Project Proposal: One-Shot Learning for Language Modelling

Talip Uçar
talip.ucar.16@ucl.ac.uk

Adrian Gonzalez-Martin
adrian.martin.18@ucl.ac.uk

Matthew Lee
matthew.lee.16@ucl.ac.uk

Adrian Daniel Szwarc
adrian.szwarc.18@ucl.ac.uk

Abstract

Humans can infer a great deal about the meaning of a word, using the syntax and semantics of surrounding words even if it is their first time reading or hearing it. We can also generalise the learned concept of the word to new tasks. Despite great progress in achieving human-level performance in certain tasks (Silver et al., 2016), learning from one or few examples remains a key challenge in machine learning, and has not thoroughly been explored in Natural Language Processing (NLP). In this work, we will define meta, zero-shot, one-shot and few-shot learning on an NLP task; and explore one-shot, and few-shot learning using the WikiText-2 (WT2) dataset with the help of a meta-learning procedure. We take (Vinyals et al., 2016) as our baseline, and explore the problem using different distance metrics (cosine, Euclidean, and Poincaré), as well as experimenting with attention mechanisms. Moreover, one of the key reasons for the fast progress in Computer Vision (CV) in this field was the existence of standardised benchmarks. NLP is still missing a reliable benchmark for one-shot, and few-shot learning. Thus, we attempt to develop a benchmark for the benefit of the larger research community.

1 Introduction

In the last few years there has been great progress in the applications of deep neural networks, especially in CV and NLP. But their success has mostly relied on the availability of large datasets and these networks require re-training on new data for new tasks. These shortcomings of the current state of the art neural networks make us wonder: can we do better? Can we build networks that can learn from few examples? Although there have been some recent works in this domain, they mostly approached this problem in CV, with a few exceptions in NLP (Vinyals et al., 2016; Lampinen and McClelland, 2017). To our knowledge, (Vinyals et al., 2016) is the first instance where one-shot learning in NLP is clearly defined on a language problem. Our work will take (Vinyals et al., 2016) as a baseline to explore one-shot, and few-shot learning in NLP. To put our discussion on a solid ground, we need to define the important concepts.

We define meta learning as ”learning to learn” (Bengio et al., 1992; Thrun and Pratt, 1998), or ”learning to learn words” in the context of this work. It is a process of developing a good representation for any word. For example, hearing “Apple” could refer to a fruit or a company, depending on the context.

Zero-shot learning is being able to solve a task despite not having received any training examples of that task. One-shot is being able to learn from a single labelled example whereas few-shot uses a few labelled examples. These small training data regimes employ meta-learning strategies to be effective.

How we do learning in k-shot learning is a little different than the typical learning scenario. Each sample-target pair in typical supervised learning is considered as a training point. In the case of few-shot learning, for example, every new sample is considered a task in itself. The dataset is split into two parts, a support set S and a prediction set P, i.e. (S,P) for both in training, and testing. In a k-shot N-class classification task, support set S contains k labelled examples for each of the N classes.

In our work we will carry out the following one-shot learning task: given a query sentence with a missing word and a support set of sentences, in which each sentence has a missing word and a corresponding 1-hot label, find a label within the support set that matches the missing word of the query sentence the best. This will be discussed in more detail in Section 3, Project Proposal.

2 Literature review

Discussion of one-shot learning appears in the psychology literature in the late 80s in (Biederman, 1987). Its history in machine learning concerns mainly computer vision, in which the oldest reference we can find is in (Burl et al., 1998) for a Bayesian approach to object recognition. There is a spike in interest generated by (Fei-Fei et al., 2006) for further vision work on object categories. And with the recent resurgence of deep learning, a new body of work on one-shot, and few-shot learning has started to emerge. There are three main approaches to one-shot, and few-shot learning:

a) **Metric-based**. Focuses on distance metrics (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018)

b) **Model-based**. Uses recurrent networks with attention as well as external or internal memory (Santoro et al., 2016; Munkhdalai and Yu, 2017)

c) **Optimization-based**. Optimizes the model parameters explicitly (Ravi and Larochelle, 2017; Finn et al., 2017; Nichol et al., 2018).

Table 1 summarises these observations.

We will narrow our discussion to metric-based models that happen to combine some of other ideas such as attention, and embeddings. In the following sub-sections, we will discuss the aforementioned four one-shot meta-learning papers listed under metric-based approaches in chronological order of their publication date as well as improving performance. The performance of all models is shown in table 2 in Section-3.

2.1 Convolutional Siamese Neural Network

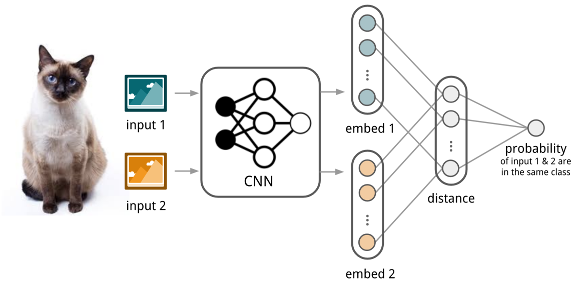


Figure 1: Siamese Convolutional Networks

Source: (Weng, 2018)

This is an application of the Siamese networks of (Bromley et al., 1994) by (Koch et al., 2015) to the task of one-shot learning. It consists of two identical neural networks (twin networks) that shares the same weights and parameters, and are trained to predict whether two input images are from the same class, or not. Some of the key factors in this work can be summarised as follows:

- It uses convolutional layers to implement an embedding function $f(x, \theta)$ to encode image x into feature space, where θ are learned parameters for encoding.
- $L1$ distance, $\|f(x_i, \theta) - f(x_j, \theta)\|_1$, is used as a similarity measure between two embeddings, $f(x_i, \theta)$ and $f(x_j, \theta)$.
- A final feed-forward layer with sigmoid function converts this distance to a probability:

$$\boldsymbol{p} = \sigma(W \|f(x_i, \theta) - f(x_j, \theta)\|_1) \qquad (1)$$

- Cross-entropy loss is used since this is a binary classification task.

This work was published before the meta-learning framework described in the introduction was conceived. Instead, the authors use their Siamese network as a pretrained network. They apply the pretrained Siamese network to new tasks just as in regular transfer learning. This method is weaker because it looses its effectiveness as new tasks become more and more different to the original one. To the best of our understanding, there is no reason why Siamese networks could not be trained in the meta-learning framework with potential performance gains, and potentially with different distance metrics.

2.2 Matching Networks

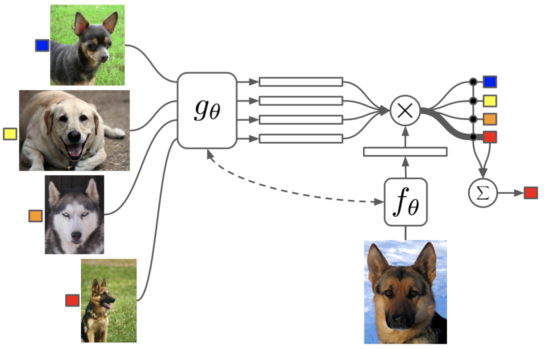


Figure 2: Matching Networks

Source: (Vinyals et al., 2016)

	Model-based	**Metric-based**	**Optimization-based**
Key idea	RNN, memory	Embeddings, distance metric	Training optimization
How is $P(\boldsymbol{y} \mid \boldsymbol{x})$ modeled?	$f(\boldsymbol{y}, \mathcal{S})$	$\sum_i k(\boldsymbol{x}, \boldsymbol{x}_i)\boldsymbol{y}_i$	$P(\boldsymbol{y} \mid \boldsymbol{x})$

Table 1: Summary of different approaches to Few-Shot Learning

We describe the next paper in more depth as it forms the basis of our project. (Vinyals et al., 2016) proposes matching networks for one-shot meta-learning. They also propose the meta-learning framework described in the introduction. Their task is to learn a classifier for any given support set $S = \{x_i, y_i\}_{i=1}^{k}$ (k-shot classification). This classifier defines a probability distribution over output labels y given a test example x. For the NLP case, the task is finding the missing word in a sentence: given a query sentence with a missing word, and a support set, S, of sentences, in which each sentence has a missing word and a corresponding 1-hot label, find the label from the support set that matches the query sentence the best.

There are few key parts to their model that we identify:

- The training process is designed to match inference at test time.
- Output is defined as a sum of labels of support samples weighted by an attention kernel $a(x, x_i)$ which is a measure of similarity between test sample x and support sample x_i.

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \tag{2}$$

- The attention kernel uses two embedding functions, f and g, for encoding the test sample and the support set samples respectively. Cosine similarity, c(.), between the embedding vectors of two data points is used as the attention weight. They use softmax function to normalize cosine similarities:

$$a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}), g(x_i))}}{\sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}} \tag{3}$$

- They improve the embedding functions by taking as input the whole support set S together with the original input, so that the learned embedding is adjusted based on the relationship with other support samples.
- The improved embeddings employ a bidirectional LSTM, with S treated as a sequence. The ith stage of the LSTM has the input $g'(x_i)$ - the simple embedding g of the ith training example described above, where we use the prime to distinguish this from $g(x_i, S)$. The LSTM is also a function of the last hidden state $\overrightarrow{h}_{i-1}$ and last output vector $\overrightarrow{c}_{i-1}$ as usual. Thus backward and forward we have:

$$\overrightarrow{h}_i, \overrightarrow{c}_i = \texttt{LSTM}(g'(x_i), \overrightarrow{h}_{i-1}, \overrightarrow{c}_{i-1}) \tag{4}$$
$$\overleftarrow{h}_i, \overleftarrow{c}_i = \texttt{LSTM}(g'(x_i), \overleftarrow{h}_{i-1}, \overleftarrow{c}_{i-1}) \tag{5}$$

- An additional, related issue is that we might also want S to affect how we embed the test image. Similar to before, $f(\hat{x})$ becomes $f(\hat{x}, S)$. Specifically, $f(\hat{x}, S)$ is a one way LSTM with an attention mechanism that unfolds over K steps:

$$f(\hat{x}, S) = \texttt{attLSTM}(f'(\hat{x}), g(S), K) \tag{6}$$

where like g', f' is our initial embedding from some of other neural network; $g(S)$ represents $g(x_i, S)$ already applied to all x_i and K is the number of steps as noted. After K steps we have:

$$\hat{h}_k, c_k = \texttt{LSTM}(f'(\hat{x}), [h_{k-1}, r_{k-1}], c_{k-1}) \tag{7}$$
$$h_k = \hat{h}_k + f'(\hat{x}) \tag{8}$$
$$r_{k-1} = \sum_{i=1}^{|S|} a(h_{k-1}, g(x_i)) g(x_i) \tag{9}$$
$$a(h_{k-1}, g(x_i)) = \texttt{softmax}(h_{k-1}^{T} g(x_i)) \tag{10}$$

- This $a(.)$ is a different attention mechanism to the one already mentioned, and is known as "content" based attention; r_{k-1} is the "readout" from $g(S)$ - we want to be able to read the training embeddings to influence the test embeddings; and $[h_{k-1}, r_{k-1}]$ implies concatenation of h_{k-1} and r_{k-1}.
- The training objective of matching networks. Consider a set of labels L, say (”cats”, ”dogs”, ”fish”, ”lions”, ”snakes”), sampled

from task T, say, classifying animals. Formally, T is a distribution over possible label sets L. We sample from L to create the support set S and a prediction set P both of which contain labelled examples from L. Let θ be the parameters of the model, and P_θ a probability distribution, not the prediction set. Then the matching net is trained to minimise the error predicting P's labels, conditioned on S with the following training objective:

$$\theta = \underset{\theta}{\operatorname{argmax}} E_{L\sim T}\left[E_{S\sim L, P\sim L}\left[K\right]\right] \qquad (11)$$

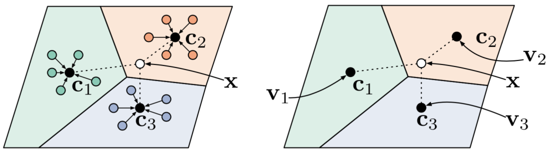
where

$$K = \sum_{(x,y)\in P} \log P_\theta(y \mid x, S) \qquad (12)$$

- Crudely, the inner E is for the training and the outer E is for the (meta-)training, though it is a little more subtle than this (see project outline for more details). After this, meta-testing will occur.

In addition to various CV tasks, they apply their model and training strategy to the language task of inferring the missing word from a sentence on the Penn Treebank dataset (Marcus et al., 1993a). They achieved accuracies of 32.4%, 36.1% and 38.2% for 1, 2 and 3-shot, respectively. They compared this to an Oracle LSTM language model (Zaremba et al., 2014) which achieved 72.8%. The oracle is not doing one-shot learning and can see all the data, so its result is treated as an upper bound. They did not try to rebuild and modify any competing models to compare to matching networks for the NLP task.

2.3 Prototypical Networks



(a) Few-shot

(b) Zero-shot

Figure 3: Prototypical Networks

Source: (Snell et al., 2017)

The prototypical networks of (Snell et al., 2017) use some of the same insights as (Vinyals et al., 2016) but they arrive at a much simpler one-shot model. Some key aspects of their model include:

- A prototype for each class which is the mean of the vectors in the embedded support for that class.

$$\boldsymbol{c}_k = \frac{1}{|S_k|} \sum_{(\boldsymbol{x}_i, y_i)\in S_k} f_\phi(\mathbf{x_i}) \qquad (13)$$

- Queried (test) points x are classified based on their distances to prototypes, and a distribution over the classes is made by taking the softmax of those distances, where a shorter distance is better:

$$p_\phi(y = k \mid \boldsymbol{x}) = \frac{\exp(-d(f_\phi(\boldsymbol{x}), \boldsymbol{c}_k))}{\sum_{k'} \exp(-d(f_\phi(\boldsymbol{x}), \boldsymbol{c}_{k'}))} \qquad (14)$$

- Learning is by SGD where the objective is to minimise the negative log-probability of the true class.

$$J(\phi) = -\log p_\phi(y = k \mid \boldsymbol{x}) \qquad (15)$$

- They use the learning framework of (Vinyals et al., 2016).

Prototypical networks are performing clustering. When a Bregman divergence is used as the distance metric, the Prototypical Network is equivalent to a mixture density estimation on the support set. The choice of distance measure fixes the family of distributions that the equivalent mixture density is using to cluster, thus the choice of distance is important because it suggests something about the underlying distribution modelling the clusters in the embedding space.

When a Euclidean distance (an example of a Bregman divergence) is used, the model is equivalent to a linear model. Euclidean distances are found to work well in practice, and this could be because all the required non-linearity is applied at the embedding stage. Note the cosine distance used by (Vinyals et al., 2016) is not a Bregman divergence.

The authors note that prototypical networks and matching networks are equivalent in the case of one-shot learning. Both can be reduced to k-nearest neighbours (k-NN).

2.4 Relation Network

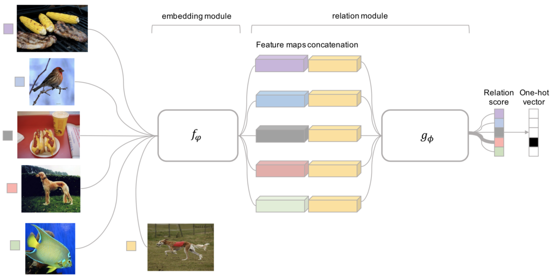


Figure 4: Relation Network

Source: (Sung et al., 2018)

Finally we look at the relation networks of (Sung et al., 2018). The salient features of the model are:

- The model consists of two modules. An embedding module f_ϕ and a relation module g_ψ.
- Learning proceeds as in (Vinyals et al., 2016).
- Testing x_i and training x_j samples (examples) are passed through the embedding module to create feature maps $f(x_i)$ and $f(x_j)$,
- These are then combined with some operator $\mathcal{C}(f(x_i), f(x_j))$, which in their case was a concatenation in depth.
- This combined feature map is passed through the relation module which produces a number between 0 and 1 denoting similarity and known as the relation score $r_{i,j} = g_\psi(\mathcal{C}(f(x_i), f(x_j)))$. For the 1-shot N-class task, N relation scores are generated for the relation between one test input x_j and the N training set examples.
- The objective function is the mean square error, because the prediction of relation scores is a regression problem:

$$\psi, \phi \leftarrow \underset{\psi,\phi}{\operatorname{argmin}} \sum_{i=1}^{m} \sum_{j=1}^{n} (r_{i,j} - \mathbf{1}(u_i == y_j))^2 \quad (16)$$

What might not be obvious is that the embedding and distance metric are both "learnable". f and g will be whatever functions minimise the loss over the many one-shot test and train examples passed through.

As can be seen in Table 2 printed in (Sung et al., 2018), relation networks outperform competing one-shot approaches on most tasks. Even this has been beaten. The recent (Rusu et al., 2019) combines ideas from the optimisation style meta-learning papers we did not study here and relation networks to beat even the relation networks with a Latent Embedding Optimisation model (LEO).

3 Project Proposal

Our work will attempt to improve on the baseline set in k-Shot Learning on Language Modelling by the matching networks architecture (Vinyals et al., 2016). Even though it is not State-of-the-Art in general, to the best of our knowledge this architecture is the only one which has been applied so far to Language Modelling tasks, thus it sets the baseline to try and improve upon. An extended research project would aim to rebuild and compare each of the one-shot models to date. Our project represents the first part of this broader research.

This section will start by discussing the potential modifications which we believe can improve the performance of the model, to then outline the task which will be used as a benchmark to compare between the baseline and different iterations of the model. We will finally discuss the particular steps we are going to take in order to achieve the project's goal.

3.1 Improvements on matching networks

Matching networks is a non-parametric approach to k-shot learning which tries to find an embedding of an input sentence into its support set to then compare how similar it is to each one using the cosine distance. Though it is a neural network, its attention mechanism where distances are employed is non-parametric.

Our research hypothesis is that we can improve on this model by

- Using Euclidean distance instead of cosine similarity.
- Using BERT embeddings (Devlin et al., 2018) to encode input sentences instead of one-hot vectors.
- Using a new similarity measure, perhaps by experimenting with the Poincaré ball (a specific realisation of hyperbolic space) for word vectors, and comparing its performance against its Euclidean counterpart on this task.

Using the Euclidean distance is suggested by (Snell et al., 2017) as one of the main sources of performance improvement over matching networks. The suspected reason (hinted by the authors) is that the Euclidean distance is a type of Bregman divergence metric, and thus the algorithm becomes equivalent to Mixture Density estimation. This leads us to believe that changing the distance metric may help to improve the results.

	1-Shot 5 Class	5-Shot 5 Class	1-Shot 20 Class	5-Shot 20 Class
Siamese Networks	98.4%	99.6%	95.0%	98.6%
matching networks	97.9%	98.7%	93.5%	98.7%
Prototypical Networks	$\mathbf{99.8}\%$	$\mathbf{99.7}\%$	96.0%	98.9%
Relation Networks	$\mathbf{99.6 \pm 0.2}\%$	$\mathbf{99.8 \pm 0.1}\%$	$\mathbf{97.6 \pm 0.2}\%$	$\mathbf{99.1 \pm 0.1}\%$

Table 2: Accuracy results of different Few-Shot models for the Omniglot dataset, as stated in (Sung et al., 2018)

.

Moreover, geometries other than Euclidean space for distributed representations have been investigated recently (Nickel and Kiela, 2017), in which an n-dimensional Poincaré ball is shown to be good at encapsulating hierarchical structure of text and graphs because this space is considered as a continuous approximation to trees. If we have enough time, we will experiment with Poincaré distance. However, to get a meaningful result, our embeddings need to come from Poincaré space as well. So, perhaps, we will need to use something similar to Poincaré Embeddings in (Nickel and Kiela, 2017) if we see fit.

We will thus update the original attention mechanism of Equation (3) as

$$a(\boldsymbol{x}, \boldsymbol{x}_i) = \frac{\exp(-d(x, x_i)}{\sum_j \exp(-d(x, x_j))}$$

Where we define $d(x, x_i)$ as

$$d(x, x_i) = \begin{cases} \|\boldsymbol{x}, \boldsymbol{x}_i\| \,, & \text{I} \\ arccosh(1 + 2\frac{\|\boldsymbol{x}, \boldsymbol{x}_i\|^2}{(1-\|\boldsymbol{x}\|^2)(1-\|\boldsymbol{x}_i\|^2)}), & \text{II} \end{cases}$$

Where Equations I and II correspond to Euclidean and Poincaré distances, respectively.

On the other hand, we think that combining the non-parametric nature of matching networks with other parametric models (like pre-trained word embeddings) seems like a powerful approach. The original paper used simple encodings which were then sum-pooled, thus it seems safe to suggest that using embeddings which may already encode part of the *"similarity"* between words may improve the baseline. This approach is, indeed, suggested by the authors as a potential improvement on the model (Vinyals et al., 2016).

Note that, in order to keep the comparison fair, we will pay attention to use embeddings that haven't been pre-trained on the same source dataset, and therefore haven't already *"seen"* the missing words.

3.2 k-Shot Language Modelling

To measure how well each model performs, we will use the k-Shot Language Modelling task described in the original paper (Vinyals et al., 2016). We first define a support set S as

$$S = \{(\boldsymbol{x}, \boldsymbol{y})\}_{i=1}^{kN}$$

where $\boldsymbol{x}$ represents a sentence with a missing word, and $\boldsymbol{y}$ represents the word which fills that gap. The support set will have N different labels, with k examples each. The task is then to, given a new sentence $\boldsymbol{x}$ with a missing word in it, find the corresponding word $\boldsymbol{y}$ from within the N labels present in the support set S, so that each of these trials becomes a N-way choice between the support labels. An example of the task is shown in Figure 5.

Note that, without loss of generality, we can also equally consider the case where instead of predicting a new input $\boldsymbol{x}$ we predict the missing words on test set of inputs $P = \{\boldsymbol{x}_i\}$.

On our project we will focus on the setups with $k = \{1, 5\}$ and $N = \{5, 20\}$. That is, One-Shot and Five-Shot Learning with support sets of 5 and 20 labels.

3.3 Training and testing

To follow the *"meta-learning"* approach outlined in Section 2, we will perform this basic task across a *"meta-train"* dataset, where each *"episode"* j will have its own support set S_j and test set P_j. Each of these episodes will have its own set of N labels which won't overlap with other episodes, as well as its own set of test examples.

After we have trained our classifier across all episodes (or after we reach a certain accuracy to avoid overfitting) we will then test it on a *"meta-test"* set with its own set of inputs and labels. Testing on these meta-test sets can be performed in one of two ways

$$\boldsymbol{x}_1 = [\texttt{The}, \langle\ldots\rangle, \texttt{shot}, \texttt{the}, \texttt{duck}, \texttt{.}] \qquad \boldsymbol{y}_1 = \texttt{hunter}$$

$$\vdots \qquad\qquad \vdots$$

$$\boldsymbol{x}_N = [\texttt{There}, \texttt{is}, \texttt{a}, \langle\ldots\rangle, \texttt{in}, \texttt{my}, \texttt{boot}, \texttt{!}] \qquad \boldsymbol{y}_N = \texttt{snake}$$

$$\boldsymbol{x} = [\texttt{The}, \langle\ldots\rangle, \texttt{called}, \texttt{his}, \texttt{dog}, \texttt{.}] \qquad \boldsymbol{y} = ?$$

Figure 5: Example of k-Shot Language Modelling

- **With fine-tuning**. Each episode within the meta-test set has its own support set S_j and test points, which are then used to fine-tune the classifier before computing the accuracy on the test labels.
- **Without fine-tuning**. Each episode within the meta-test set only has a set of points to predict and the classifier gets tested as-is.

Our work will compare both approaches (with and without fine-tuning) on the baseline and the different improvements upon it. A diagram outlining this approach can be seen in figure 6. In general, this follows the principle (also mentioned in the paper) that test and training conditions must match.

3.4 Building pairs

In order to benchmark each iteration of the model, we will need to build a set of sentence/missing word pairs.

With that goal in mind, we will first segment our dataset into sentences. Then, for each episode, we will sample N words and k sentences for each of them without replacement. The inputs $\boldsymbol{x}_i$ of the support set will then be the result of removing the sampled word from each of the sampled sentences, and the labels $\boldsymbol{y}_i$ will be each of the sampled words. The process can be seen outlined in Figure 7.

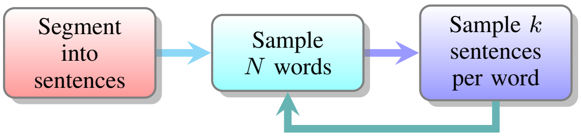


Figure 7: Schema to sample sentence/missing word pairs. The N words must be sampled without replacement on each episode.

Following this schema we will make sure that the labels and sentences are always balanced and that the different inputs never overlap between sets.

3.5 Outline of project

The first phase of the project will consist on re-implementing matching networks using `pyTorch` and `torchtext` (Paszke et al., 2017), to then replicate the results obtained by the paper on language modelling.

As part of the work, this will also require us to pre-process the dataset using `spaCy` (Honnibal and Montani, 2017). The modifications required are outlined in Section 4. This will allow us to establish a common ground to help us iterate upon with further improvements.

Afterwards, we will implement on top of it the two outlined improvements, and test them both separately and together. We will then report and analyse the different performance changes and discuss how much effect each of the changes had.

All code will be published as one of the outputs of the project.

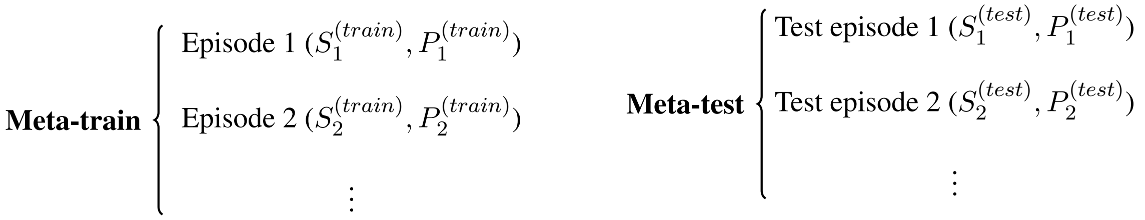
4 Feasibility study

All methods described in Section 2 offer a mechanism for effective training in a low-data regime. However, from those, only matching networks sets a valid baseline for Language Modelling. Therefore, it seems safe to think that using ideas inspired by further developments in one-shot learning should allow us to improve on this baseline.

In this section we will discuss the dataset we will use for the task described in Section 3.

4.1 Dataset

The dataset used in the original matching networks paper was the Penn Treebank (PTB) set (Marcus et al., 1993b). This dataset, built from Wall Street Journal articles, is composed of over 1 million words overall. However, it is currently distributed under a license fee.


Meta-train
Episode 1 $(S_1^{(train)}, P_1^{(train)})$
Episode 2 $(S_2^{(train)}, P_2^{(train)})$
⋮
Meta-test
Test episode 1 $(S_1^{(test)}, P_1^{(test)})$
Test episode 2 $(S_2^{(test)}, P_2^{(test)})$
⋮

(a) Meta-train episodes. The classifier is trained on each episode. Labels don't overlap between episodes.

(b) Meta-test episodes. Fine-tuning on $S_j^{(test)}$ becomes optional. Labels are different than on the meta-training episodes.

Figure 6: Meta-training schema.

To overcome the licensing issues we will use, instead, the WikiText-2 dataset (Merity et al., 2016) which is available under the Creative Commons Attribution-ShareAlike License. This dataset was assembled from articles in Wikipedia and it has over 2 million words overall. We believe this is a sensible course of action because there should be a publicly available NLP equivalent of the Omniglot data set for CV tasks.

As opposed to PTB, WikiText-2 maintains the original punctuation, case and numbers. Therefore, to remain as close as possible to the conditions of the original paper, we will first pre-process the dataset by:

- Lowercasing all input.
- Replacing all numbers by the special token N.
- Removing punctuation.

The dataset is already pre-partitioned into training set, validation set and test set containing (before pre-processing) over $2,000,000$, $217,000$ and $245,000$ words respectively and its vocabulary reaches $33,278$ unique words. Out of vocabulary rate has been measured by division of all unknown tokens by a total number of words in the dataset and it is relatively small (2.6%) compared to the Penn Treebank dataset (4.6%). These results are summarised in Table 3.

	Penn Treebank	**WikiText-2**
Tokens	$1,0365,080$	$2,551,843$
Vocabulary	$10,000$	$33,278$
OoV	4.8%	2.6%

Table 3: Summary of PTB and WT2

Therefore, WikiText-2 should give us enough power to build a similar challenge to the task matching networks was originally trained on and it should allow us to sample many different occurrences of many different labels, which makes it akin to an *"extreme classification"* setup.

More specifically, the original paper randomly sampled 9000 and 1000 labels for its training and test sets respectively, for a variable number of k example sentences up to 5. We therefore need to consider if the WikiText-2 dataset also allows such sentence/missing word pair sampling frequency. To do so, we have first segmented the whole dataset into sentences, to then count in how many different sentences each unique token in the vocabulary appears.

On the resulting plot in Figure 8 we can see how $11,658$ unique tokens appear in up to 7 sentences, and more than $6,662$ appear in between 7 to 55 different sentences. Therefore, it seems safe to think that the dataset will offer enough sampling frequency so as to generate a similar number of sentence/missing word pairs as the original paper.

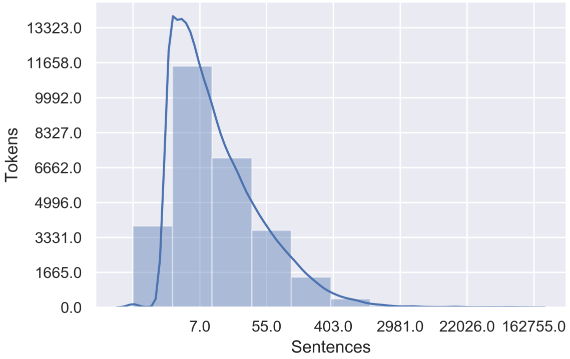


Figure 8: Histogram showing how common it is that a token appears in k different sentences.

Acknowledgments

We would like to thank Oriol Vinyals for his helpful discussions on his paper, "Matching Networks for One Shot Learning" and giving us insights and suggestions. We also thank Sebastian Ruder for giving us helpful directions on one-shot learning, and Patrick Lewis for feedback on our project. This project idea was developed from reading (Ruder, 2018). We also benefited from writings of Lilian Weng on one-shot learning (Weng, 2018).

References

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. 1992. On the optimization of a synaptic learning rule. In *Conference on Optimality in Biological and Artificial Networks*, Dallas, USA.

I. Biederman. 1987. Recognition-by-components: a theory of human understanding. In *Psychological Review*, pages 94:115–147.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a "siamese" time delay neural network. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 737–744. Morgan-Kaufmann.

Michael C. Burl, Markus Weber, and Pietro Perona. 1998. A probabilistic approach to object recognition using local photometry and global geometry. In *Proceedings of the 5th European Conference on Computer Vision-Volume II - Volume II*, ECCV '98, pages 628–641, London, UK, UK. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia. PMLR.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition.

Andrew Kyle Lampinen and James L McClelland. 2017. One-shot and few-shot learning of word embeddings. In *CoRR*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993a. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993b. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 1(2).

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. *arXiv:1609.07843 [cs]*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2554–2563, International Convention Centre, Sydney, Australia. PMLR.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999.

Maximilian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. *NIPS*, pages 6341—-6350.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *ICLR*.

Sebastian Ruder. 2018. A review of the neural history of natural language processing. Last accessed: 17/02/2019.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2019. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA. PMLR.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Sander Dieleman Marc Lanctot, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, , and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. In *Nature*.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*, pages 4080–4090.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Sebastian Thrun and Lorien Pratt, editors. 1998. *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc.

Lilian Weng. 2018. Meta-learning: Learning to learn fast. Last accessed: 17/02/2019.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. Cite arxiv:1409.2329.