Universidad del País Vasco Euskal Herriko Unibertsitatea

Natural Language Inference Models for Few-Shot Text Classification A Real-World Perspective

Author: David Romero

Advisors: Eneko Agirre and Oier Lopez de Lacalle

hap/lap

Hizkuntzaren Azterketa eta Prozesamendua
Language Analysis and Processing

Final Thesis

February 2023

Departments: Computer Systems and Languages, Computational Architectures and Technologies, Computational Science and Artificial Intelligence, Basque Language and Communication, Communications Engineer.

Abstract

Pivoting tasks as entailment problems have shown to be very effective in different applications like question answering and relation extraction. On the other hand, language models trained for entailment tasks have demonstrated to be very effective in settings with small training sets and have better generalization abilities. In view of this, in this work we recast text classification as an entailment problem, specifically, I build PET-NLI, an approach that uses the same architecture and training procedure as Pet (Pattern-Encoding Training), but in this case using natural language inference (NLI) models. Overall, PET-NLI shows to have the same benefits of PET for true few-shot scenarios despite using a different type of language model. This approach is tested on true-few shot scenarios of the RAFT benchmark. As a result, PET-NLI outperforms bigger models like GPT-3 and achieves competitive performances when it is compared with the original PET architecture and the Human Baseline.

Contents

List of Figures

List of Tables

1 Introduction

Pre-trained Language Models (PLMs) have been a crucial component of many advances in Natural Language Processing (NLP), one of the reasons for their success is due to the effectiveness that it provides in terms of the use of data. It has become a common approach to utilize large PLMs for initialization, followed by fine-tuning with gradient-based methods for downstream tasks. Given that the PLM has already been trained on a large dataset, it has learned general language representations, which allows the model to quickly learn task-specific patterns and features from smaller datasets in few-shot settings. Because of this, fine-tuning a PLM with a large set of examples is not always needed to get a good performance, making it an attractive approach for real-world cases.

PLMs such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2020) AlBERT (Lan et al., 2020), or bigger models like GPT-3 (Brown et al., 2020) have shown astounding few-shot capabilities by reformulating downstream tasks as similar cloze questions and using tasks descriptions in natural language. This is defined as Prompt-based learning, this method of training machine learning (ML) models uses a template to change the original input of a task into a sentence with an unfilled masked token, so that later a PLM is used to generate the most probable output to fill the slot, meeting the requirements of the description in the prompt, and finally, the prediction is mapped to a certain label for the task at hand. This is better shown in Figure 1.

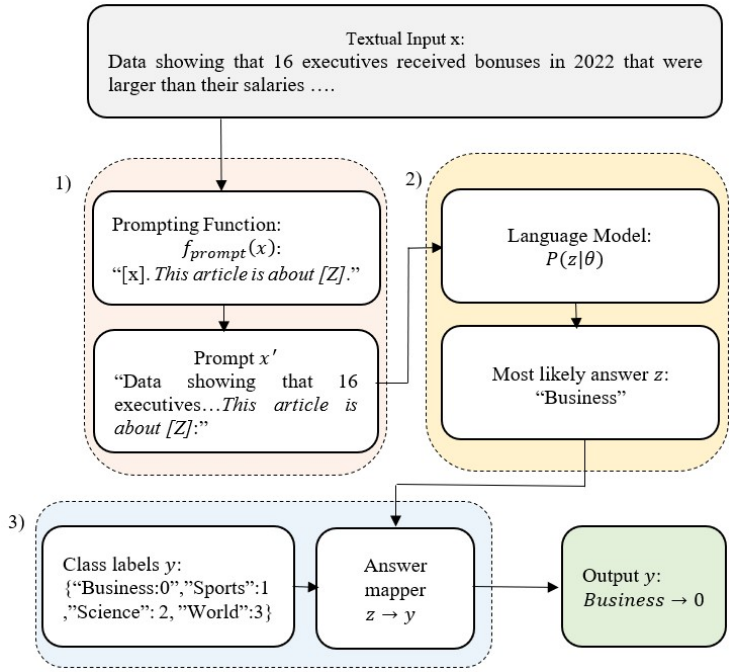

Figure 1: Prompt-based Learning. 1) A prompting function is used to insert x into a prompt template with an unfilled masked token. 2) A finetuned PLM is used to fill the slot with the most probable answer. 3) The selected answer is mapped to a certain label.

By providing a clear and explicit description of a task in the form of a natural language prompt the model can better understand the characteristics and requirements of the task even with a limited amount of data (Wang et al., 2022). Works like (Shin et al., 2020; Gao et al., 2021) have demonstrated the potential of using natural language prompts with PLMs, showing that this setting helps the model to utilize its knowledge more effectively, strengthening the connection between input and output. While Prompt-based methods work really well for different tasks, this approach may struggle to accurately make good predictions in cases where the label to be predicted requires a more detailed or elaborate description. On the other hand, its success is also dependent on the prompt design, manually determining the right prompts can be a time-consuming and error-prone process that requires expertise, this has led researchers to explore various approaches (Schick and Schütze, 2021a; Zhou et al., 2021, 2022; Arora et al., 2023) to design or use prompts in ways that can make PLMs get good performances. Finally, Perez et al. (2021) has shown that previous works have overestimated the ability of prompt-based approaches in true few-shot settings, where there are no development sets available, casting doubts in their performance due to the difficulties of these models in settings where prompts and hyperparameters cannot be tuned.

In addition to prompt-based methods, pivot tasks have frequently been used for few and zero-shot learning. More related to our study, tasks like text classification (Yin et al., 2019; Wang et al., 2021) have been successfully pivoted as NLI problems. These works have shown that text classification reformulated as a textual entailment task can obtain really good performances when it is compared to standard few-shot learning and fine-tuning methods. The main idea of the approach is to convert the class label into a textual description, and use this description along with the training example as the hypothesis and premise respectively, for then use an NLI model to infer whether the premise entails or not the hypothesis. If an "entailment" relation is predicted, the corresponding label represented by the hypothesis is classified as a certain class for the task at hand. This approach can be seen in Figure 2.

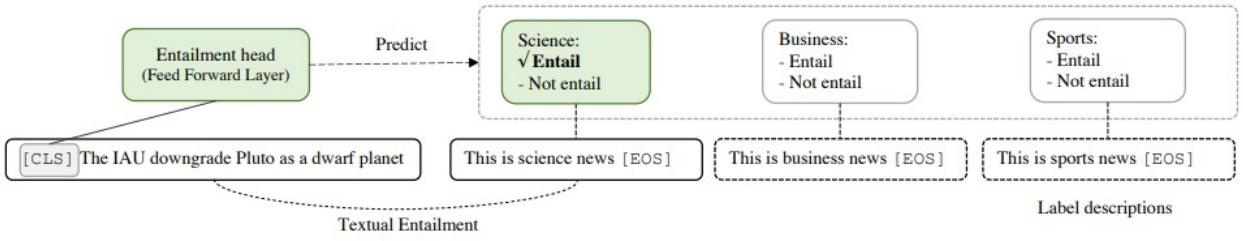

Figure 2: Entailment-based text classification. (Wang et al., 2021)

In addition, Wang et al. (2021) has shown that by converting tasks as entailment problems, PLMs are effective few-shot learners, and this could be due to when a model perform natural language inference is doing real language understanding, helping PLMs to generalize its knowledge to other tasks that can be reformulated as entailment problems, making it an effective approach for few-shot settings.

In this thesis, I take inspiration from these works, I aim to recast text classification as an entailment problem and apply this approach to the Real-world Annotated Few-shot Tasks (RAFT) (Alex et al., 2021) . Particularly, I build the same architecture as Pattern-Exploiting Training (PET) (Schick and Schütze, 2021a), which is a training method that combines textual instructions with example-based finetuning, whose key feature is that it uses unlabeled data for few-shot scenarios. However, unlike PET which uses masked language models trained with prompt-based methods for text classification, I use pre-trained NLI models which are finetuned using label verbalizations as hypotheses and training examples as premises, I call this approach PET-NLI.

Due to RAFT not having a development set available in which a certain approach can be tested or the hyperparameters of a model can be finetuned, I take the following strategy: First, I simulate few-shot scenarios using three other English datasets that resemble real-world settings, "Yelp Reviews", "AG's news" and "Yahoo Questions and Answers", I use these datasets to analyze the performance of the proposed approach and obtain the optimal hyperparameters, to then, test the proposed architecture with the best findings on the real-world tasks given by the RAFT benchmark. Overall, PET-NLI achieves a competitive average performance when it is compared against the original PET architecture, by either outperforming or achieving the same performance in 6 out of 11 tasks of RAFT. On the other hand, PET-NLI clearly outperforms bigger models like GPT-3 in the total average performance and either surpasses or gets a similar performance of non-expert humans on 5 out of 11 tasks from RAFT.

The objective of this research is to the analyze the performance of NLI models on real-world few-shot text classification tasks, for this I build PET-NLI, an approach that uses NLI models to perform text classification in true few-shot scenarios. This work is focused on the comparison of this approach with the original PET architecture which uses masked language models. This thesis is organized as follows: Chapter 2 explains the state of the art, which includes the background information for this work. Chapter 3 present the architecture of PET-NLI, Chapter 4 describes the RAFT benchmark, Chapter 5 explains the experimental setup, and finally Chapter 6 shows a detail overview of our results, to then describe our conclusions and future work in Chapter 7.

2 State of the Art

In this work, I will build PET-NLI, an approach that uses NLI models to perform text classification in few-shot scenarios. I test this approach in the RAFT benchmark and compare its performance against the original PET architecture. To provide context for this comparison, in this section, I will first introduce of the topic of Natural Language Inference and Few Shot text classification, to then describe the PET architecture in greater detail.

2.1 Natural Language Inference

Natural Language Inference (NLI) is a task of natural language processing initially presented at (Dagan et al., 2006), which involves determining whether a given hypothesis logically follows from a premise. NLI models are machine learning (ML) models that are trained to perform this task. These models are based on architectures like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2020) or AlBERT (Lan et al., 2020) and are trained to take as input a pair of sentences, the premise and the hypothesis, and give as output a label indicating whether the premise Entails, contradicts, or is neutral with respect to the hypothesis (Bowman et al., 2015; Welleck et al., 2019). In more detail, these labels represent the following relationships: "Entailment" where the premise entails or logically implies the hypothesis; "Neutral" where the premise and the hypothesis are unrelated or unclear between each other; and "Contradiction" where the premise contradicts or disproves the given hypothesis. An example can be seen in the following figure:

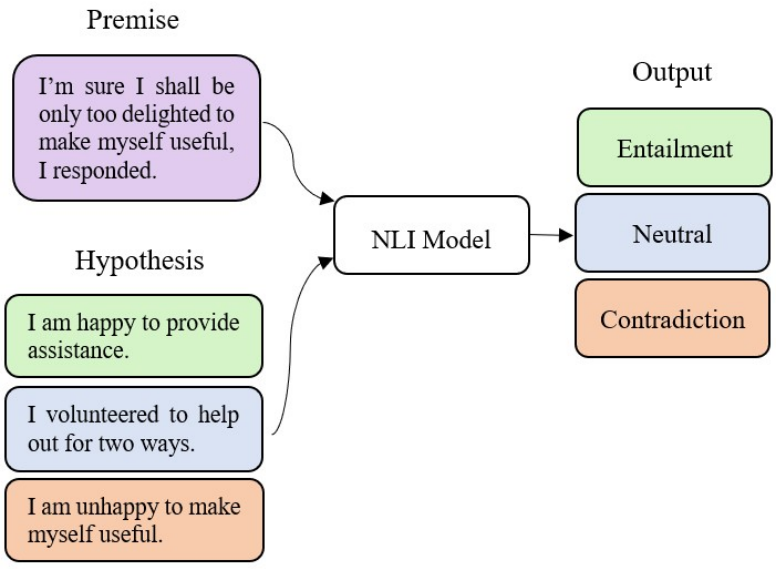

Figure 3: An example of the input and output of an NLI model

In order to accurately determine the relationship between two sentences, an NLI model must be able to understand the underlying meaning and relationships within the premise and hypothesis, as well as be able to reason about the relationships between different

concepts and ideas . ML models that perform well on this task are generally thought to have a strong understanding of natural language, and are able to generalize their knowledge to other tasks. One key aspect that has helped researchers to improve these models is the availability of large, high-quality NLI datasets like SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), ANLI (Nie et al., 2020) or FEVER (Thorne et al., 2018). In the next paragraphs, we will take a closer look at these datasets and discuss some of their key characteristics and features.

2.1.1 Stanford Natural Language Inference Corpus (SNLI)

SNLI (The Stanford Natural Language Inference Corpus) is a widely-used resource for Natural Language Processing research. It contains 570,152 English sentence pairs (premises and hypotheses) for the training set, and 10,000 examples for the development and test sets. For each sentence pair, the premise is obtained from image captions, and the hypotheses are provided by humans for one of the following labels: entailment, contradiction and neutral, representing each the logical relationship between the premise and the hypothesis. The SNLI dataset is notable for its large size and the fact that it was created by humans in naturalistic scenarios. In addition, the quality of its annotations was validated through a separate phase showing a consensus among different annotators. Therefore, the SNLI database is a valuable resource for NLP research and can be used to evaluate the performance of machine learning models on various tasks related to natural language understanding. (Bowman et al., 2015)

2.1.2 Multi-Genge Natural Language Inference (MNLI)

The MultiNLI (Multi-Genre Natural Language Inference) is a large dataset that was created to improve the coverage and difficulty of SNLI, and for its use in the development and evaluation of ML models in sentence understanding. It consists of 392,702 English sentence pairs (premises and hypotheses) for the training set, and 20,000 examples for the development and test sets. Contrary to SNLI, which have sentences derived from only a single text genre (image captions), being not sufficiently demanding to be a good benchmark for Natural Language Understanding, MultiNLI includes different genres from written and spoken english, where its premises are obtained from ten different sources, including face-to-face conversations, government press releases, letters, 9/11 reports, Oxford University press, magazine articles, telephone conversations, travel guides, linguistic posts, and fiction texts. This variety makes it a useful resource for evaluating the generalization ability of NLP models across different text types. On the other hand, the hypotheses were created by humans for one of the following labels: entailment, contradiction and neutral. Overall, the MultiNLI database can be used to evaluate the performance of machine learning models on tasks related to natural language understanding and text classification, particularly in the context of diverse text contents. (Williams et al., 2018)

2.1.3 Adversarial Natural Language Inference (ANLI)

The ANLI (The Adversarial Natural Language Inference) dataset is a large-scale NLI benchmark designed to challenge the performance of machine learning models on the task of Natural Language Inference. Overall, it consists of 162,865 sentence pairs (premises and hypotheses) for the training set, and 3,200 examples for the development and test sets. One key feature of the ANLI dataset is that it was created by an iterative, adversarial human-and-model-in-the-loop dataset collection, where first, a human annotator devises examples that are not predicted correctly by a selected NLP model, then those examples are added to a new trained set that is used to train a stronger model, this process is repeated three rounds, with the model getting stronger using a harder test set in each round. Similar to MNLI, for each sentence pair in the dataset, the premise is created from longer, multi-sentence source material and the hypotheses were created by humans and later verified by other annotators. Overall, the ANLI dataset is particularly useful for studying the robustness of NLP models as it was created to be a challenge for measuring the progress in Natural Language Understanding. (Nie et al., 2020)

2.1.4 Fact Extraction and VERification (FEVER)

The FEVER (Fact Extraction and VERification) is a large resource for Natural Language Processing research focused on the task of verifying claims against textual sources. It consists of 145,449 examples in the training set, and an additional 9,999 examples in the development and test sets. Each example in the dataset consists of a claim, evidence, and a label indicating whether the claim is "SUPPORTED," "REFUTED," or "NOTENOUGH-INFO." The claims in the FEVER dataset were generated from information in Wikipedia and then modified in various ways, to then assign a label based on the relationship between the claim and the evidence. The FEVER dataset is useful for evaluating the performance of machine learning models on tasks related to fact-checking and veracity prediction, and can be used to study the ability of these models to handle diverse text genres. (Thorne et al., 2018).

Overall, because NLI models have a deep understanding of the meaning and relationships between words and phrases in a natural language context, it plays a crucial role in a variety of applications. Thus, as these models continue to improve, they are increasingly being used as a benchmark for Natural Language Understanding (NLU), being important in applications such as text summarization and machine translation (Sadat and Caragea, 2022), and improving the performance of downstream tasks such as fact verification (Martín et al., 2021), fake news detection (Sadeghi et al., 2022) or relation extraction (Sainz et al., 2021).

2.2 Few-Shot Text Classification

Few-shot text classification is an NLP task in which a model is trained to classify text into different classes, using only a few training examples per class to learn from. (Wei

et al., 2021) Since GPT-3 (Brown et al., 2020), researchers have been exploring the potential of using pretrained language models effectively with a limited number of training examples. Several studies have focused on this approach, known as few-shot learning, as a way to leverage the power of these models even when a large amount of labeled data is not available, with the goal for the model to generalize well from few examples and accurately classify new, unseen text into the appropriate class (Wang et al., 2022).

There are many approaches that use few-shot learning with pretrained language models, such as in context learning (ICL) (Devlin et al., 2019; Radford et al., 2019), parameter-efficient fine-tuning (PEFT) (Liu et al., 2022), and pattern exploiting training (PET) (Schick and Schütze, 2021a). In this work we use PET as a baseline to compare the performance of NLI models, therefore, this method is described in greater detail below.

2.2.1 Pattern Exploiting Training (PET)

Pattern Exploiting Training (PET) is a semi-supervised method that combines textual instructions with example-based finetuning for few-shot text classification. PET is designed to reformulate input examples as cloze-style patterns, and recast labels to words using a verbalizer. This is done to help PLMs to better understand a task and maximize the use of their knowledge for downstream applications. This method fine-tune many PLMs for different pattern-verbalizer pairs, and then use those models to annotate unlabeled examples to create more training data for few-shot settings, where standard classifiers can be trained. PET has shown to be effective in scenarios with small training sets giving improvements when it is compared against standard supervised and semi-supervised approaches (Schick and Schütze, 2021a). Its architecture can be seen in Figure 4.

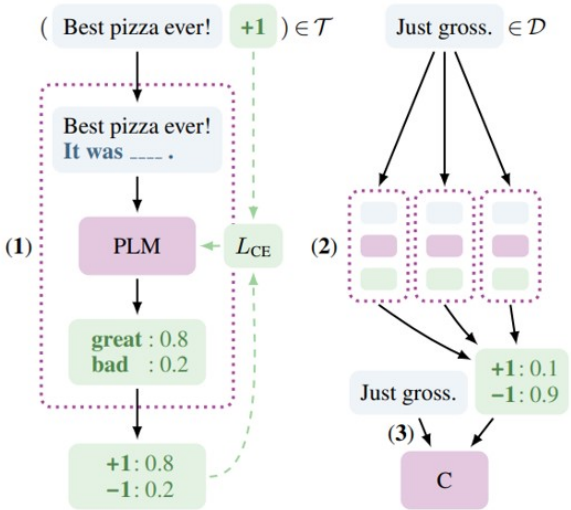

Figure 4: Pet Architecture. (1) A PLM is finetuned with a pattern that describes the task, (2) The ensemble of models trained on different patterns are used to annotate the unlabaled dataset D , (3) A classifier C is trained on the resulting soft-labeled dataset.

The architecture of PET is described in a greater detail in the following subsections:

Input: Let M be a masked language model with vocabulary V that uses as input a sequence of sentences $x = (s_1, ..., s_k)$ with $s_i \in V^*$. PET defines a certain pattern as a function P that takes x and gives as output a sentence $P(x)$ that has exactly one mask token. On the other hand, the corresponding label L of each sentence is mapped to a certain word, this is done by a verbalizer defined as an injective function $v : L \to V$. That is, PET takes an input x, and applies P to get an input representation $P(x) \in V^*$, which is used by the language model M to determine the most likely substitute w for the mask, being the substitute the predicted label. A pattern verbalizer pair (PVP) is defined as $p = (P, v)$. (Schick and Schütze, 2021a)

Training: For each sequence $z \in V^*$ containing exactly one mask token and $w \in V$, the function $M(w|z)$ is defined as the unnormalized score given by the language model to w at the masked position. Therefore, given an input x, the score given for the label $l \in L$, is defined as:

$$s_p(l|x) = M(v(l)|P(x))$$

And its probability distribution using softmax:

$$q_p(l|x) = \frac{e^{s_p(l|x)}}{\sum_{l' \in L} e^{s_p(l'|x)}}$$

The MLM is finetuned using cross-entropy and the language modeling loss between the predictions $q_p(l|x)$ and the true distribution of training examples $(x|l)$. (Schick and Schütze, 2021a)

Combination of PVPs: A key aspect of PET is the combination of PVPs, this is done is order to avoid the hard task of finding text prompts that get good performance in a certain task. For this reason, a technique similar to knowledge distillation (Hinton et al., 2015) is applied. First, a set P of $PVPs$ is defined for a certain task. Then, those $PVPs$ are used as follows (Schick and Schütze, 2021a):

1. For each $p \in P$ PET finetune a different language model M_p. This would imply the training of many language models, however since the scenario assumed in PET is the access to a small training set, the fine-tuning of these models would be cheap, even for many $PVPs$.

2. The ensemble of fine-tuned language models $M = M_p|p \in P$ is used to annotate an unlabeled set of examples D. Then, the unnormalized scores obtained for each example in D are combined as follows:

$$S_m(l|x) = \frac{1}{Z} \sum_{p \in P} w(p) \cdot S_p(l|x)$$

Where $Z = \sum_{p \in P} w(p)$, being $w(p)$ weighting terms for the PvPs. Pet explores the use of $w(p) = 1$, being the combination an average of all the unnormalized scores, or the set of $w(p)$ as the accuracy obtained on the training set before fine-tuning. Once the scores have been combined, the resulting combination is used as input to a softmax function with a temperature $T = 2$ to get a soft probability distribution q.

3. Finally, a standard classifier is trained on the soft-labeled dataset.

Overall, PET has shown to be very effective in few-shot settings, overcoming the difficulties that prompt-based approaches have with small datasets (Perez et al., 2021). This method achieves strong performances in different tasks without manual prompt tuning or hyperparameter optimization in large training sets.

3 PET-NLI

This section provides a detailed overview of the proposed approach called PET-NLI, a training strategy for few-shot text classification using Natural Language Inference Models. This approach reformulates text classification as an entailment problem and follows the same semi-supervised training procedure of PET (Schick and Schütze, 2021a). The architecture of PET-NLI can be seen in Figure 5.

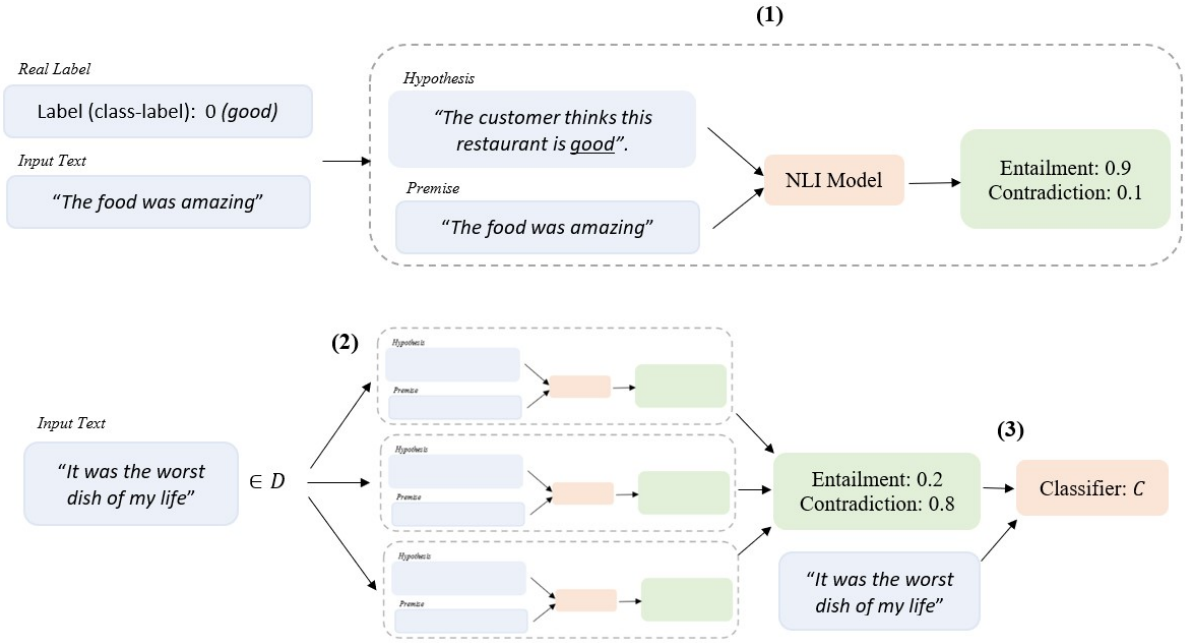

Figure 5: Pet-NLI for text reviews. (1) A pretrained NLI model is finetuned using premises and hypotheses that describe the input texts and the corresponding labels respectively, (2) The ensemble of models trained on different hypothesis templates are used to annotate an unlabeled dataset D , (3) A classifier C is trained on the resulting soft-labeled dataset.

As illustrated in Figure 5 PET-NLI works in different steps, these will be described in a greater detail in the following subsections.

3.1 Individual NLI Models

As a first step, let M be an NLI model with vocabulary V and L the set of labels for the task at hand A. Given a set of input examples $x = (s_1, s_2, ..., s_n)$ and their corresponding labels $l_n \in L$, we define a verbalizer v as an injective function $v : l \rightarrow t$ that maps each label to a textual description t. Thus, we use each input example s_n as a premise p and for the hypothesis h we take the verbalization t and make a textual template describing the label l. We denote as PHP to the premise-hypothesis pair $z = (p, h)$. Therefore, M uses z as input to predict the relation between the premise and the hypothesis, the set of possible outputs of M is represented as $c = (\text{"}entailment\text{"}, \text{"}contradiction\text{"})$.

In order to make the final predictions for text classification, we take slightly different strategies depending on the task at hand A . In the case of binary classification, we use a single hypothesis representing only one of the two classes, if an "entailment" relationship is predicted the label represented by the hypothesis would be chosen, on the other hand, if "contradiction" is predicted the opposite label would be selected as the prediction . For example, as shown in Figure 1, consider the task of text reviews where we have to predict if a customer review is "good" or "bad" for a certain place. Given an input sentence and its label:

$$s_1 : \textit{"The food was amazing"}\ l_1 : 0$$

We use s_1 as the premise and for the hypothesis we first verbalize the label as "good" to then use a textual template to describe the label l_1:

$$\begin{gathered} \textit{Premise} : \textit{"The food was amazing"} \\ \textit{Hypothesis} : \textit{"The customer thinks this restaurant is good."} \end{gathered}$$

Therefore, the task for M would be to use this PHP and predict if the premise entails the hypothesis, if the probability given to the "entailment" relationship is bigger than the "contradiction" probability, the prediction would be "good", if not the predicted label would be "bad" for the given input sentence s_1, this could also be described as follows:

$$m = argmax(Pe(z_1), Pc(z_1))$$

Where m the maximum probability between Pe which is the entailment probability, and Pc that is the contradiction probability given to the PHP z_1 . On the other hand, for classification tasks that have more than 2 classes, we test a single premise against multiple hypotheses, each one representing a different class. For example, consider the same task described previously, but now with three classes (good, bad, terrible), we would represent the premise and the hypotheses as follows:

$$\begin{gathered} \textit{Premise} : \textit{"The food was amazing"} \\ \textit{Hypothesis}_1 : \textit{The customer thinks this restaurant is good.} \\ \textit{Hypothesis}_2 : \textit{The customer thinks this restaurant is bad.} \\ \textit{Hypothesis}_3 : \textit{The customer thinks this restaurant is terrible.} \end{gathered}$$

Thus, to select the predicted label, we take into account only the "entailment" probability given by the model for each premise-hypothesis pair z , and select the highest probability among all, as follows:

$$m = argmax(Pe_1(z_1), Pe_2(z_2), ..., Pe_n(z_n))$$

Therefore, the PHP pair z that obtains the highest entailment probability would be the selected one. The label represented by the hypothesis in the selected PHP will be the final prediction.

3.2 Training and PHP Combination

One of the main challenges of using label textual descriptions in few-shot scenarios, is that we do not have large development sets to find which textual descriptions perform well, in addition, a label could be described in multiple ways and choosing the right one can be a time consuming and error-prone process. Therefore, similar to PET, we approach this challenge using knowledge distillation, this allows us to combine multiple PHPs, each one with a different template for the hypothesis, eliminating the need to find only one among multiple variations. Thus, we define a set P of multiple PHPs , which are used as follows:

1. We finetune a different NLI model for each $PHP \in P$, on the other hand, we denote $M(r|z)_{r \in c}$ as the unnormalized score that the model assigns to each possible relation $r \in c$ given the premise-hypothesis pair z . Thus, given an input x, the score for the label $l \in L$ is given as follows:

$$s_p(l|x) = M(r|p, h)$$

   Then, the normalized scores are obtained by using the softmax activation function:

$$q_p(l|x) = \frac{e^{s_p(l|x)}}{\sum_{l' \in L} e^{s_p(l'|x)}}$$

   For training, we finetune the NLI model using cross-entropy between the predictions $q_p(l|x)$ and the true one-hot distribution of training examples (x, l).

2. Due to this approach assumes the access to a small training set T and a larger set of unlabeled examples D, we use the ensemble of finetune NLI models $M = \{M_z | z \in P\}$ to annotate the unlabeled examples from D. Then, we combine the unnormalized class scores obtained from the models for each example $x \in D$ as follows:

$$S_M(l|x) = \frac{1}{Y} \sum_{z \in P} w(z) \cdot S_p(l|x)$$

   Where $Y = \sum_{p \in P} w(z)$, being $w(z)$ weighting terms for the PHPs. In our approach we use $w(z) = 1$. Once the unnormalized scores have been combined, these are transformed into a probability distribution q using softmax with a temperature of $T = 2$. We collect all pairs (x,q) in a training set Tc.

3. Finally, we finetune a PLM with a sequence classification head C using the soft-labeled dataset Tc, we train the classifier using the Kullback-Leibler (KL) divergence loss between q and the probability distribution $C(l|x)$ given by the classifier C .

4 RAFT

To test our approach in true few-shot scenarios we apply PET-NLI to RAFT (Real World Annotated Few-Shot Tasks) (Alex et al., 2021). RAFT is a benchmark that is focused on tasks that resemble real-world few-shot scenarios, revealing areas where current approaches struggle, like tasks with multiple classes and reasoning over long texts. In addition, its datasets are naturally occurring, have intrinsic value for many real-world applications, possess realistic class distributions, and provide an open-domain setting. This benchmark posses 11 datasets, for each one RAFT provides a training set with input examples, labels, and textual instructions used by human annotators to solve each task, and a test set that is not available, these datasets are described in greater detail below:

4.1 Adverse Drug Effects Corpus V2 (ADE)

The Ade Corpus V2 (Gurulingappa et al., 2012) is a collection of sentences from medical case reports that have been labeled to indicate their association with adverse drug effects. Thus, the task for this dataset is to classify whether a medical report is related or not with negative drugs effects. The dataset has a training set with 50 examples, and 5000 examples for the test set. (Alex et al., 2021) The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:
  Label the sentence based on whether it is related to an adverse drug effect (ADE). Details are described below:
  Drugs: Names of drugs and chemicals that include brand names, trivial names, abbreviations and systematic names were annotated. Mentions of drugs or chemicals should strictly be in a therapeutic context. This category does not include the names of metabolites, reaction byproducts, or hospital chemicals (e.g. surgical equipment disinfectants).
  Adverse effect: Mentions of adverse effects include signs, symptoms, diseases, disorders, acquired abnormalities, deficiencies, organ damage or death that strictly occur as a consequence of drug intake.

- **Training Sample**:
  Sentence: ' No regional side effects were noted . '
  Label: ' not ADE-related (2) '

4.2 Banking 77

The Banking 77 dataset (Casanueva et al., 2020) contains online banking customer service queries labeled with 77 possible intents (e.g. change pin, declined transfer, verify my identity). It contains 50 training examples, and 5000 examples for testing. This dataset provides a big challenge due to some intent categories are partially overlapped in certain

queries, in addition all the possible intents are not shown in the training set, which requires that the designed system should be able to generalize to unseen labels during testing (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:
  The following is a banking customer service query. Classify the query into one of the 77 categories available.

- **Training Sample**:
  Query: ' Is it possible for me to change my Pin?'
  Label: ' change pin (23)'

4.3 NeurIPS impact statement risks (NIS)

The NeurIPS impact statement risks dataset was created by including the broader impact statements of NeurIPS 2020 papers in the dataset collected by Ashurst et al. (2022). These impact statements were annotated based on whether they mention a harmful application or not. This dataset contains 50 examples for training and 150 examples for the test set (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:
  Label the impact statement based on whether it mentions a harmful application of the research done in the paper. Make sure the statement is sufficient to conclude there are harmful applications of the research being done, not a past risk that this research is solving.

- **Training Sample**:
  Paper title: ' Auto - Panoptic : Cooperative Multi-Component Architecture ... '
  Paper link: 'https ://proceedings.neurips.cc/paper/2020/file /... '
  Impact statement: ' This work makes the first attempt to search for all key... '
  ID: '0 ',
  Label: ' doesn't mention a harmful application (1)'

4.4 One Stop English (OSE)

The One Stop English dataset (Vajjala and Lučić, 2018) consists of a collection of articles from the Guardian newspaper, which were labeled by teachers to suit three levels of adult English in the categories of: elementary, intermediate and advanced. Thus, the task for this dataset is to classify the English level shown in each news article. This dataset consists of 50 training examples, and 516 examples for testing (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:
  The following is an article sourced from The Guardian newspaper, and rewritten by teachers to suit three levels of adult English as Second Language (ESL) learners: elementary, intermediate, and advanced. Predict the level of the article.

- **Training Sample**:
  Article: ' For 85 years , it was just a grey blob on classroom maps of the ..... '
  Label: ' intermediate (3) '

4.5 Overruling

The Overruling dataset (Zheng et al., 2021) contains sentences from a law corpus, in which the tasks consists of classifying those as overruling or non-overruling. Overruling is defined as a sentence that nullifies a previous case decision in law. This dataset consists of 50 examples for training and 2350 for testing (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:
  In law, an overruling sentence is a statement that nullifies a previous case decision as a precedent, by a constitutionally valid statute or a decision by the same or higher ranking court which establishes a different rule on the point of law involved. Label the sentence based on whether it is overruling or not.

- **Training Sample**:
  Sentence: ' in light of both our holding today and previous rulings in johnson ...'
  Label: ' overruling (2) '

4.6 Semiconductor org types (SOT)

The Semiconductor org types dataset was created by collecting the name of institutions and papers that have contributed to semiconductor conferences, in which the task is to classify those into the following categories: "University", "Company" and "Research-Institute". This dataset contains 50 examples in the training set and 449 examples for the test set (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:
  The dataset is a list of institutions that have contributed papers to semiconductor conferences in the last 25 years, as catalogued by IEEE and sampled randomly. The goal is to classify the institutions into one of three categories: "university", "company" or "research institute".

- **Training Sample**:
  Paper title: '3Gb/s AC-coupled chip-to-chip communication using a low-swing....'
  Organization name: 'North Carolina State Univ.,Raleigh,NC,US '
  Label: ' university (3)'

4.7 Systematic Review Inclusion (SRI)

The Systematic review inclusion (SRI) dataset (Gurulingappa et al., 2012) contains papers data from meta-review studying interventions, and the tasks consist of identify whether a paper should be included or not in a meta-review to determine which interventions work to promote charitable donations. This dataset contains a training set of 50 examples and a test set with 2243 examples (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:

  Identify whether this paper should be included in a meta-review which includes the findings of systematic reviews on interventions designed to promote charitable donations.
  Papers should be included if they meet all of these criteria:

  1. systematic reviews, scoping reviews, or similar reproducible reviews;
  2. reviews describing monetary charitable donations;
  3. reviews assessing any population of participants in any context; and
  4. peer reviewed and written in English (due to logistical constraints).

  They shouldn't be included if they meet any of these criteria:

  1. primary research reporting new data (e.g., randomised experiments);
  2. non-systematic reviews, theory papers, or narrative reviews;
  3. reviews on cause-related marketing; and
  4. reviews of other kinds of prosocial behaviour (e.g., honesty, non-financial donations like volunteering, blood, or organ donations).

- **Training Sample**:
  Title: ' Prototyping and transforming facial textures for perception research '
  Abstract: ' Wavelet based methods for prototyping facial textures for artific... '
  Authors: ' Tiddeman , B .; Burt , M .; Perrett , D . '
  Journal: ' IEEE Comput Graphics Appl '
  Label: ' not included (2) '

4.8 TAI safety research (TAI)

TAI safety research is a dataset that includes data from the formation of a bibliographic database for research on the safety of transformative artificial intelligence (TAI). This dataset has 50 examples for training and 1639 for testing. Thus, the task consists of classifying whether a work could be defined or not as TAI safety research (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:

  Transformative AI (TAI) is defined as AI that precipitates a transition comparable to (or more significant than) the agricultural or industrial revolution.
  Label a paper as "TAI safety research" if:

  1. The contents of the paper are directly motivated by, and substantively inform, the challenge of ensuring good outcomes for TAI. The paper need not mention TAI explicitly, but it must be motivated by it, since there are far too many papers that are merely relevant to safety. Judging motivation is, unfortunately, inherently subjective, but this is necessary to avoid penalizing papers that do not explicitly mention TAI for appearance reasons, while also not including every paper on, e.g., adversarial examples (which are motivated by capabilities and near-term safety). If the paper would likely have been written even in the absence of TAI-safety concerns, it is excluded. Ultimately, we want to support researchers who are motivated by TAI safety and allow them to find each other's work.
  2. There is substantive content on AI safety, not just AI capabilities. That said, for more speculative papers it is harder to distinguish between safety vs. not safety, and between technical vs. meta, and we err on the side of inclusion. Articles on the safety of autonomous vehicles are generally excluded, but articles on the foundations of decision theory for AGI are generally included.
  3. The intended audience is the community of researchers. Popular articles and books are excluded. Papers that are widely released but nevertheless have substantial research content (e.g., Bostrom's Superintelligence) are included, but papers that merely try to recruit researchers are excluded.
  4. It meets a subjective threshold of seriousness/quality. This is intended to be a very low threshold, and would, for instance, include anything that was accepted to be placed on the ArXiv. Web content not intended for review (e.g., blog posts) is only accepted if it has reached some (inevitably subjective) threshold of notability in the community. It is of course infeasible for us to document all blog posts that are about TAI safety, but we do not want to exclude some posts that have been influential but have never been published formally.

5. Peer review is not required. White papers, preprints, and book chapters are all included.

Otherwise, label it as "not TAI safety research"

- **Training Sample**:
Title: ' Malign generalization without internal search',
Abstract Note: " In my last post, I challenged the idea that inner alignment...",
Url: ' https://www.alignmentforum.org/posts/y ... ' ,
Publication Year: '2020 ',
Item Type: ' blogPost ' ,
Author: ' Barnett , Matthew ' ,
Publication Title: ' AI Alignment Forum ' ,
Label: ' TAI safety research (1)'

4.9 Terms of Service (ToS)

The Terms of service (ToS) dataset (Lippi et al., 2018) contains a set of clauses from Terms of Services of on-line platforms, where the task consists of classifying whether the given clause is unfair or not to consumers. This dataset contains a training set with 50 examples and a test set with 5000 examples (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:

Label the sentence from a Terms of Service based on whether it is potentially unfair. If it seems clearly unfair, mark it as potentially unfair.
According to art. 3 of the Directive 93/13 on Unfair Terms in Consumer Contracts, a contractual term is unfair if: 1) it has not been individually negotiated; and 2) contrary to the requirement of good faith, it causes a significant imbalance in the parties rights and obligations, to the detriment of the consumer.
Details on types of potentially unfair clauses are found below:

1. The jurisdiction clause stipulates what courts will have the competence to adjudicate disputes under the contract. Jurisdiction clauses giving consumers a right to bring disputes in their place of residence were marked as clearly fair, whereas clauses stating that any judicial proceeding takes a residence away (i.e. in a different city, different country) were marked as clearly unfair.
2. The choice of law clause specifies what law will govern the contract, meaning also what law will be applied in potential adjudication of a dispute arising under the contract. Clauses defining the applicable law as the law of the consumer's country of residence were marked as clearly fair. In every other case, the choice of law clause was considered as potentially unfair.

3. The limitation of liability clause stipulates that the duty to pay damages is limited or excluded, for certain kind of losses, under certain conditions. Clauses that explicitly affirm non-excludable providers' liabilities were marked as clearly fair. Clauses that reduce, limit, or exclude the liability of the service provider were marked as potentially unfair when concerning broad categories of losses or causes of them, such as any harm to the computer system because of malware or loss of data or the suspension, modification, discontinuance or lack of the availability of the service. Also those liability limitation clauses containing a blanket phrase like "to the fullest extent permissible by law", were considered potentially unfair. Clause meant to reduce, limit, or exclude the liability of the service provider for physical injuries, intentional damages as well as in case of gross negligence were marked as clearly unfair.
4. The unilateral change clause specifies the conditions under which the service provider could amend and modify the terms of service and/or the service itself. Such clause was always considered as potentially unfair.
5. The unilateral termination clause gives provider the right to suspend and/or terminate the service and/or the contract, and sometimes details the circumstances under which the provider claims to have a right to do so.
6. The contract by using clause stipulates that the consumer is bound by the terms of use of a specific service, simply by using the service, without even being required to mark that he or she has read and accepted them. We always marked such clauses as potentially unfair.
7. The content removal gives the provider a right to modify/delete user's content, including in-app purchases, and sometimes specifies the conditions under which the service provider may do so.
8. The arbitration clause requires or allows the parties to resolve their disputes through an arbitration process, before the case could go to court. It is therefore considered a kind of forum selection clause. However, such a clause may or may not specify that arbitration should occur within a specific jurisdiction. Clauses stipulating that the arbitration should (1) take place in a state other then the state of consumer's residence and/or (2) be based not on law but on arbiter's discretion were marked as clearly unfair. Clauses defining arbitration as fully optional would have to be marked as clearly fair

- **Training Sample**:
  Sentence: ' Crowdtangle may change these terms of service, as described ... ',
  Label: ' potentially unfair (2)'

4.10 TweetEval Hate (TEH)

TweetEval Hate is a dataset that includes the hate-speech detection task from the TweetEval Dataset (Barbieri et al., 2020) , which is a benchmark for tweet classification in English.

Thus, the task consist of given a certain tweet, classify it whether is "hate-speech" or "not-hate-speech". The dataset have 50 training examples and 2966 examples for testing (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:

  Label whether the following tweet contains hate speech against either immigrants or women. Hate Speech (HS) is commonly defined as any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics. Detailed guidelines are provided below, please read before labeling.

  Hate speech against immigrants
  More specifically, HS against immigrants may include:

  – insults, threats, denigrating or hateful expressions
  – incitement to hatred, violence or violation of rights to individuals or groups perceived as different for somatic traits (e.g. skin color), origin, cultural traits, language, etc.
  – presumed association of origin/ethnicity with cognitive abilities, propensity to crime, laziness or other vices
  – references to the alleged inferiority (or superiority) of some ethnic groups with respect to others
  – delegitimization of social position or credibility based on origin/ethnicity
  – references to certain backgrounds/ethnicities as a threat to the national security or welfare or as competitors in the distribution of government resources
  – dehumanization or association with animals or entities considered inferior

  While answering the question "Is this tweet hateful?", you must take into account the following aspects:

  1. the tweet content MUST have IMMIGRANTS/REFUGEES as main TARGET, or even a single individual, but considered for his/her membership in that category (and NOT for the individual characteristics)
  2. we must deal with a message that spreads, incites, promotes or justifies HATRED OR VIOLENCE TOWARDS THE TARGET, or a message that aims at dehumanizing, hurting or intimidating the target

  The joint presence of both elements in a tweet is considered essential to determine whether the tweet has hateful contents, therefore if both of them occur, your answer will be 'Yes'.

In case even just one of these conditions is not detected, HS (at least against immigrants) is assumed not to occur, then your answer will be 'No'.
Here a list of other aspects that are NOT considered hate speech for our purposes:

- HATE SPEECH AGAINST OTHER TARGETS
- offensive language
- blasphemy
- historical denial
- overt incitement to terrorism
- offense towards public servants and police officers
- defamation

Hate speech against women
Label the tweet as hate speech if it is misogynous against women. A tweet is misogynous if it expresses hating towards women in particular (in the form of insulting, sexual harassment, threats of violence, stereotype, objectification and negation of male responsibility).

- **Training Sample**:
  Tweet: ' New to Twitter – any men on here know what the process is to get ... '
  Label: ' not hate speech (2)'

4.11 Twitter Complaints (TC)

The Twitter complaints (TC) dataset includes a set of tweets from nine domains (e.g. software, transport), which are annotated by whether they contain a certain type of complaint or not. This dataset contains a training set with 50 examples and 3399 examples for the test set (Alex et al., 2021). The full instruction given by RAFT to solve this task, along with a training example and its corresponding class label (label) are shown below:

- **Task Instruction**:
  A complaint presents a state of affairs which breaches the writer's favorable expectation. Label the tweet text based on whether it contains a complaint.

- **Training Sample**:
  Tweet text: ' @HMRCcustomers No this is my first job '
  Label: ' no complaint (2) '

Finally, Table 1 shows a general overview of all the datasets on the Raft benchmark.

Dataset Name	**Input Type**	**Train Set Size**	**Test Set Size**	**Number of Classes**
ADE Corpus V2 (ADE)	*'Sentence', 'Label'*	50	5000	2
Banking77 (B77)	*'Query', 'Label'*	50	5000	77
NeurIPS impact statement risks (NIS)	*'Paper Title', 'Paper Link', 'Impact Statement' 'ID', 'Label'*	50	150	2
OneStopEnglish (OSE)	*'Article', 'Label'*	50	516	3
Overruling (Over)	*'Sentence', 'Label'*	50	2350	2
Semiconductor org types (SOT)	*'Paper Title', 'Organization name', 'Label'*	50	449	3
Systematic Review Inclusion (SRI)	*'Title', 'Abstract' 'Authors', 'Journal' 'Label'*	50	2243	2
TAI Safety Research (TAI)	*'Title', 'Abstract Note' 'Url', 'Publication Year' 'Item Type' 'Author', 'Publication Title', 'Label'*	50	1639	2
Terms of Service (ToS)	*'Sentence', 'Label'*	50	5000	2
TweetEval Hate (TEH)	*'Tweet', 'Label'*	50	2966	2
Twitter Complaints (TC)	*'Tweet Text', 'Label'*	50	3399	2

Table 1: Overview of the datasets in the RAFT benchmark

5 Experimental Setup

This section provides a detailed overview of the pretrained language models we have used in our approach, we also show the setup, the chosen datasets, and the textual templates we have made for the hyperparameter tuning process. Then, we describe the application of PET-NLI to the RAFT benchmark, where we describe our chosen strategies for each dataset in order to perform our final experiments. Finally, we present our training monitoring strategies and the chosen metrics to evaluate the performance of PET-NLI.

5.1 Models

PET-NLI finetune multiple language models for its distillation process, due to this, the hyperparameter tuning procedure which requires testing multiples values for different parameters would be very computationally expensive and time demanding. Thus, to reduce this process, we use a small base variant of RoBERTa (Liu et al., 2020) that has been pretrained on multiple NLI datasets, making it a suitable model for our hyperparameter tuning process, the chosen model has been pretrained on SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018) . This model was loaded from the Hugging Face library with the following path: *"cross-encoder/nli-roberta-base"*. Once the optimal hyperameters have been found, for testing our approach on RAFT we use AlBERT (Lan et al., 2020), which has shown (Schick and Schütze, 2021b; IV et al., 2021) to outperform RoBERTa in regular and few-shot settings, in addition we use this model to maintain a fair comparison with the original PET architecture. This model was trained on SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), FEVER (Thorne et al., 2018) and ANLI (Nie et al., 2020), and was loaded from the Hugging Face library with the path: *"ynie/albert-xxlarge-v2-snli-mnli-fever-anli-R1-R2-R3-nli"*. We present a brief description of these models below:

5.1.1 RoBERTa (Robustly Optimized BERT Pre-training Approach)

RoBERTa (Liu et al., 2020) is a large language model developed as a variant of the popular BERT architecture. At its core, RoBERTa is a transformer-based (Vaswani et al., 2017) bidirectional model, which means that it uses an encoder made up of multiple layers of self-attention and feed-forward neural networks. RoBERTA uses the same architecture of BERT, however its differences come from the training process. First, RoBERTa is trained for longer, using bigger batch sizes and a larger dataset. Second, it removes the next sentence prediction objective (NSP) used by BERT during training, this is a binary classification loss to predict whether two segments of text follows each other, this is done to force RoBERTa to focus more on language understanding. Finally, it is trained using dynamic masking with longer input sequences, this means that RoBERTa randomly mask a portion of the input tokens rather than always masking the same portion like BERT, allowing the model to learn more robust representations of the input text. All of these changes led to get better performances in tasks like language understanding and question answering, achieving state of the art results at the time of its development.

5.1.2 AlBERT (A Lite Bert)

AlBERT (Lan et al., 2020) is a bidirectional transformer-based language model, its overall architecture is similar to RoBERTa and BERT, and it was developed with the goal of solving the memory issues of training large language models in GPUs and TPUs, thus, it has a few key differences with the other models. First, it implements factorized embedding parameterization, which is a technique that reduces the number of parameters in the embedding layer by sharing some of the parameters across multiple tokens. Second, it implements a technique called cross-layer parameter sharing, which is a technique that allows sharing the parameters across multiple layers of the encoder, preventing those from growing when the network gets bigger. The application of both techniques have shown to reduce the number of parameters for BERT without hurting its performance. On the other hand, it also implements a self-supervise loss for sentence order prediction, which was done to address the ineffectiveness of the next sentence prediction loss proposed by BERT. At the time of its development, AlBERT showed that with these modifications the model established new state-of-the-art performances on different tasks, beating the performances obtained by BERT and RoBERTa while having fewer parameters.

5.2 Hyperparamenter Tuning

RAFT is the choosen benchmark to test PET-NLI, however because of its datasets does not have development and test sets available, it is difficult to directly test our approach or find the optimal hyparameters for our model. For true few-shot settings, the same hyperparameter values should ideally achieve good performances across different tasks (Perez et al., 2021), thus, to tackle this problem we simulate few-shot scenarios using three other datasets that have their training and evaluation sets available, which makes it more straigthforward to experiment, test our approach and find the best practices and hyperparameters for our model, and then later apply those to RAFT. We experiment with "Yelp Reviews", "AG's news", and "Yahoo Question and Answers", which are datasets that resemble real-world classification tasks in multiple domains. To perform our experiments in these datasets, we tried to simulate as much as possible the scenarios that the model is going to tackle in RAFT. Therefore, for each dataset we generate 3 training sets of 50 examples each, this is done by randomly selecting examples from the original training set, while ensuring the same number of examples for each class. Additionally, we randomly take 1,000 unlabeled examples from the original training set, which will be used as the unlabeled set for the distillation process of PET-NLI. On the other hand, for each dataset we use the original test sets for evaluation. These datasets will be described in a greater detail in the following subsections, along with a description of the verbalization process and the chosen hypotheses templates for each dataset.

Yelp Reviews The Yelp Reviews dataset (Zhang et al., 2015) consist of English text reviews from the Yelp website. It is used as a benchmark for text classification, where the task is to predict the rating as a number of stars given by users to a restaurant in a scale

from 1 to 5. The database has 650,000 training examples in total, 130,000 for each class (number of stars) and 50,000 examples for testing. For example, a text review with its corresponding label is given as follows:

- **Text Review**:*"Love this place. Never had a bad meal. Good portions and great people. Be ready to stand in line."*
  Label:*5 stars*

For using this dataset as input to the NLI models, we define each text review as a premise, and for the hypotheses we first verbalize each label as follows:

$$Verbalizations : 1 \rightarrow terrible, 2 \rightarrow bad, 3 \rightarrow okay, 4 \rightarrow good, 5 \rightarrow great$$

Given the verbalizations and due to the distillation process of PET-NLI we define multiple templates for the labels textual descriptions, which are going to be used as hypotheses. In the case of the given example these templates would be formulated as follows:

- **Template 1**: *"Inference: The customer thinks this restaurant is* ***great****."*
- **Template 2**: *"Inference: The customer thinks this place is* ***great****."*
- **Template 3**: *"Inference: This place is rated as* ***great****."*

Thus, we finetune three NLI models for knowledge distillation, each one using the same input text review as premise and a different "Template" for the hypothesis.

AG's News: AG's News (Zhang et al., 2015) is a classification dataset obtained from news articles on the web. The task is to classify each news article in one of the following categories: World (1), Sports (2), Business (3) and Science/Tech (4), where each news article is conformed by a headline and a text body. The dataset has 120,000 training examples in total, 30,000 for each class and 7,600 examples in the test set. An example of a news article and its label is shown below:

- **Text Review**:*"Car prices down across the board. The cost of buying both new and second hand cars fell sharply over the past five years."*
  Label: *3*

Due to the use of NLI models, we define each text review as a premise, and for the hypotheses we first verbalize each label as follows:

$$Verbalizer : 1 \rightarrow World, 2 \rightarrow Sports, 3 \rightarrow Business, 4 \rightarrow Tech$$

Given the verbalizations and due to the distillation process of PET-NLI we define multiple templates for the labels textual descriptions, which are going to be used as hypotheses. In the case of the given example these templates would be formulated as follows:

- **Template 1**: *"Inference: The topic of this article is* ***Business****."*
- **Template 2**: *"Inference: The category of this article is* ***Business****."*
- **Template 3**: *"Inference: This article is related to* ***Business****."*

Similar to "Yelp", we finetune three NLI models for knowledge distillation, each one using the same input text review as premise and a different "Template" for the hypothesis.

Yahoo Questions and Answers: The Yahoo Answers dataset (Zhang et al., 2015) is a text classification dataset that contains three different fields: 'question title', 'question content' and 'best answer'. For our work, just the 'question title' and the 'best answer' are used. The task consists in classifying those fields among 10 categories, including Society and Culture (0), Science and Mathematics (1), Health (2), Education and Reference (3), Computers and Internet (4), Sports (5), Business and Finance (6), Entertainment and Music (7), Family and Relationships (8), Politics and Government (9) . It has 1,400,000 training examples in total, with 140,000 for each class, and 60,000 examples for testing. An example of a question title, the best answer and its label is shown below:

- **Question Title**:*"Why does not an optical mouse work on a glass table?"*
  Best Answer: *Optical mice use an LED and a camera to rapidly capture images of the surface beneath the mouse. The information from the camera is analyzed by a DSP and used to detect imperfections in the underlying surface .....*
  Label: 4

Similar to the other databases, we define each text review as a premise, and for the hypotheses we first verbalize each label as follows:

$$Verbalizer: 0 \rightarrow Society, 1 \rightarrow Science, 2 \rightarrow Health, 3 \rightarrow Education, 4 \rightarrow Computers$$

$$5 \rightarrow Sports, 6 \rightarrow Business, 7 \rightarrow Entertainment, 8 \rightarrow Relationship, 9 \rightarrow Politics$$

Given the verbalizations and due to the distillation process of PET-NLI we define multiple templates for the labels textual descriptions, which are going to be used as hypotheses. In the case of the given example these templates would be formulated as follows:

- **Template 1**: *"Inference: The topic of this question is* ***Business****."*
- **Template 2**: *"Inference: The category of this question is* ***Business****."*
- **Template 3**: *"Inference: This question is related to* ***Business****."*

Thus, we finetune three NLI models for the distillation process, each one using the same input text review as premise and a different "Template" for the hypothesis.

5.2.1 Setup

As shown in previous sections, PET-NLI fine-tune multiple NLI models for its distillation process, and then use the ensemble of finetune models to annotate an unlabeled set of examples, which creates a bigger dataset where an standard classifier is trained to perform text classification. In order to find the optimal hyperparameters for the NLI models and the final classifier, we first focus on the learning rate, where we test the following values $\{1 \cdot 10^{-4}, 5 \cdot 10^{-4}, 1 \cdot 10^{-5}, 5 \cdot 10^{-5}, 1 \cdot 10^{-6}\}$, then we test the training steps $\{25, 50, 100, 150, 200\}$, and finally with the batch size $\{2, 4, 8, 16, 32\}$. As we first start finding the optimal values for the learning rate, we set the default values for the training steps in 100 and the batch size in 16. Thus, we change each hyperparameter independently, keeping the rest at their default values. We change the learning rates and batch sizes simultaneously for the individual NLI models and the final classifier , on the other hand, the number of training steps is varied only for the individual NLI models, while for the classifier we use 1000 training steps. For finetuning the models we use "run-glue.py" found at *"github.com/huggingface/transformers/tree/main/examples/pytorch/text-classification"*, with standard model parameters, weight decay=0.01, adam beta=0.9, adam beta2=0.98, adam epsilon= $1e^{-6}$. Given that we create three training sets per dataset, we repeat all of our experiments for all of them using a different seed for each one and report average performance (The individual performances are shown in annexes).

5.3 Experiments on RAFT

Once we have found the optimal hyperparameter values for our model , we test our approach on RAFT. To apply PET-NLI on this benchmark we follow a similar setup to the one used during hyperparameter tuning. In general, for the NLI models we use the same input sentences given in each dataset as premises, and as hypotheses we use textual descriptions of the labels, similar to PET and due to the distillation process, we define two types of templates, a simple and a complex one, the first uses a simple textual description using the verbalization of the label, and the second one uses the same textual description but at the beginning of it we append the task instruction given for each dataset, the motivation behind using a task description in the complex template is that we aim the model could better understand the task at hand, give a better performance, and provide a better contribution in the distillation process. It should be emphasized that similar to PET we make some changes to the original task instructions of SRI, TAI, ToS and TEH due to their complexity. Our approach is described for each dataset in greater detail below:

- **Adverse Drug Effects Corpus V2 (ADE)**: Ade corpus V2 is a binary classification task (Ade-related, Ade-not-related), therefore, we use a hypothesis template for only one class, in this case for the "Ade-related" class, thus if the NLI model predicts an entailment relationship the "Ade-related" class will be selected if not the "Ade-not-related" class will be the chosen one, and for the premise, we use the 'sentence' given in the input examples. Our templates for this dataset are shown below:

1. **Simple Template**
   Premise: sentence
   Hypothesis: *' Inference: This sentence is related to an adverse drug effect.'*
2. **Complex Template**
   Premise: sentence
   Hypothesis: Task Instruction.*'Inference: This text is related to an adverse drug effect.'*

- **Banking 77**: This dataset has multiples classes (e.g. change pin, declined transfer, verify my identity), therefore for PET-NLI, we use a different hypothesis for each class, and for the premise we use the 'query' given in the input examples, thus, the premise-hypothesis pair that gets the higher entailment probability will be the selected one to make a prediction. Our chosen templates for an excerpt of all classes are shown below:

  1. **Simple Template**
     Premise: query
     Hypothesis:*' Inference: The correct category for this query is change pin.'*

     Premise: query
     Hypothesis:*' Inference: The correct category of this query is declined transfer.'*

     .....
  2. **Complex Template**
     Premise: Query
     Hypothesis: Task Instruction.*'Inference: The correct category for this query is change pin.'*

     Premise: Query
     Hypothesis: Task Instruction.*'Inference: The correct category of this query is declined transfer.'*

     .....

- **NeurIPS impact statement risks (NIS)**: NIS is a dataset for binary classification (harmful-application, not-harmful application), therefore, we only use a hypothesis template for one class, in this case for the "harmful-application" class, if the NLI model predicts an entailment relationship, the "harmful-application" class will be selected if not, the "not-harmful-application" class will be the chosen one. For the premise we only use the 'Impact Statement' field given in the input examples. Our selected templates for this dataset are shown below:

  1. **Simple Template**
     Premise: Impact Statement

Hypothesis: *'Inference: This impact statement mentions a harmful application.'*

2. **Complex Template**
Premise: Impact Statement
Hypothesis: Task Description.*'Inference: This impact statement mentions a harmful application.'*

- **One Stop English (OSE)**: OSE has three classes, for PET-NLI we set the premise as the field 'article' given in the input example, and due to it has multiple classes we set a different hypothesis for each class, thus the premise-hypothesis pair that gets the higher entailment probability will be the selected one to make a prediction. Our chosen templates for all classes are shown below:

1. **Simple Template**
Premise: article
Hypothesis: *' Inference: The level of this article is advanced. '*

Premise: article
Hypothesis: *' Inference: The level of this article is elementary. '*

Premise: article
Hypothesis: *' Inference: The level of this article is intermediate. '*

2. **Complex Template**
Premise: article
Hypothesis: Task Description.*'Inference: The level of this article is advanced.'*

Premise: article
Hypothesis: Task Description.*'Inference: The level of this article is elementary.'*

Premise: article
Hypothesis: Task Description.*'Inference: The level of this article is intermediate.'*

- **Overruling**: Overruling is a dataset for binary classification (overruling, not overruling), therefore, we only use a hypothesis template for one class, in this case for the "overruling" class, thus if the NLI model predicts an entailment relationship the "overruling" class will be selected, if not the "not-overruling" class will be the chosen one. For the premise we use the field 'sentence' given in the input examples. Our selected templates are shown below:

1. **Simple Template**
Premise: Impact Statement
Hypothesis: *' Inference: This sentence is overruling.'*

2. **Complex Template**
   Premise: Impact Statement
   Hypothesis: Task Description.*'Inference: This sentence is overruling.'*

- **Semiconductor org types (SOT)** The SOT dataset has three classes (Company, Research Institute, University), therefore we set a different hypothesis for each class, and for the premise in this case we make two descriptions, for the first one we append the 'Paper title' (x_1) and the 'Organization name' (x_2) fields given in the input example, and in the second one, we invert the order, we first put the 'Organization name' (x_2) and then append the 'Paper title' (x_1) field, as it is shown below:

  1. **Simple Template**
     Premise: *Organization name:* x_1 *Paper Title:* x_2
     Hypothesis: *' Inference: The category of the institution is Company. '*

     Premise: *Organization name:* x_1 *Paper Title:* x_2
     Hypothesis: *' The category of the institution is Research-Institute. '*

     Premise: *Organization name:* x_1 *Paper Title:* x_2
     Hypothesis: *' The category of the institution is University. '*

     Premise: *Paper Title:* x_2 *Organization name:* x_1
     Hypothesis: *' Inference: The category of the institution is Company. '*

     Premise: *Paper Title:* x_2 *Organization name:* x_1
     Hypothesis: *' The category of the institution is Research-Institute. '*

     Premise: *Organization name:* x_2 *Organization name:* x_1
     Hypothesis: *' The category of the institution is University. '*

  2. **Complex Template**
     Premise: *Organization name:* x_1 *Paper Title:* x_2
     Hypothesis: Task Description.*'Inference: The category of the institution is Company.'*

     Premise: *Organization name:* x_1 *Paper Title:* x_2
     Hypothesis: Task Description.*'Inference: The category of the institution is Research-Institute.'*

     Premise: *Organization name:* x_1 *Paper Title:* x_2
     Hypothesis: Task Description.*'Inference: The category of the institution is University. '*

Premise: *Paper Title:* x_2 *Organization name:* x_1
Hypothesis: Task Description. *' Inference: The category of the institution is Company. '*

Premise: *Paper Title:* x_2 *Organization name:* x_1
Hypothesis: Task Description. *' The category of the institution is Research-Institute. '*

Premise: *Paper Title:* x_2 *Organization name:* x_1
Hypothesis: Task Description. *' The category of the institution is University.'*

- **Systematic Review Inclusion (SRI)**: SRI is a dataset for binary classification (included, not-included), therefore, we only use a hypothesis template for one class, in this case for the "included" class, thus if the NLI model predicts an entailment relationship the "included" class will be selected, if not the "not-included" class will be the chosen one. For the premise, in this case we append three fields given in the input example, the 'Title', 'Abstract', and 'Journal'. In this dataset we briefly modify the task description, this is shown below along with the selected templates.

  - Modified Task Description: Identify whether this paper should be included in a meta-review which includes the findings of systematic reviews on interventions designed to promote charitable donations. Included reviews should describe monetary charitable donations, assess any population of participants in any context, and be peer reviewed and written in English. They should not report new data, be nonsystematic reviews, consider cause-related marketing or other kinds of prosocial behaviour.

  1. **Simple Template**
     Premise: *Title:* x_1 *Abstract:* x_2 *Journal:* x_3
     Hypothesis: *'Inference: This paper should be included in a meta-review which includes the findings of systematic reviews on interventions designed to promote charitable donations'*
  2. **Complex Template**
     Premise: *Title:* x_1 *Abstract:* x_2 *Journal:* x_3
     Hypothesis: Task Description.*'Inference: This paper should be included in a meta-review which includes the findings of systematic reviews on interventions designed to promote charitable donations'*

- **TAI safety research (TAI)**: The TAI dataset has two classes (Tai-safety-research, not-TAI-safety-research), therefore, we only use a hypothesis template for one class, in this case for the "Tai-safety-research" class, thus if the NLI model predicts an entailment relationship, the "Tai-safety-research" class will be selected if not, the "not-TAI-safety-research" class will be the chosen one. For the premise we concatenate the fields of 'Title' x_1 and 'Abstract' x_2 given in the input example. In this

dataset we briefly modify the task description, which is shown below along with our selected templates:

- Modified Task Description:Transformative AI (TAI) is defined as AI that precipitates a transition comparable to (or more significant than) the agricultural or industrial revolution. Label a paper as "TAI safety research" if: 1. The contents of the paper are directly motivated by, and substantively inform, the challenge of ensuring good outcomes for TAI, 2. There is substantive content on AI safety, not just AI capabilities, 3. The intended audience is the community of researchers, 4. It meets a subjective threshold of seriousness/quality, 5. Peer review is not required.

1. **Simple Template**
   Premise: *Title:* x_1 *Abstract:* x_2
   Hypothesis: *'Inference: This paper is a TAI safety research paper.'*
2. **Complex Template**
   Premise: *Title:* x_1 *Abstract:* x_2
   Hypothesis: Task Description.*'Inference: This paper is a TAI safety research paper.'*

- **Terms of Service (ToS)**: ToS is a dataset with two classes (potentially-unfair, not-potentially-unfair), thus, for our approach we only use a hypothesis template for one class, in this case for the "potentially-unfair" class, thus if the NLI model predicts an entailment relationship, the "potentially-unfair" class will be selected if not, the "not-potentially-unfair" class will be the chosen one. For the premise we use the field 'sentence' given in the input example. For this dataset we also briefly modify the task description, this is shown below along with our selected templates:

  - Modified Task Description: Label the sentence from a Terms of Service based on whether it is potentially unfair. If it seems clearly unfair, mark it as potentially unfair. According to art. 3 of the Directive 93/13 on Unfair Terms in Consumer Contracts, a contractual term is unfair if: 1) it has not been individually negotiated; and 2) contrary to the requirement of good faith, it causes a significant imbalance in the parties rights and obligations, to the detriment of the consumer.

  1. **Simple Template**
     Premise: Sentence
     Hypothesis: *'Inference: This sentence is potentially unfair.'*
  2. **Complex Template**
     Premise: Sentence
     Hypothesis: Task Description.*'Inference: This sentence is potentially unfair.'*

- **TweetEval Hate (TEH)**: TEH is a dataset for a binary classification (hate-speech, not-hate-speech), thus, we only use a hypothesis template for one class, in this case

for the "hate-speech" class, thus if the NLI model predicts an entailment relationship the "hate-speech" class will be selected if not the "not-hate-speech" class will be the chosen one. For the premise we use the field 'Tweet' given in the input example. In this dataset we also briefly modify the task description, this is shown below along with the selected templates for this dataset:

- Modified Task Description: Label whether the following tweet contains hate speech against either immigrants or women. Hate Speech (HS) is commonly defined as any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics.

1. **Simple Template**
   Premise: Tweet
   Hypothesis: *'Inference: This tweet contains hate speech against either immigrants or women.'*
2. **Complex Template**
   Premise: Tweet
   Hypothesis: Task Description.*'Inference: This tweet contains hate speech against either immigrants or women.'*

- **Twitter Complaints (TC)**: The TC dataset has two classes (complaint, no complaint), thus, for our approach we only use a hypothesis template for one class, in this case for the "complaint" class, thus if the NLI model predicts an entailment relationship, the "complaint" class will be selected if not, the "no-complain" class will be the chosen one. For the premise we use the field 'Tweet text' given in the input example. Our selected templates are shown below:

  1. **Simple Template**
     Premise: Tweet text
     Hypothesis: *'Inference: This tweet contains a complaint.'*
  2. **Complex Template**
     Premise: Tweet text
     Hypothesis: Task Description.*'Inference: This tweet contains a complaint.'*

5.4 Training Monitoring

As shown in (Perez et al., 2021; Devlin et al., 2019) finetuning pretrained language models with small training sets can be unstable, which usually led to poor performances. Therefore to detect those finetuning issues, we monitor the performance of training runs by evaluating the accuracy on the training set. A training run is considered unsuccessful if the accuracy is below 50%. As it is common to achieve 100% accuracy when finetuning with up to 50 examples, a lower accuracy is a signal of an unsuccessful fine-tuning process. If this

situation occurs we restart training using a different seed or increasing the training steps. This will be described in a greater detail in section 6.2.

5.5 Metrics

For evaluating our models we use two type of metrics, first, for the hyperparameter tuning process we use the 'Accuracy' metric and for the evaluation on RAFT we use 'F1-score', which is the metric used by the benchmark for comparison. These metrics are described in greater detail in the following subsections:

5.5.1 Accuracy

In classification tasks, accuracy is defined as the fraction of examples that were correctly predicted by a certain model. It is defined as the number of correct predictions divided by the total number of examples. For binary classification tasks, this metric could be represented as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Where:
TP: Is the number of instances that are correctly classified as positive by the model.
TN: Is the number of instances that are correctly classified as negative by the model.
FP: Is the number of instances that are incorrectly classified as positive by the model.
FN: Is the number of instances that are incorrectly classified as negative by the model.

5.5.2 F1-Score

The F1-score is a measure of a model's accuracy that takes into account both precision and recall. It is defined as the harmonic mean of precision and recall and is represented as follows:

$$F1 - score = 2 * \frac{Precision \cdot Recall}{Precision + Recall}$$

Where:

Precision measures how well the model avoids false positives and is represented as:

$$Precision = \frac{TP}{TP+FP}$$

Recall measures how well the model finds all the positive instances, is represented as:

$$Recall = \frac{TP}{TP+FN}$$

The F1-score is a balance between precision and recall, and is commonly used as a metric for evaluating the performance of classification models. A high F1-score indicates that the model has a high level of both precision and recall, while a low F1-score indicates that the model needs improvement in at least one of these areas. The F1-score is particularly useful in cases where the class distribution is imbalanced, or when the cost of false positives and false negatives are not equal.

6 Experiments and Results

In this section we show the experiments done to evaluate this approach and to find the optimal hyperparameters for PET-NLI. Then, we show our experiments using the eleven datasets of the RAFT benchmark, as well as a detail overview and discussion of our findings.

6.1 Hyperparameter tuning in few-shot settings

In order to obtain the optimal hyperparameters for PET-NLI we simulate true few-shot scenarios using "Yelp Reviews", "AG's News" and "Yahoo Question and Answers", with these datasets we also aim to investigate the effectiveness of PET-NLI. Thus, as shown in the experimental setup we define three different training sets and hypothesis templates for each dataset, therefore we perform our experiments and report average performance from three individual NLI models, each one using a different hypothesis template for each training set, and from PET-NLI that combines the three NLI models using a distillation process. Table 2, 3 and 4 show our experiments for both approaches using different values for learning rate, batch size and training steps respectively (These tables show the average performance for all the NLI models, the individual performances for each template and training set are shown in annexes). In addition, Figure 6 shows these performances in graphs to show a clearer comparison between PET-NLI and the individual NLI models.

	Individual NLI Models					PET-NLI				
Datasets	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
AG's	82.54	24.93	80.40	**84.84**	70.14	84.59	25.89	82.0	**87.04**	72.23
Yelp	36.25	24.41	41.68	**43.37**	35.31	38.80	26.84	43.89	**46.50**	38.50
Yahoo	66.81	46.28	67.44	**71.33**	62.58	68.09	48.32	69.76	**72.85**	64.68

Table 2: Average performance using different learning rate values.

	Individual NLI Models					PET-NLI				
Datasets	25	50	100	150	200	25	50	100	150	200
AG's	81.53	83.87	**84.84**	84.56	84.06	85.52	86.02	**87.04**	86.85	86.95
Yelp	41.74	42.35	43.37	44.51	**44.74**	43.74	45.35	46.50	46.70	**46.99**
Yahoo	67.27	70.41	**71.33**	71.23	71.01	70.01	72.17	**72.85**	72.38	72.74

Table 3: Average performance using different training steps.

	Individual NLI Models					PET-NLI				
Datasets	2	4	8	16	32	2	4	8	16	32
AG's	76.18	80.95	83.93	**84.84**	82.53	77.86	83.47	86.31	**87.04**	84.61
Yelp	29.89	32.54	35.62	**43.37**	43.08	31.88	35.21	38.33	**46.50**	45.24
Yahoo	62.50	66.81	69.05	71.33	**71.34**	65.57	68.63	70.96	**72.85**	72.81

Table 4: Average performance using different batch sizes.

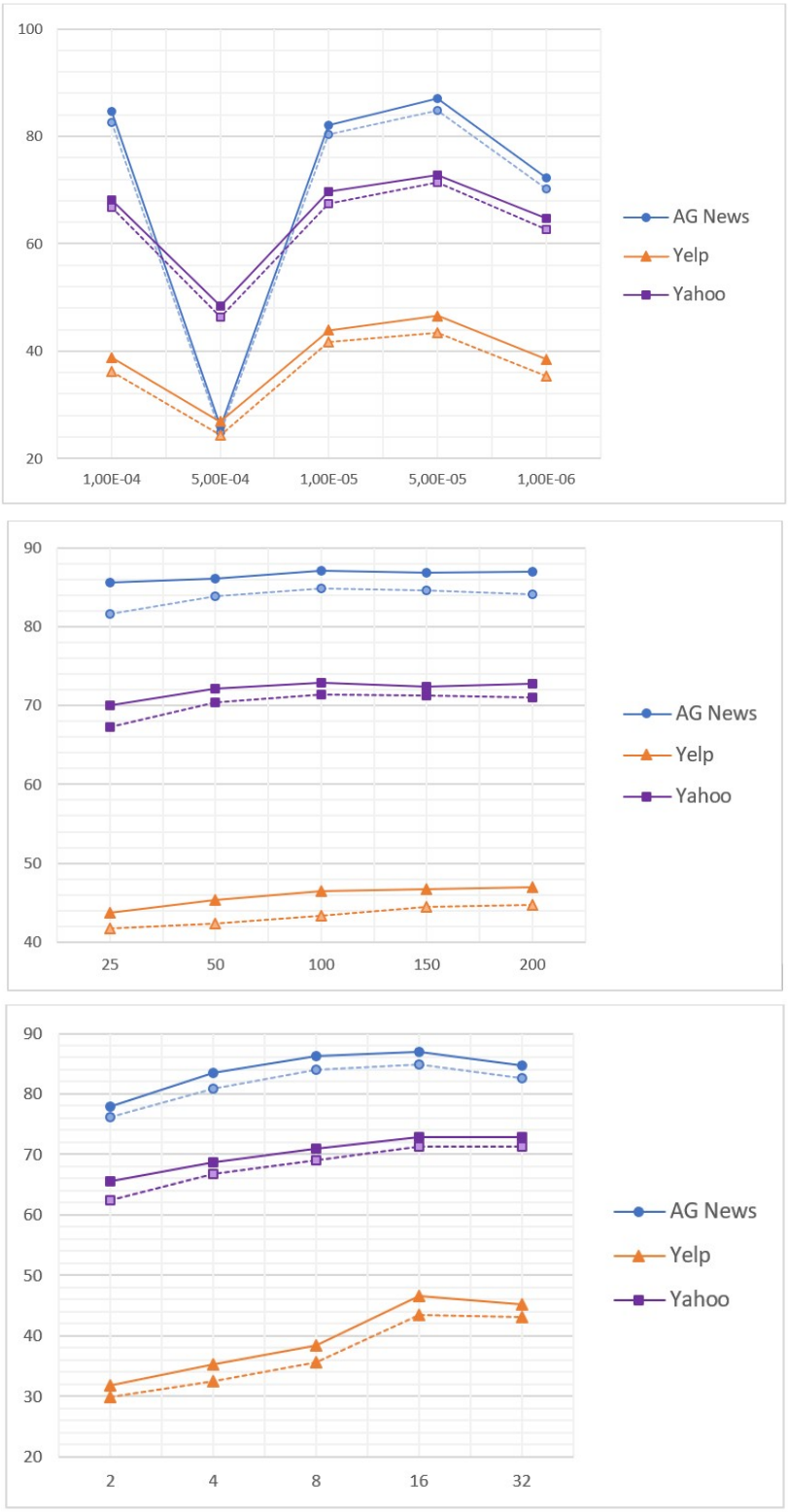

Figure 6: Comparison of performance of PET-NLI (solid lines) and individual NLI models (dotted lines) for different values of learning rate, training steps and batch sizes respectively.

6.1.1 Results

Tables 2, 3 and 4 show the best model performance in bold and the best dataset performance is underlined. Overall, it can be seen that PET-NLI outperforms the individual NLI models in all cases by an average of 2.2% in accuracy , which shows that using knowledge distillation is effective to combine models that use multiple templates, along with the creation of a bigger dataset from unlabeled examples to perform text classification. Thus, the use of different sources of information and bigger datasets shows to be beneficial for getting better performances, therefore, it appears that by either using masked language models in the case of the original PET architecture or NLI models in the case of PET-NLI this training procedure outperforms individual models in text classification. In addition, analyzing the performances on the three datasets, the best results are obtained for AG-News, followed by Yahoo, and lastly Yelp with the worst overall performances. An intuition of why the models obtain poor performances on Yelp could be due to the difficulty on the closeness of its classes (e.g good and great), contrary to AG and Yahoo that have more exclusive ones (e.g Business and Health, Tech and Sports) in addition, in these datasets many examples mention the actual class in its input text, facilitating the classification task in these cases.

Regarding finding the optimal hyperparameters for PET-NLI, the learning rate values of $1 \cdot 10^{-5}$ and $5 \cdot 10^{-5}$ get the best performances, where $5 \cdot 10^{-5}$ gives slightly better results. On the other hand, the training steps show a performance relatively stable across all the tested values, however using 100 training steps is enough to get a good performance. Finally, the batch size values also show a relative stable behaviour except in Yelp, in all datasets, a batch size of 16 provides the best results. Among all hyperparameters, it can be seen that learning rate has the bigger influence in performance, due to it appears to be more sensitive to change, while the training steps and batch sizes show to have less impact, being more stable across different values.

6.2 PET-NLI for RAFT

After finding the optimal hyperparameters, we use our insights to apply PET-NLI to the eleven datasets of the RAFT benchmark. However, for this we make the following changes:

- **Models:** In order to maintain a fair comparision with the original architecture of PET, we use AlBERT *(ynie/albert-xxlarge-v2-snli-mnli-fever-anli-R1-R2-Rt3-nli)* instead of RoBERTA *(cross-encoder/nli-roberta-base)* for our experiments, even though RoBERTA is faster to train, AlBERT has shown to get better performances in few-shot scenarios (Schick and Schütze, 2021b; IV et al., 2021).

- **Training Steps and Monitoring:** Overall, we train the final classifier using 1000 training steps, however in cases where the unlabeled set has more than 4,000 examples we increase the training process to 2000 training steps, otherwise the models do not achieve good performances. There were cases while monitoring the training runs that models using the **ToS** and **OSE** datasets showed to be underfitted, and changing

Method	ADE	B77	NIS	OSE	Over	SOT	SRI	TAI	ToS	TEH	TC	Avg
GPT-2	60.0	12.1	56.1	24.5	49.8	38.0	49.2	61.2	49.8	31.1	72.3	45.8
GPT-Neo	45.2	14.9	40.8	34.3	68.2	40.6	49.3	60.5	56.5	**55.4**	63.6	48.1
AdaBoost	54.3	02.3	62.6	47.5	83.8	45.5	50.6	55.6	56.0	44.3	62.5	51.4
snlt	60.3	24.8	58.5	30.2	83.1	33.6	49.2	62.6	54.0	44.9	79.1	52.8
GPT-3	68.6	29.9	67.9	43.1	**93.7**	76.9	**51.6**	**65.6**	57.4	52.6	82.1	62.7
SetFit	72.6	53.8	**87.2**	52.1	90.7	68.2	49.3	62.8	62.0	53.2	**83.7**	66.9
PET	**82.2**	59.3	85.7	**64.6**	90.8	**81.6**	49.3	63.8	57.6	48.3	82.4	**69.6**
PET-NLI	78.9	**61.9**	59.0	35.9	92.8	75.7	49.3	63.8	**62.4**	50.0	81.4	64.6
Human-Baseline	83.0	60.7	85.7	64.6	91.7	90.8	46.8	60.9	62.7	72.2	89.7	73.5

Table 5: F1 macro scores of PET-NLI and other baselines on the RAFT benchmark. The highest performing model is highlighted in bold, and the best overall performance (including human annotators) is underlined. The last column indicates the average performance across all eleven datasets. These baselines are taken from the RAFT leaderboard *(https://huggingface.co/spaces/ought/raft-leaderboard)*"

only the training seed did not solved the issue, in these cases doubling the number of training steps solved the problem. On the other hand, the **SRI** dataset also showed this issue along with constant predictions, however in this case this was not solved, because it was produced due to the imbalance of its training set, where out of 50 training examples only one of them had a different label than the rest, thus in this case we did not performed any further analysis.

6.2.1 Results and Discussion

Table 5 shows the performance of PET-NLI along with other baselines in the eleven datasets of the RAFT benchmark. Overall, PET-NLI does not outperform PET in the total average performance, obtaining five points less of average F1-score, however it achieves a higher performance in four tasks, which are **B77**, **Over**, **ToS**, and **TEH**, by 2.6, 2.8, 4.8 and 2.3 points respectively, furthemore, it obtains an equal performance on **TAI** and **SRI**.

While it is successful on those tasks, PET-NLI performs poorly compared to PET on **NIS** and **OSE**, which are classified for RAFT as long input tasks, meaning that, these datasets require reasoning over long input texts, thus, the limited context window used by models like AlBERT could be a contributing factor of this performance. In the case of **OSE**, this poor performance was expected, due to the high instability showed by the NLI models during the finetuning process, which was solved by trying multiple training seeds until the models show an adequate training behavior. While with **NIS**, this was not the case, with this task the models were trained without any issues. On the other hand, it should be noted that if we take out those datasets (NIS, OSE), PET-NLI would slightly outperform PET with an average performance of 68.46 vs 68.36 respectively, however, note that the difference might not be statistically significant.

When it is compared to the other baselines, PET-NLI outperforms bigger models like GPT-3 (Brown et al., 2020), but achieves less performance than SetFit (Tunstall et al., 2022). On the other hand, in the case of human performance, PET-NLI outperforms this baseline in **B77**, **Over**, **SRI**, **TAI**, by 1.2, 1.1, 2.5 and 2.9 points of F1-score and achieves almost the same performance on **ToS**. Overall, our approach performs well (compared to the other baselines) on tasks that require domain expertise like **ADE**, **TAI**, **Over** and **ToS**, as well as surprisingly well in tasks with a lot of classes like **B77**, where even though it did not observe all the classes during training, it obtains a good performance in the test set, showing a good generalization of its knowledge to unseen classes. On the other hand, it performs poorly on tasks that require using long input texts like **NIS**, **OSE**, **SRI** and **TAI**. A clearer comparison between PET-NLI, PET and the Human Baseline can be seen in the following figure:

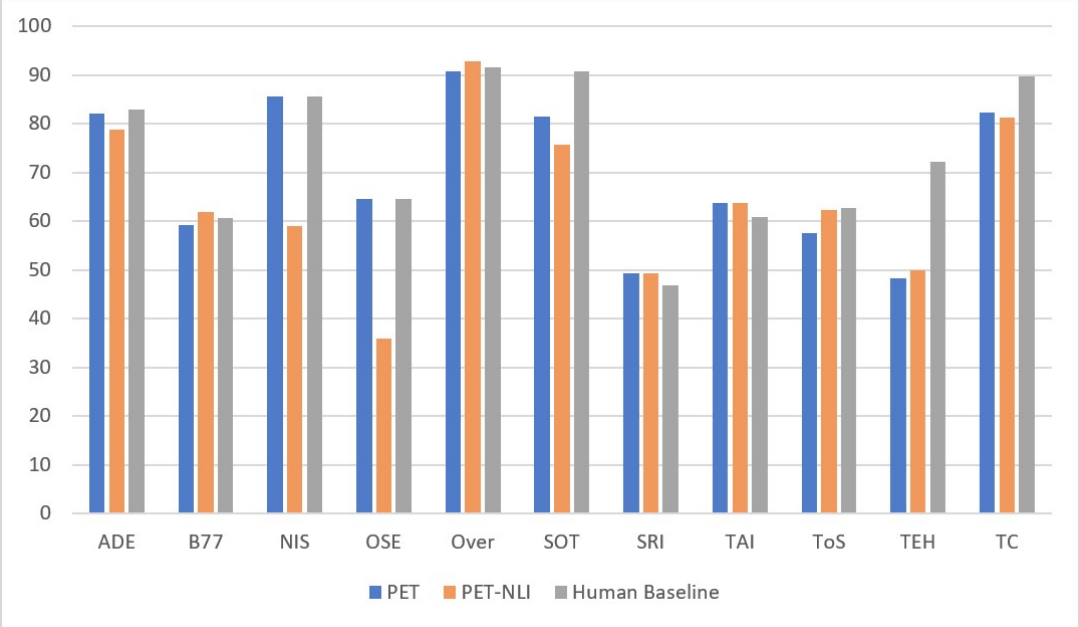

Figure 7: Comparison of performance for RAFT between PET-NLI, PET and the Human Baseline.

7 Conclusions

This work is developed with the aim of testing NLI models to perform few-shot text classification. Specifically, I build PET-NLI, an approach that combines textual instructions with example-based finetuning. Unlike the original PET architecture which uses masked language models trained with prompt-based methods, I use pretrained NLI models which use premises and hypotheses as input to perform text classification. Also, taking inspiration from (Yin et al., 2019; Wang et al., 2021), which show that when pivoting tasks as entailment problems PLMs show to be very effective few-shot learners, we test our approach on true few-shot scenarios presented in the RAFT benchmark.

Regarding our experimental phase, we first use three other english datasets to find the optimal hyperparameters for PET-NLI, verify the effectiveness of the training procedure, and compare its performance against individual NLI models. For this purpose, we use the input texts of each dataset as premises and label textual descriptions as hypotheses, thus, we get an entailment prediction when a premise entails a given hypothesis that represents a certain label, performing in this way text classification. In these scenarios, PET-NLI outperforms the individual NLI models, showing the effectiveness of its architecture and training procedure, to use knowledge distillation and combine the information of multiple label textual descriptions (hypotheses), while individual NLI models are restricted to use only one. As a result, the use of PET-NLI can facilitate the use of hypotheses templates in few-shot settings, avoiding the hard task of trying to find only one that could give a good performance in scenarios with small training sets.

Once we have verified our approach and found the optimal hyperparameters, we apply PET-NLI to RAFT. For this benchmark, regarding the input of the model (premises-hypotheses) we applied the same strategies as our first experimental setup, on the other hand, we change the pretrained language model and make some individual changes on certain datasets regarding the number of training steps of the final classifier. As a result of our experiments, we saw that PET-NLI does not outperform the original PET architecture in the final average performance, however, our approach gets a better or equal performance in 6 out of 11 tasks of RAFT. On the other hand, we show that PET-NLI specifically does not perform well on tasks that use long input texts, mainly in OSE and NIS, and if those are taken out, our approach would slightly outperform PET. Overall, due to our approach uses the same architecture and training procedure as PET, but it changes the type of pretrained language model, we can conclude that in true few-shot scenarios, masked language models trained with prompt-based methods outperform NLI models that use premises and hypotheses as input for text classification. On the other hand, when our approach was compared to bigger architectures like GPT-2 and GPT-3, it showed to get better performances, however it does not outperform others like SetFit. Regarding the human baseline, PET-NLI does not get a better total average performance, but it outperforms this baseline in 4 out of 11 tasks and gets equal performance on 1 task.

Thus, our experimental results show that despite using a different type of models, the benefits of the architecture and training procedure is the same for PET and PET-NLI, mainly demonstrating its efficiency in scenarios with small training sets, showing

that getting good performances is possible without the need of large datasets to perform hyperparameter tuning or find the optimal textual inputs for the model. However, in our approach it is especially difficult to recast as entailment problems tasks that have a lot of classes like B77, because with this approach we need a hypothesis for each class, which implies that for cases like B77 that has 77 classes, the model during testing has to evaluate 77 hypotheses, making this a very inefficient and time-demanding process to evaluate each example, as well as it creates huge testing sets that could be hard to save and use in certain settings. Therefore, it could be considered that this approach of using NLI models for text classification would be an easy and efficient approach only on binary tasks.

As future work, I consider that it would be interesting to train NLI models using sliding attention windows like the ones proposed by Birbird (Zaheer et al., 2020) or Longformer (Beltagy et al., 2020), which would allow the model to use bigger contexts that just 512 tokens, in this way the model could obtain better performances in tasks that require reasoning over long input texts. On the other hand, I believe that is important to create a more efficient way of using NLI models for classification tasks with many classes in order to make the approach more efficient and applicable.

References

Neel Alex, Eli Lifland, Lewis Tunstall, Abhishek Thakur, Pegah Maham, C. Riedel, Emmie Hine, Carolyn Ashurst, Paul Sedille, Alexis Carlier, Michael Noetel, and Andreas Stuhlmüller. Raft: A real-world few-shot text classification benchmark. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/ca46c1b9512a7a8315fa3c5a946e8265-Paper-round2.pdf.

Simran Arora, Avanika Narayan, Mayee F Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Re. Ask me anything: A simple strategy for prompting language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=bhUPJnS2g0X.

Carolyn Ashurst, Emmie Hine, Paul Sedille, and Alexis Carlier. Ai ethics statements: Analysis and lessons learnt from neurips broader impact statements. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 2047–2056, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393522. doi: 10.1145/3531146.3533780. URL https://doi.org/10.1145/3531146.3533780.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.148. URL https://aclanthology.org/2020.findings-emnlp.148.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020. URL https://arxiv.org/abs/2004.05150.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://aclanthology.org/D15-1075.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In Joaquin Quiñonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33428-6.

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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

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)*, pages 3816–3830, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL https://aclanthology.org/2021.acl-long.295.

Harsha Gurulingappa, Abdul Mateen, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*, http://dx.doi.org/10.1016/j.jbi.2012.04.008, 04 2012. doi: 10.1016/j.jbi.2012.04.008.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

Robert L. Logan IV, Ivana Balazevic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. *CoRR*, abs/2106.13353, 2021. URL https://arxiv.org/abs/2106.13353.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Marco Lippi, Przemyslaw Palka, Giuseppe Contissa, Francesca Lagioia, Hans-Wolfgang Micklitz, Giovanni Sartor, and Paolo Torroni. CLAUDETTE: an automated detector of potentially unfair clauses in online terms of service. *CoRR*, abs/1805.01217, 2018. URL http://arxiv.org/abs/1805.01217.

Haokun Liu, Derek Tam, Muqeeth Mohammed, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=rBCvMG-JsPd.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020. URL https://openreview.net/forum?id=SyxS0T4tvS.

Alejandro Martín, Javier Huertas-Tato, Álvaro Huertas-García, Guillermo Villar-Rodríguez, and David Camacho. Facter-check: Semi-automated fact-checking through semantic similarity and natural language inference. *CoRR*, abs/2110.14532, 2021. URL https://arxiv.org/abs/2110.14532.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.441. URL https://aclanthology.org/2020.acl-main.441.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11054–11070. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/5c04925674920eb58467fb52ce4ef728-Paper.pdf.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Mobashir Sadat and Cornelia Caragea. Learning to infer from unlabeled data: A semi-supervised learning approach for robust natural language inference, 2022. URL https://arxiv.org/abs/2211.02971.

Fariba Sadeghi, Amir Jalaly Bidgoly, and Hossein Amirkhani. Fake news detection on social media using a natural language inference approach. *Multimedia Tools Appl.*, 81 (23):33801–33821, sep 2022. ISSN 1380-7501. doi: 10.1007/s11042-022-12428-8. URL https://doi.org/10.1007/s11042-022-12428-8.

Oscar Sainz, Oier Lopez de Lacalle, Gorka Labaka, Ander Barrena, and Eneko Agirre. Label verbalization and entailment for effective zero and few-shot relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1199–1212, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.92. URL https://aclanthology.org/2021.emnlp-main.92.

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*, pages 255–269, Online, April 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL https://aclanthology.org/2021.eacl-main.20.

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*, pages 2339–2352, Online, June 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.185. URL https://aclanthology.org/2021.naacl-main.185.

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)*, pages 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.346. URL https://aclanthology.org/2020.emnlp-main.346.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. Efficient few-shot learning without prompts, 2022. URL https://arxiv.org/abs/2209.11055.

Sowmya Vajjala and Ivana Lučić. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 297–304, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-0535. URL https://aclanthology.org/W18-0535.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Han Wang, Canwen Xu, and Julian McAuley. Automatic multi-label prompting: Simple and interpretable few-shot classification. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5483–5492, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.401. URL https://aclanthology.org/2022.naacl-main.401.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. Entailment as few-shot learner. *CoRR*, abs/2104.14690, 2021. URL https://arxiv.org/abs/2104.14690.

Jason Wei, Chengyu Huang, Soroush Vosoughi, Yu Cheng, and Shiqi Xu. Few-shot text classification with triplet networks, data augmentation, and curriculum learning. *CoRR*, abs/2103.07552, 2021. URL https://arxiv.org/abs/2103.07552.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. Dialogue natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1363. URL https://aclanthology.org/P19-1363.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. 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)*, pages 3914–3923, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1404. URL https://aclanthology.org/D19-1404.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle,

M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help? assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, ICAIL '21, page 159–168, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385268. doi: 10.1145/3462757.3466088. URL https://doi.org/10.1145/3462757.3466088.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *CoRR*, abs/2109.01134, 2021. URL https://arxiv.org/abs/2109.01134.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers, 2022. URL https://arxiv.org/abs/2211.01910.

A ANNEXES

This section shows the individual performances of the NLI models for AG NEWS, YELP AND YAHOO Questions and Answers, with different values of learning rates, training steps, and batch sizes. The following tables show the best performances in bold.

A.1 AG NEWS Hyperparameters

- Template 1: "The topic of this article is _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	82.88	26.39	81.11	**84.07**	71.32
Training Set 2 - Seed 12	80.68	28.69	81.50	**85.0**	69.09
Training Set 3 - Seed 24	83.69	27.45	79.21	**85.30**	73.34

Table 6: AG accuracy values (%) using different learning rates and the first template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	83.03	**85.10**	84.07	83.64	84.04
Training Set 2 - Seed 12	80.97	83.22	85.0	**85.63**	84.90
Training Set 3 - Seed 24	81.32	83.03	**85.30**	84.67	83.85

Table 7: AG accuracy values (%) using different training steps and the first template.

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	77.59	81.0	**84.93**	84.07	83.40
Training Set 2 - Seed 12	76.42	82.38	84.39	85.0	**85.47**
Training Set 3 - Seed 24	75.13	80.45	84.46	**85.30**	82.04

Table 8: AG accuracy values (%) using different batch sizes using the first template.

- Template 2: "The category of this article is _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	84.51	23.42	78.56	**85.31**	70.0
Training Set 2 - Seed 12	81.85	24.0	80.59	**85.50**	67.17
Training Set 3 - Seed 24	84.11	25.34	79.27	**85.68**	71.43

Table 9: AG accuracy values (%) using different learning rates and the second template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	83.89	**85.43**	85.31	83.93	84.28
Training Set 2 - Seed 12	81.48	84.69	85.50	85.53	**85.72**
Training Set 3 - Seed 24	81.67	83.77	**85.68**	85.42	84.82

Table 10: AG accuracy values (%) using different training steps and the second template.

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	75.21	80.35	80.77	**85.31**	83.35
Training Set 2 - Seed 12	77.32	82.64	84.65	**85.50**	82.56
Training Set 3 - Seed 24	76.54	79.26	84.12	**85.68**	81.45

Table 11: AG accuracy values (%) using different batch sizes using the second template.

- Template 3: "This article is related to _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	81.63	22.46	81.88	**83.38**	69.63
Training Set 2 - Seed 12	80.33	23.67	81.34	**84.27**	66.75
Training Set 3 - Seed 24	83.27	23.01	80.25	**85.12**	72.61

Table 12: AG accuracy values (%) using different learning rates and the third template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	82.51	83.25	83.38	**84.71**	83.43
Training Set 2 - Seed 12	78.59	83.06	**84.27**	83.02	84.11
Training Set 3 - Seed 24	80.39	83.34	**85.12**	84.53	84.92

Table 13: AG accuracy values (%) using different training steps and the third template.

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	76.13	81.46	**84.94**	83.38	82.46
Training Set 2 - Seed 12	75.71	80.35	83.56	**84.27**	83.77
Training Set 3 - Seed 24	75.65	80.72	83.62	**85.12**	80.32

Table 14: AG accuracy values (%) using different batch sizes using the third template.

A.2 YELP Hyperparameters

- Template 1: "The customer thinks this restaurant is _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	32.60	20.09	38.92	**46.97**	32.67
Training Set 2 - Seed 12	31.77	25.43	40.60	**43.32**	34.35
Training Set 3 - Seed 24	31.50	23.21	42.38	**44.45**	33.99

Table 15: Yelp accuracy values (%) using different learning rates and the first template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	42.36	44.29	**46.97**	45.98	46.63
Training Set 2 - Seed 12	42.35	37.67	43.32	**45.32**	44.25
Training Set 3 - Seed 24	40.67	42.45	44.45	44.12	**45.42**

Table 16: Yelp accuracy values (%) using different training steps and the first template.

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	30.73	33.23	35.90	**46.97**	46.23
Training Set 2 - Seed 12	30.18	33.62	34.26	43.32	**44.32**
Training Set 3 - Seed 24	28.61	31.31	34.67	44.45	**44.61**

Table 17: Yelp accuracy values (%) using different batch sizes using the first template.

- Template 2: "The customer thinks this place is _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	40.04	22.65	**42.67**	42.51	39.91
Training Set 2 - Seed 12	35.67	26.12	**40.56**	37.65	36.52
Training Set 3 - Seed 24	30.77	24.67	**42.91**	40.57	35.52

Table 18: Yelp accuracy values (%) using different learning rates and the second template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	42.73	44.50	42.51	**45.75**	44.62
Training Set 2 - Seed 12	41.45	40.54	37.65	43.12	**44.52**
Training Set 3 - Seed 24	40.56	44.63	40.57	43.56	**43.78**

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	29.56	33.77	42.37	**42.51**	**42.51**
Training Set 2 - Seed 12	30.84	34.22	35.65	37.65	**39.35**
Training Set 3 - Seed 24	29.71	31.17	33.51	**40.57**	39.35

Table 19: Yelp accuracy values (%) using different batch sizes using the second template.

- Template 3: "This place is rated as _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	40.62	23.44	42.20	**45.70**	37.76
Training Set 2 - Seed 12	40.23	23.86	41.53	**44.34**	36.46
Training Set 3 - Seed 24	43.06	27.32	43.43	**44.34**	36.46

Table 20: Yelp accuracy values (%) using different learning rates and the third template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	43.25	39.29	**45.70**	44.63	44.89
Training Set 2 - Seed 12	42.56	44.18	**44.82**	44.54	43.73
Training Set 3 - Seed 24	39.56	43.67	44.34	43.64	**44.94**

Table 21: Yelp accuracy values (%) using different training steps and the third template.

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	30.32	32.82	34.89	**45.70**	44.62
Training Set 2 - Seed 12	29.21	33.27	34.63	**44.82**	44.35
Training Set 3 - Seed 24	29.94	29.53	34.78	**44.34**	43.12

Table 22: Yelp accuracy values (%) using different batch sizes using the third template.

A.3 YAHOO Hyperparameters

- Template 1: "The topic of this question is _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	71.86	45.18	69.55	**72.44**	61.0
Training Set 2 - Seed 12	69.83	47.32	65.64	**72.05**	63.56
Training Set 3 - Seed 24	68.57	47.84	69.12	**70.21**	62.34

Table 23: Yahoo accuracy values (%) using different learning rates and the first template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	68.45	71.37	72.44	**73.51**	72.82
Training Set 2 - Seed 12	68.12	70.53	**72.05**	70.34	71.26
Training Set 3 - Seed 24	66.57	68.35	70.21	**71.62**	70.42

Table 24: Yahoo accuracy values (%) using different training steps and the first template.

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	64.32	67.41	70.35	**72.44**	71.21
Training Set 2 - Seed 12	63.65	66.32	69.41	72.05	**72.52**
Training Set 3 - Seed 24	61.02	69.25	68.34	**70.21**	69.43

Table 25: Yahoo accuracy values (%) using different batch sizes using the first template.

- Template 2: "The category of this question is _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	60.86	44.77	68.72	**73.26**	62.63
Training Set 2 - Seed 12	59.31	47.78	65.74	**71.17**	63.42
Training Set 3 - Seed 24	60.27	46.96	66.21	**70.50**	64.42

Table 26: Yahoo accuracy values (%) using different learning rates and the second template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	68.23	73.23	**73.26**	72.63	72.51
Training Set 2 - Seed 12	67.74	70.64	**71.17**	69.62	70.25
Training Set 3 - Seed 24	65.21	69.45	70.50	70.46	**70.61**

Table 27: Yahoo accuracy values (%) using different training steps and the second template.

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	65.28	70.24	71.52	**73.26**	72.75
Training Set 2 - Seed 12	62.53	68.31	69.14	**71.17**	70.51
Training Set 3 - Seed 24	60.50	65.67	67.34	70.50	**71.38**

Table 28: Yahoo accuracy values (%) using different batch sizes and the second template.

- Template 3: "This question is related to _ _ _ _ _."

	Learning Rates				
Training Runs	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
Training Set 1 - Seed 0	**72.82**	45.86	69.13	70.26	61.35
Training Set 2 - Seed 12	67.35	43.18	63.32	**69.16**	62.67
Training Set 3 - Seed 24	70.46	47.64	69.67	**73.03**	61.56

Table 29: Yahoo accuracy values (%) using different learning rates and the third template.

	Training Steps				
Training Runs	25	50	100	150	200
Training Set 1 - Seed 0	69.36	68.45	70.26	**72.52**	71.46
Training Set 2 - Seed 12	66.27	70.32	69.16	**70.61**	70.52
Training Set 3 - Seed 24	65.57	71.46	**73.03**	69.72	69.35

Table 30: Yahoo accuracy values (%) using different training steps and the third template.

	Batch Size				
Training Runs	2	4	8	16	32
Training Set 1 - Seed 0	64.62	68.64	68.51	70.26	**71.67**
Training Set 2 - Seed 12	60.68	66.32	68.25	69.16	**70.21**
Training Set 3 - Seed 24	59.94	64.17	68.64	**73.03**	71.46

Table 31: Yahoo accuracy values (%) using different batch sizes using the third template.