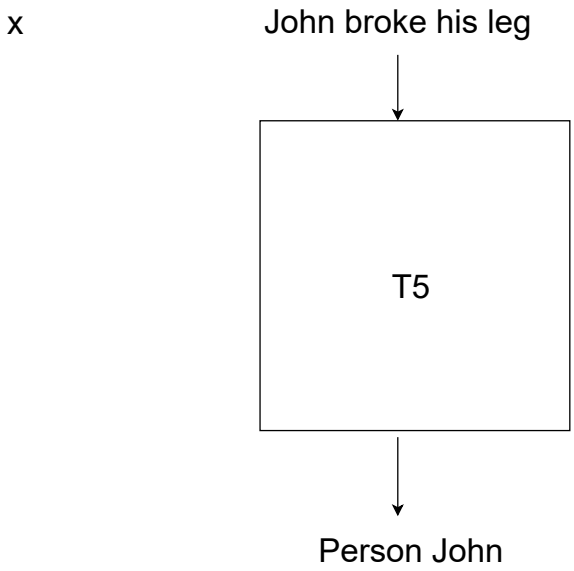

Fig. 4.2: Main NER task using the T5 model. The model is expected to generate a sequence containing the types of named entities in the input sentence followed by the named entities themselves.

- all sentences whose tokens are all part of a named entity or stopwords or punctuation marks.

The dataset size of the auxiliary task ended up being less than half the main dataset size. We used the extra prefix tokens <prefix1>,<prefix2> to differentiate the 2 tasks. We tried learning rates 0.0001, 0.001 for Adafactor optimizer. We also tried AdamW optimizer with learning rate 0.0001. We set batch size to 12. Some experiments were done with gradient accumulation 2. We trained for 6 epochs. Early stopping is also used on F1 in order to avoid overfitting. We set dropout to 0.1.

4.5.2 Evaluation

Suppose you have the aforementioned sentence 'John broke his leg'. As stated previously, for the auxiliary task we mask the named entities, i.e., 'John'. We consider as a correct answer generating the types of named entities in the input sentence, i.e., 'Person'.

For both tasks we calculated the F1-score using exact match. For more information about this metric, we refer readers to Section 3.5.2.

Models	EntityDev F1	ContextDev F1
RoBERTa-CRF	61.6	84.8
RB-CRF+Entity Switching*	63.1	83.7
RB-CRF+Random Masking*	63.8	84.3
RB-CRF+Mixing Up*	59.0	85.0
SimpleT5(ours)	55.6	76.2
MultiTaskT5(ours)	56.8	76.9

Tab. 4.2: Validation performance on OntoRock datasets. EntityDev implies attacks were made on the named entities. ContextDev implies attacks were made on the context . * implies data augmentantion techniques were used. MultiTaskT5 indicates results on the standard NER dataset when training concurently the main and auxiliary task.

Models	EntityDev F1	ContextDev F1
MultiTaskT5 (ours)	56.8	76.9
MultiAuxTaskT5 (ours)	70.3	70.6

Tab. 4.3: MultiTaskT5 indicates results on the standard NER dataset when training concurently main and auxiliary task. MultiAuxTaskT5 indicates results on the auxiliary task we created when training concurently main and auxiliary task.

4.5.3 Results

We compare our methods with the existing in the literature. RoBERTa with a CRF layer on top is one baseline. To improve the robustness of NER models, data augmentantion techniques [25] have been proposed. Entity Switching replaces each named entity with a different named entity of the same type. Random Masking randomly changes every letter of every named entity. Mixing Up concatenates the first half of a sentence with the second half of another sentence, where both sentences contain a common named entity type. We provide other methods' results as illustrated in RockNER paper [25].

Regarding our methods, we train T5 on OntoRock using only the main NER task (SimpleT5). We observe worse performance compared to Roberta-CRF and other methods which use data augmentantion ideas in both datasets. All methods showed much better performance on Context-Dev, where context-level attacks were made, comparing to Entity-Dev where entity-level attacks were made. This confirms the view that the primary factor that influences NER results is learning the named entities.

Next, we use concurrently both the auxiliary and the main NER task (MultiTaskT5). As we can see in Table 4.2 only marginal improvements were observed on main NER task of RockNER dataset when trained concurrently with the auxiliary NER-dataset. Surprisingly, the auxiliary task (MultiAuxTaskT5) performed pretty well even for the ContextDev (see Table 4.3). The above demonstrate that T5 may provide the correct answer in the auxiliary task for the wrong reasons and not due to making better use of the context.

Models	EntityDev F1
MultiTaskT5 (ours)	56.8
MultiTaskT5-Negative (ours)	56.7
MultiAuxTaskT5 (ours)	70.3
MultiAuxTaskT5-Negative (ours)	71.3

Tab. 4.4: MultiTaskT5 and MultiAuxTaskT5 are the models we reported in Table 4.3. MultiTaskT5-Negative indicates results on the standard NER dataset when training concurently main and auxiliary task using additionally some negative examples. MultiAuxTaskT5-Negative indicates results on the auxiliary task we created when training concurently main and auxiliary task using additionally some negative examples.

T5 probably exploits some spurious correlations to perform relatively well on the auxiliary task, because this doesn't transfer to improvement in the NER task. In order to reduce spurious correlations between the named entity types and the masked tokens of T5 (the <extra_id_> tokens), we added some negative examples during training. We used Sentence-T5 [37] to find training sentences from those which did not have any named entities, that are similar (in terms of sentence embedding similarity) to the training examples of auxiliary task. For example, suppose Sentence-T5 selected the sentence 'What kind of memory ?'. We added a sentinel token '<extra_id_0>' at a random position inside the selected sentence and T5 should generate '<extra_id_0> none <extra_id_1>'. Nevertheless, as we can see in Table 4.4, no substantial improvements were observed. Performance in the auxiliary task (MultiAuxTaskT5-Negative) continued to be reasonably high.

5 Conclusions

5.1 Key takeaways

In this thesis we examined different scenarios of the Named Entity Recognition (NER) task. Initially, following the work of [24, 61], we reformulated NER as a Question Answering (QA) problem. The model we used was Unified-QA [18] which is trained on a variety of QA tasks.

First, we only kept few-shot examples for some named entity types, whereas the other types contained a lot of examples in the CoNLL03 dataset [51]. Our approach outperforms most other methods like BERT or prompt-based template-NER [7] except for LightNER [6]. In another few-shot setting, where every named entity type has only few examples, our method similarly performed better than most other methods on the MIT-Restaurant [27] dataset. These results can be explained by the fact that the model we used, Unified-QA [18], was trained on multiple QA tasks.

Nevertheless, when trained on the full CoNLL03 dataset [51] and evaluated on its test set, our method performed worse than the results in the literature. When the already fine-tuned model on CoNLL03 was further trained on MIT Restaurant and evaluated on MIT-Restaurant test set, similarly our method had no gains in performance from using the fine-tuned on CoNLL03 model. These results illustrate that Unified-QA does not work well when training on a lot of data.

Concerning NER robustness, previous work found that the primary factor that influences NER results is learning the named entities themselves [2]. This characteristic of NER models leads also to bias problems [1, 36]. We tried to alleviate this problem by using an auxiliary task that tries to predict the named entity type using only the context. This task was trained concurrently with the standard NER task using the T5 model [40]. However, no substantial improvements were observed in the main NER task of the OntoRock dataset, whereas in the auxiliary task T5 performed moderately well. We suspect that this result is attributed to spurious correlations between the context and the named entity types or spurious correlations between the named entity types and the masked tokens of T5 (the <extra_id_> tokens).

5.2 Further work

In future work, we plan to better understand the spurious correlations present in the auxiliary task. We will also try alternative training strategies for the auxiliary and the main task, e.g. using an auxiliary loss weight in order to better control the contribution of the auxiliary task. We also plan to implement this approach in models trained with Masked Language Modeling (MLM), which are more suitable for the task of Named Entity Recognition. MLM models do not have the problem of text-to-text models (such as T5) which can generate non-successive words as named entities. MLM models also performed much better on the OntoRock dataset compared to T5. We also plan to evaluate on the validation set which contains both context and entity attacks and then evaluate on Test-sets.

Bibliography

[1]Oshin Agarwal, Yinfei Yang, Byron C Wallace, and Ani Nenkova. ""Entity-switched datasets: An approach to auditing the in-domain robustness of named entity recognition models". 2020.

[2]Oshin Agarwal, Yinfei Yang, Byron C. Wallace, and Ani Nenkova. "Interpretability Analysis for Named Entity Recognition to Understand System Predictions and How They Can Improve". In: *Computational Linguistics* 47.1 (2021), pp. 117–140.

[3]Sriram Balasubramanian, Naman Jain, Gaurav Jindal, Abhijeet Awasthi, and Sunita Sarawagi. "What's in a Name? Are BERT Named Entity Representations just as Good for any other Name?" In: *Proceedings of the 5th Workshop on Representation Learning for NLP*. 2020, pp. 205–214.

[4]Eyal Ben-David, Nadav Oved, and Roi Reichart. "Pada: A prompt-based autoregressive approach for adaptation to unseen domains". 2021.

[5]Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1877–1901.

[6]Xiang Chen, Ningyu Zhang, Lei Li, Xin Xie, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. "Lightner: A lightweight generative framework with prompt-guided attention for low-resource ner". 2021.

[7]Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. "Template-Based Named Entity Recognition Using BART". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online, 2021, pp. 1835–1845.

[8]Roxana Danger, Ferran Pla, Antonio Molina, and Paolo Rosso. "Towards a Protein–Protein Interaction information extraction system: Recognizing named entities". In: *Knowledge-Based Systems* 57 (2014), pp. 104–118.

[9]Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, 2019, pp. 4171–4186.

[10]Tianyu Gao, Adam Fisch, and Danqi Chen. “Making Pre-trained Language Models Better Few-shot Learners”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online, 2021, pp. 3816–3830.

[11]Tao Gui, Jiacheng Ye, Qi Zhang, Zhengyan Li, Zichu Fei, Yeyun Gong, and Xuanjing Huang. “Uncertainty-Aware Label Refinement for Sequence Labeling”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online, 2020, pp. 2316–2326.

[12]Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. “WARP: Word-level Adversarial ReProgramming”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online, 2021, pp. 4921–4933.

[13]Adi Haviv, Jonathan Berant, and Amir Globerson. “BERTese: Learning to Speak to BERT”. In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online, 2021, pp. 3618–3623.

[14]Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[15]Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. “Parameter-Efficient Transfer Learning for NLP”. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 2790–2799.

[16]Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. “Few-Shot Named Entity Recognition: An Empirical Baseline Study”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic, 2021, pp. 10408–10423.

[17]Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. “How Can We Know What Language Models Know?” In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 423–438.

[18]Daniel Khashabi, Yeganeh Kordi, and Hannaneh Hajishirzi. "UnifiedQA-v2: Stronger Generalization via Broader Cross-Format Training". 2022.

[19]Daniel Khashabi, Shane Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sameer Singh, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, and Yejin Choi. "PROMPT WAYWARDNESS: The Curious Case of Discretized Interpretation of Continuous Prompts". 2021.

[20]Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[21]Brian Lester, Rami Al-Rfou, and Noah Constant. "The Power of Scale for Parameter-Efficient Prompt Tuning". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic, 2021, pp. 3045–3059.

[22]Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online, 2020, pp. 7871–7880.

[23]Xiang Lisa Li and Percy Liang. "Prefix-Tuning: Optimizing Continuous Prompts for Generation". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online, 2021, pp. 4582–4597.

[24]Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. "A Unified MRC Framework for Named Entity Recognition". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online, 2020, pp. 5849–5859.

[25]Bill Yuchen Lin, Wenyang Gao, Jun Yan, Ryan Moreno, and Xiang Ren. "RockNER: A Simple Method to Create Adversarial Examples for Evaluating the Robustness of Named Entity Recognition Models". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic, 2021, pp. 3728–3737.

[26]Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. "What Makes Good In-Context Examples for GPT-3?" 2021.

[27]Jingjing Liu, Panupong Pasupat, Scott Cyphers, and Jim Glass. "Asgard: A portable architecture for multilingual dialogue systems". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 8386–8390.

[28]Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing". 2021.

[29]Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "Roberta: A robustly optimized bert pretraining approach". 2019.

[30]Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. "Cutting down on prompts and parameters: Simple few-shot learning with language models". 2021.

[31]Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity". 2021.

[32]Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. "The Expressive Power of Neural Networks: A View from the Width". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6232–6240.

[33]Xuezhe Ma and Eduard Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, 2016, pp. 1064–1074.

[34]Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. "The natural language decathlon: Multitask learning as question answering". 2018.

[35]Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. "Metaicl: Learning to learn in context". 2021.

[36]Shubhanshu Mishra, Sijun He, and Luca Belli. ""Assessing Demographic Bias in Named Entity Recognition". 2020.

[37]Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. "Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models". 2021.

[38]Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. "Language Models as Knowledge Bases?" In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China, 2019, pp. 2463–2473.

[39]Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. "Towards Robust Linguistic Analysis Using OntoNotes". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. 2013, pp. 143–152.

[40]Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67.

[41]Pranav Rajpurkar, Robin Jia, and Percy Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia, 2018, pp. 784–789.

[42]Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, 2016, pp. 2383–2392.

[43]Roger Ratcliff. "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." In: *Psychological review* 97.2 (1990), p. 285.

[44]Lisa F. Rau. "Extracting Company Names from Text". In: *Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications* (1991), pp. 29–32.

[45]Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. "Multitask Prompted Training Enables Zero-Shot Task Generalization". In: *The Tenth International Conference on Learning Representations*. 2022.

[46]Timo Schick and Hinrich Schütze. "Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online, 2021, pp. 255–269.

[47]Timo Schick and Hinrich Schütze. "It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online, 2021, pp. 2339–2352.

[48]Noam Shazeer and Mitchell Stern. "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018, pp. 4596–4604.

[49]Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online, 2020, pp. 4222–4235.

[50]Matus Telgarsky. "Benefits of Depth in Neural Networks". In: *Journal of Machine Learning Research* 49 (2016), pp. 1517–1539.

[51]Erik F. Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 142–147.

[52]Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA, 2017, pp. 6000–6010.

[53]Albert Webson and Ellie Pavlick. "Do Prompt-Based Models Really Understand the Meaning of their Prompts?" 2021.

[54]Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. ""Finetuned language models are zero-shot learners". 2021.

[55]Sam Wiseman and Karl Stratos. "Label-Agnostic Sequence Labeling by Copying Nearest Neighbors". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy, 2019, pp. 5363–5369.

[56]Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. "LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online, 2020, pp. 6442–6454.

[57]Juntao Yu, Bernd Bohnet, and Massimo Poesio. "Named Entity Recognition as Dependency Parsing". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online, 2020, pp. 6470–6476.

[58]Weizhe Yuan, Graham Neubig, and Pengfei Liu. "Bartscore: Evaluating Generated Text as Text Generation". In: *Advances in Neural Information Processing Systems* 34 (2021).

[59]Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. "Calibrate Before Use: Improving Few-shot Performance of Language Models". In: *Proceedings of the 38th International Conference on Machine Learning*. 2021, pp. 12697–12706.

[60]Zexuan Zhong, Dan Friedman, and Danqi Chen. "Factual Probing Is [MASK]: Learning vs. Learning to Recall". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online, 2021, pp. 5017–5033.

[61]Morteza Ziyadi, Yuting Sun, Abhishek Goswami, Jade Huang, and Weizhu Chen. "Example-based named entity recognition". 2020.

List of Acronyms

NER Named Entity Recognition

NLP Natural Language Processing

QA Question Answering

LM Language Model

MLM Masked Language Model

List of Figures

List of Tables