Trajectory-Based Meta-Learning for Out-Of-Vocabulary Word Embedding Learning

Gordon Buck
University of Cambridge
ghb28@cantab.ac.uk

Andreas Vlachos
University of Cambridge
av308@cam.ac.uk

Abstract

Word embedding learning methods require a large number of occurrences of a word to accurately learn its embedding. However, out-of-vocabulary (OOV) words which do not appear in the training corpus emerge frequently in the smaller downstream data. Recent work formulated OOV embedding learning as a few-shot regression problem and demonstrated that meta-learning can improve results obtained. However, the algorithm used, model-agnostic meta-learning (MAML) is known to be unstable and perform worse when a large number of gradient steps are used for parameter updates. In this work, we propose the use of Leap, a meta-learning algorithm which leverages the entire trajectory of the learning process instead of just the beginning and the end points, and thus ameliorates these two issues. In our experiments on a benchmark OOV embedding learning dataset and in an extrinsic evaluation, Leap performs comparably or better than MAML. We go on to examine which contexts are most beneficial to learn an OOV embedding from, and propose that the choice of contexts may matter more than the meta-learning employed.

1 Introduction

Distributional methods for learning word embeddings require a sufficient number of occurrences of a word in the training corpus to accurately learn its embedding. Even though the embeddings can be trained on raw text implying that an embedding for every word is obtained, in practice out-of-vocabulary (OOV) words do occur in the downstream applications embeddings are used, for example due to domain-specific terminology. Nevertheless, OOV words are often content words such as names which convey important information for downstream tasks; for example, drug names are key in the biomedical domain. However, the amount of downstream language data is typically much smaller than the corpus used for training word embeddings, thus methods that rely on distributional properties of words across large amounts of data perform poorly (Herbelot and Baroni, 2017).

Researchers often assign OOV words to random embeddings or to an "unknown" embedding, however these solutions fail to capture the distributional properties of words. Zero-shot approaches (Pinter et al., 2017; Kim et al., 2016; Bojanowski et al., 2017) attempt to predict the embeddings for OOV words from their characters alone. These approaches rely on inferring the meaning of a word from its subword information, such as morphemes or WordPiece tokens used in BERT (Devlin et al., 2019). While this works well for many words, it performs poorly for names and words where morphology is not informative.

Given that an OOV word occurs once, the chance of a second occurrence is much higher than the first (Church, 2000). Hence while OOV words can be rare and not seen in training, it is reasonable to expect that a limited number of occurrences will be present in the data of a downstream application. Few-shot approaches (Garneau et al., 2018; Khodak et al., 2018; Hu et al., 2019) leveraged this to predict the embeddings for OOV words from just a few contexts, often in conjunction with their morphological information. Hu et al. (2019) proposed an attention-based architecture for OOV word embedding learning as a few-shot regression problem. The model is trained to predict the embedding of a word based on a few contexts and its character sequence. Such a model is trained by simulating OOV words in the training corpus, with their target embeddings provided by learning them on the same corpus. As OOV words must have their embeddings inferred from contexts outside the training corpus, the authors show that using an adaptation of the model-agnostic meta-learning (MAML) algorithm (Finn et al., 2017) to adapt the model's

Proceedings of the Second Workshop on Domain Adaptation for NLP, pages 146–155
April 20, 2021.

parameters to the target domain improves the quality of the learned OOV word embeddings.

However, MAML is known to be unstable due to the calculation of gradients requiring backpropagation through multiple instances of the model, as the learning process must be unrolled to calculate gradients with respect to the initial parameters (Antoniou et al., 2019). In practice, the learning process is often truncated to a small number of gradient steps, but has been shown to have a short-horizon bias (Wu et al., 2018), causing it to underperform.

In this work we explore OOV word embedding learning using *Leap* (Flennerhag et al., 2019), a meta-learning framework which takes into consideration the entire learning trajectory, not only the beginning and end points. Each task is associated with a loss surface over the model's parameters on which the learning process travels, and the aim is to minimize the expected length of this process across tasks. Leap also does not require backpropagation through the learning process, allowing it to adapt over a larger number of gradient steps and thus not suffering from the short-horizon bias that MAML is prone to.

We conduct an intrinsic evaluation of MAML and Leap on the dataset of Lazaridou et al. (2017) which simulates OOV words by combining the contexts of two semantically similar words to form a 'chimera'. We find that Leap performs better than MAML at adapting model parameters to a new corpus. We also conduct an extrinsic evaluation on NER in the biomedical domain where the results are comparable to MAML, without improving in most cases on a random embedding baseline. Finally, we examine which contexts are more beneficial to learn an embedding from, and note that the contexts from which an embedding is learned matters more than the meta-learning method employed.

2 Meta-Learning

Meta-learning algorithms aim to capture knowledge across a variety of learning tasks such that fine-tuning a model on a specific task both avoids overfitting and leads to good performance. Approaches include learning a similarity function by which to cluster and classify data points (Vinyals et al., 2016; Snell et al., 2017); learning an update rule for neural network optimization (Ravi and Larochelle, 2017); and learning an initialization from which to fine-tune a model. We consider algorithms for the latter.

Formally, we consider task $\mathcal{T}$ to consist of a dataset $D_\mathcal{T}$ of labelled examples (x, y) and loss function $\mathcal{L}_\mathcal{T}(\theta) \rightarrow \mathbb{R}$ which maps a model's parameters θ to a real-valued loss. For batch gradient descent this loss is constant for a given set of parameters, however for stochastic gradient descent it depends on the sampled examples. During training, tasks $\mathcal{T} \sim p(\mathcal{T})$ are sampled from a distribution $p(\mathcal{T})$ over the tasks the model should be able to adapt to. The aim is then to train the model's parameters such that they capture features common to all tasks in $p(\mathcal{T})$, and thus are a promising initialization for any of these tasks. Below we describe the approaches taken by MAML and Leap.

2.1 MAML

During training with MAML, for each task $\mathcal{T}$ the model's parameters are updated from an initialization θ to $\theta^K_\mathcal{T}$ through K gradient steps according to $\mathcal{L}_\mathcal{T}$. The final parameters $\theta^K_\mathcal{T}$ are then used to calculate the final task losses, and the original parameters θ are updated to minimize their sum. This meta-objective is given below in equation 1, where $u_\mathcal{T}(\theta) = \theta - \alpha\nabla_\theta\mathcal{L}_\mathcal{T}(\theta)$ is a single gradient step:

$$\min_\theta \sum_{\mathcal{T}\sim p(\mathcal{T})} \mathcal{L}_\mathcal{T}(\theta^K_\mathcal{T}) = \sum_{\mathcal{T}\sim p(\mathcal{T})} \mathcal{L}_\mathcal{T}(u^K_\mathcal{T}(\theta)) \quad (1)$$

The final task losses $\mathcal{L}_\mathcal{T}(\theta^K_\mathcal{T})$ are usually computed using examples held out from training $\theta^K_\mathcal{T}$ to simulate a testing loss. The meta-optimization is then performed by backpropagating with respect to the original parameters, θ, rather than the trained parameters $\theta^K_\mathcal{T}$. This aims to optimize θ such that a small number of gradient steps, K, on a particular task produces a low testing loss. Algorithm 1 below gives the overview of the training.

Algorithm 1: MAML

Input: $p(\mathcal{T})$: a distribution over tasks
Input: α, β: step size parameters
define $u_\mathcal{T}(\theta) = \theta - \alpha\nabla_\theta\mathcal{L}_\mathcal{T}(\theta)$;
initialize model parameters θ;
while *not done* **do**
  sample batch $\mathcal{B}$ of tasks $\mathcal{T} \sim p(\mathcal{T})$;
  forall $\mathcal{T} \in \mathcal{B}$ **do**
    $\theta^K_\mathcal{T} \leftarrow u^K_\mathcal{T}(\theta)$;
  end
  $\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}\in\mathcal{B}} \mathcal{L}_\mathcal{T}(\theta^K_\mathcal{T})$;
end

Algorithm 2: Leap

Input: $p(\mathcal{T})$: a distribution over tasks
Input: α, β: step size parameters
initialize model parameters θ;
while *not done* **do**
  $\nabla \bar{F} \leftarrow 0$;
  sample batch $\mathcal{B}$ of tasks $\mathcal{T} \sim p(\mathcal{T})$;
  forall $\mathcal{T}$ **do**
    $\theta_{\mathcal{T}}^0 \leftarrow \theta$;
    $\psi_{\mathcal{T}}^0 \leftarrow \theta$;
    forall $i \in \{0, ..., K-1\}$ **do**
      $\theta_{\mathcal{T}}^{i+1} \leftarrow \theta_{\mathcal{T}}^i - \alpha \nabla_{\theta_{\mathcal{T}}^i} \mathcal{L}_{\mathcal{T}}(\theta_{\mathcal{T}}^i)$;
      $\psi_{\mathcal{T}}^{i+1} \leftarrow \psi_{\mathcal{T}}^i - \alpha \nabla_{\psi_{\mathcal{T}}^i} \mathcal{L}_{\mathcal{T}}(\psi_{\mathcal{T}}^i)$;
      $\nabla \bar{F} \leftarrow \nabla \bar{F} + \frac{(\mathcal{L}_{\mathcal{T}}(\theta_{\mathcal{T}}^i) - \mathcal{L}_{\mathcal{T}}(\psi_{\mathcal{T}}^{i+1})) \nabla_{\theta_{\mathcal{T}}^i} \mathcal{L}_{\mathcal{T}}(\theta_{\mathcal{T}}^i) + (\theta_{\mathcal{T}}^i - \psi_{\mathcal{T}}^{i+1})}{\|\bar{\gamma}_{\mathcal{T}}^{i+1} - \gamma_{\mathcal{T}}^i\|_2}$;
    end
  end
  $\theta \leftarrow \theta - \frac{\beta}{|\mathcal{B}|} \nabla \bar{F}$;
end

Computing $\nabla_\theta \sum_{\mathcal{T} \in \mathcal{B}} \mathcal{L}_{\mathcal{T}}(\theta_{\mathcal{T}}^K)$ requires backpropagation through the learning process for each task $\mathcal{T} \in \mathcal{B}$. Gradients are computed by backpropagation through $K+1$ different instances of the model's parameters, which can cause both exploding and diminishing gradient problems, and becomes unstable for large K (Antoniou et al., 2019). However, truncating the learning process with a small K results in a short-horizon bias (Wu et al., 2018), where the learned parameters adapt poorly to tasks over a number of gradient steps larger than K. To extend to more steps, a first order approximation of MAML has been shown to achieve similar performance (Nichol et al., 2018; Finn et al., 2017). However, this still considers only the initial and the final parameters and loss, and for larger K the intermediate steps become more significant.

2.2 Leap

In Leap the learning process is viewed as a path along a loss surface $\mathcal{L}$, traversed by K gradient steps from initial to final parameters. The intuition behind Leap is that geometrical similarities between learning processes associated with different tasks can be exploited for transfer learning. In particular, Leap seeks to find an initialization that reduces the expected length of learning processes.

Following Flennerhag et al. (2019), we consider the learning process to be a sequence of discrete points $\{\gamma^i\}_{i=0}^K$ with $\gamma^i = (\theta^i, \mathcal{L}(\theta^i))$ corresponding to K gradient updates, and as such we consider the learning process to be the shortest path passing through these points. The length of a learning process is then approximated as the cumulative chordal distance of the path from initial parameters $\theta = \theta^0$ to final parameters θ^K.The cumulative chordal distance approximates the length of the arc passing through the points $\{\gamma^i\}_{i=0}^K$, and is key to minimizing the length of the learning process rather than simply moving the initial parameters towards the final parameters:

$$d(\theta; \mathcal{L}) = \sum_{i=0}^{K-1} \|\gamma^{i+1} - \gamma^i\|_2 \tag{2}$$

Considering again a distribution of tasks $p(\mathcal{T})$, an initialization θ is associated with an expected learning process length $\mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})}[d(\theta; \mathcal{L}_{\mathcal{T}})]$. When dealing with complex non-convex loss surfaces, minimizing only the expected learning process length makes no guarantees about the final loss $\mathcal{L}(\theta^K)$ and thus may inadvertently find final parameters θ^K with lower performance. Note that MAML takes this into account directly in its objective by optimizing for the final loss on a separate held-out set. Leap instead enforces this as part of its meta-objective by requiring that the optimized initialization θ must not converge to a higher loss than a baseline initialization $\psi = \psi^0$ for all tasks

$\mathcal{T}$. That is, $\mathcal{L}_{\mathcal{T}}(\theta^K) \leq \mathcal{L}_{\mathcal{T}}(\psi^K)$ assuming convergence after K gradient steps. For this purpose, Leap defines an objective which optimizes the initialization for expected learning process length only along the task-specific learning processes originating from a baseline initialization. This ensures that the learning processes originating from the optimized initialization will have no greater final loss than their counterpart originating from the baseline initialization, given that both consist of K gradient steps. The objective is given in equation 3, where points $\{\gamma^i_{\mathcal{T}}\}^K_{i=0}$ lie along the learning process for task $\mathcal{T}$ originating from the optimized initialization θ, while points $\{\bar{\gamma}^i_{\mathcal{T}}\}^K_{i=0}$ lie along the respective learning process originating from the fixed baseline initialization ψ.

$$\begin{aligned} \bar{d}(\theta; \mathcal{L}_{\mathcal{T}}, \psi) &= \sum_{i=0}^{K-1} \|\bar{\gamma}^{i+1}_{\mathcal{T}} - \gamma^i_{\mathcal{T}}\|_2 \\ \min_{\theta} \bar{F}(\theta; \psi) &= \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})}[\bar{d}(\theta; \mathcal{L}_{\mathcal{T}}, \psi)] \end{aligned} \tag{3}$$

Gradient descent on the objective $\bar{F}$ with respect to θ pulls the parameters θ^i towards ψ^{i+1}. Parameters θ are initialized to be equal to ψ such that each gradient descent update pulls θ forward along the learning processes originating from it.

Algorithm 2 describes the training with Leap. Tasks $\mathcal{T} \sim p(\mathcal{T})$ are sampled from the distribution $p(\mathcal{T})$ and the model's parameters are updated to $\theta^K_{\mathcal{T}}$ through K gradient steps for each task $\mathcal{T}$ as in MAML (lines 1-10; the learning trajectory is expanded in lines 8, 9 and 10, as it is needed in Leap). The gradient $\nabla \bar{F}$ is incrementally computed at each point $(\theta^i_{\mathcal{T}}, \mathcal{L}_{\mathcal{T}}(\theta^i_{\mathcal{T}}))$ during the task training (line 11). $\nabla \bar{F}$ is always evaluated at $\theta = \psi$, with the update term on line 11 pulling θ^i towards $\psi^{i+1} = \theta^{i+1}$. This is performed in order to take into account each point in the learning trajectory, as opposed to the start and end points in MAML. The initialization θ is then updated according to the accumulated gradient $\nabla \bar{F}$ (line 14), and the algorithm continues with ψ set to the updated θ. This is done implicitly when θ is updated, which ensures that any future θ improves the task loss over the one already obtained, instead of just over the initial random initialization.

3 Leap for OOV Embedding Learning

In this section we describe our application of Leap to learning embeddings for OOV words. This technique is generally applicable to any word embedding regression function H_θ trainable by gradient descent, though in our experiments we use the HiCE architecture (Hu et al., 2019).

The regression function H_θ is trained to predict a word's embedding given K contexts, and possibly morphological information, with words and their contexts sampled from some large training corpus $\mathcal{D}_T$. Each word (either a training target or in a context) is represented as a pre-trained embedding in the input to H_θ. We note that H_θ can be trained only on words with sufficient occurrences in $\mathcal{D}_T$ such that their pre-trained embeddings can be accurately learned. The trained H_θ can then be used to infer the embeddings for OOV words, for which we do not have a pre-trained embedding. However, OOV words often form part of domain-specific vocabularies, the semantics of which are not captured by word embeddings trained on generic large corpora (Kameswara Sarma et al., 2018). To counteract this, we adapt the trained parameters θ_T of the word embedding regression function to the domain in which we infer the OOV embeddings.

When adapting θ_T we consider both $\mathcal{D}_T$, and the corpus on which we wish to infer OOV embeddings, $\mathcal{D}_N$. We wish to transfer the knowledge encoded in θ_T to the task of predicting OOV embeddings for words in $\mathcal{D}_N$. One approach would be to simply fine-tune θ_T on $\mathcal{D}_N$; that is, sample words and their contexts from $\mathcal{D}_N$ which have their pre-trained embeddings from $\mathcal{D}_T$ known and train H_θ as before on these words. This ignores that $\mathcal{D}_N$ is much smaller than a corpus usually used for word embedding training, and direct training on it is likely to lead to overfitting to the corpus rather than adapting to its domain, which in turn hurts the quality of inferred OOV embeddings, an instance of catastrophic forgetting (French, 1999). Instead of fine-tuning, meta-learning algorithms can be applied. Hu et al. (2019) use MAML, and we extend this work with the use of Leap.

We adapt θ_T by applying Leap across two tasks; inferring OOV words in $\mathcal{D}_T$ and $\mathcal{D}_N$. Optimizing the objective in equation 4 moves the adapted parameters θ to minimize the lengths of the two corresponding learning processes. The loss function $\mathcal{L}(\cdot|\mathcal{D})$ used throughout is the cosine distance between predicted and pre-trained embeddings.

$$\min_{\theta} \sum_{C \in \{T,N\}} \bar{d}(\theta; \mathcal{L}(\cdot|\mathcal{D}_C), \theta_T) \tag{4}$$

This pulls the model parameters θ along the learning processes for $\mathcal{D}_T$ and $\mathcal{D}_N$ originating from the

pre-trained parameters θ_T. The learning process for $\mathcal{D}_T$ should already be short, as θ_T is trained to convergence on $\mathcal{D}_T$. Optimization of the pull-forward objective must naturally pull the parameters away from this point of convergence to minimize the length of the learning process for $\mathcal{D}_N$. However, as each task is weighted equally, the model parameters cannot move towards the convergence point for $\mathcal{D}_N$ if this results in a larger divergence from the convergence point for $\mathcal{D}_T$. It is thus necessary for the parameters to move into areas which encode knowledge sharing between $\mathcal{D}_T$ and $\mathcal{D}_N$. This reduces overfitting on $\mathcal{D}_N$ by ensuring θ does not move too far from the convergence point for $\mathcal{D}_T$.

While each word's embedding is inferred from only a few contexts, the total number of words available for training in $\mathcal{D}_N$ is large enough so θ can be adapted over a larger number of gradient steps. We posit that this, in conjunction with knowledge sharing that considers the entire learning trajectory rather than just the beginning and end points, results in higher quality OOV embeddings than those that can be obtained with MAML.

4 Experiments

4.1 Intrinsic Evaluation

To obtain an intrinsic evaluation of the methods proposed, we require a dataset that simulates the natural occurrences of OOV words in a real-world setting, and defines a notion for evaluating how close an embedding is to representing an OOV word's meaning. Following Hu et al. (2019), we use the 'Chimera' dataset for evaluation, a popular benchmark dataset for OOV words.

The Chimera dataset (Lazaridou et al., 2017) is constructed specifically to simulate unseen words occurring naturally in text. Each unseen word is a *chimera*, which is a novel concept created by combining two related but distinct concepts; for example a gorilla and a bear. In total there are 33 chimeras, generated by first taking a base concept, called a *pivot*, and matching this with a compatible concept by traversing a list of terms ranked by similarity to the pivot. In the case of the 'gorilla/bear' chimera, the pivot is the gorilla and the compatible term is the bear. Each chimera is then associated with passages of 2, 4 and 6 sentences, with half containing the pivot and half containing the compatible concept. The occurrences of the pivot and the compatible term are replaced with a nonce word that represents the chimera; for example 'mohalk'.

Each passage is then annotated by human subjects with similarity scores between the nonce word and six different words, called *probes*, specific to the chimera. These similarity scores are then averaged across human subjects, resulting in six scores for each passage indicating the similarity between the chimera and each probe word.

For each of the 2-shot, 4-shot and 6-shot cases the chimera's embedding is inferred given each sentence (shot) as a context and the pivot's character sequence. Following Lazaridou et al. (2017) we measure the performance of the embeddings inferred by looking at their cosine similarity to the probe embeddings and calculating the Spearman correlation to the similarity judgements by the human subjects.

Throughout all experiments HiCE is trained on WikiText-103 (Merity et al., 2017) with pre-trained embeddings provided by SkipGram (Mikolov et al., 2013) on the same corpus. Where MAML and Leap are used, word embeddings are adapted to the Chimera dataset. Both approaches were implemented using PyTorch (Paszke et al., 2017) and the code will become publicly available. To implement MAML we use `higher` (Grefenstette et al., 2020), a PyTorch library for higher order optimization such as backpropagation through gradient descent updates, as is required for MAML. This allows us to compute the second order gradients that are required for MAML, rather than using a first order approximation. The learning rates α and β for each of MAML and Leap were chosen based on each algorithm's stability during training; for MAML we used $\alpha = 5 \times 10^{-4}, \beta = 1 \times 10^{-5}$ and for Leap we used $\alpha = 5 \times 10^{-4}, \beta = 1 \times 10^{-4}$. These learning rates are in line with the publicly available code of Hu et al. (2019). The number of gradient steps used for adaptation with MAML was 4, while with Leap we increased this to 64, taking advantage of its ability to train tasks over longer horizons.[1]

Table 1 gives the average Spearman correlations for HiCE, its combination with MAML as proposed by Hu et al. (2019), and its combination with Leap as proposed in this work, for the 2-shot, 4-shot and 6-shot cases. We also include the results for related works taken from their corresponding papers. These include fasttext (Bojanowski et al., 2017); the additive method (a simple averaging of context word embeddings) (Lazaridou et al., 2017); a la

[1]The code used in our experiments is available here: http://github.com/Gordonbuck/ml-oov-we

	2-shot	4-shot	6-shot
fasttext	0.1775	0.1738	0.1294
additive	0.3376	0.3624	0.4080
a la carte	0.3634	0.3844	0.3941
nonce2vec	0.3320	0.3668	0.3890
HiCE	0.3611 $\pm$ 0.0054	0.3882 $\pm$ 0.0049	0.4134 $\pm$ 0.0043
+MAML	0.3574 $\pm$ 0.0058	0.3901 $\pm$ 0.0072	0.4169 $\pm$ 0.0057
+Leap	**0.3695 $\pm$ 0.0022**	**0.4022 $\pm$ 0.0043**	**0.4262 $\pm$ 0.0051**
pre-trained	0.4173	0.4367	0.4410

Table 1: Average Spearman's correlations for each k-shot case and method. Resulting of HiCE are given with 95% confidence intervals. Bold indicates the best results.

carte (a linear transformation-based modification of the averaging method) (Khodak et al., 2018); and nonce2vec (a modification of the Word2Vec algorithm for few-shot learning) (Herbelot and Baroni, 2017). We also give the scores obtained when using the pivot's pre-trained embedding as the chimera's embedding to indicate a ceiling, following Hu et al. (2019). However, we would expect the OOV embedding to differ from the pivot's pre-trained embedding, since the semantics of a chimera are a combination of the pivot and compatible concept.

HiCE+Leap achieves the best results across all k-shot settings in our experiments. The results for HiCE, HiCE+MAML and HiCE+Leap are all obtained by averaging the results over 10 different random seeds, and we give a 95% confidence interval for each. We take this approach to highlight the known instability of training MAML across random seeds (Antoniou et al., 2019), even with no hyperparameter changes. Leap consistently performs better than MAML and with a lower variance; in all cases, the average Spearman's correlation for MAML lies outside of the confidence interval range given for Leap. We also see that due to MAML's instability it can actually lower the performance of pre-trained HiCE in the 2-shot case. Outside of HiCE-based methods, a la carte performs best in the 2- and 4-shot cases, and additive performs best in the 6-shot case. These scores similarly lie outside of confidence interval ranges for HiCE+Leap, which overall performs best in each case.

Hu et al. (2019) reported 0.3781, 0.4053 and 0.4307 with HiCE+MAML in the in the 2-, 4- and 6-shot cases respectively, but did not report experiments with multiple random seeds. While were able to obtain similar results for HiCE+MAML in some of our experiments, they were outside the confidence intervals we obtained, illustrating the relative instability in training with MAML (Antoniou et al., 2019). The results of Hu et al. (2019) are also lower than the highest results we obtained with HiCE+Leap (0.3896, 0.4116 and 0.4395 for 2-, 4- and 6-shot).

4.2 Extrinsic Evaluation

To gauge the quality of the OOV embeddings for downstream tasks we evaluate their performance when applied to NER. For this purpose we use the JNLPBA 2004 Bio-Entity Recognition Task dataset (Collier and Kim, 2004). We choose this dataset as the biomedical domain differs significantly from the domain of Wikipedia that HiCE is pre-trained on, and contains many OOV technical terms. Hu et al. (2019) use this dataset also but did not provide their datasplits; thus. while we were able to confirm their results, we cannot compare against them directly.

The JNLPBA dataset is constructed from 2000 abstracts for training and 404 abstracts for testing, each extracted from a bibliographic database of biomedical information and hand annotated with 36 classes corresponding to chemical classifications. These classifications are simplified into 5 classes for the purpose of the bio-entity recognition task; *protein*, *DNA*, *RNA*, *cell-line* and *cell-type*. In total there are 18546 training and 3856 test sentences.

Following Hu et al. (2019), HiCE is trained on the WikiText-103 corpus, and we adapt word embeddings to the biomedical domain by using the JNLPBA dataset as a corpus. Contrary to the Chimera dataset, we consider contexts at the abstract level rather than only the sentence level. We train different embeddings for OOV words for each of the 2-shot, 6-shot and 10-shot cases, considering only those OOV words with 2, 6 or 10 occurrences or more respectively. In total we infer embeddings

	2-shot	6-shot	10-shot
random	**0.7226**	0.7206	0.7209
HiCE	0.7116	0.7213	0.7232
+MAML	0.7135	0.7232	0.7269
+Leap	0.7141	**0.7256**	**0.7282** †

Table 2: Micro-averaged F1 score for each of the 2-shot, 6-shot and 10-shot settings. Bold indicates the best results and (†) indicates the result is better than random embeddings at a 0.05 significance level.

for 4643 OOV distinct words (types) in the 2-shot case, 1310 in the 6-shot case and 702 in the 10-shot case.[2] The contexts used to infer the embedding of a word are chosen at random from the contexts that word appears. The inferred embeddings, alongside the embeddings for in-vocabulary words, are used as input to train the LSTM-CRF architecture of Lample et al. (2016).

For each of the k-shot cases a separate test set is created by subsampling, such that the respective test set contains only those sentences with an OOV word whose embeddings has been inferred. This ensures that the test sets focus on the quality of the inferred OOV embeddings. For the 2-shot case there are 2876 test sentences; 2451 for 6-shot; and 2134 for 10-shot. The results for each k-shot setting are given in Table 2, reported in micro-averaged F1 score. The results obtained using random OOV embeddings as input are given as a baseline.

Results are marginally improved with any of the proposed methods for the 6-shot and 10-shot cases, with HiCE+Leap producing the best results. We perform a paired t-test on each pair of results within a k-shot case. However, we do not find the differences between methods to be significant, with only HiCE+Leap in the 10-shot case performing significantly better than random embeddings, at a 0.05 significance level.

We find that performance increases across all methods as we use more contexts. However, for the 2-shot case we find that results are lower than if random embeddings are used. With fewer contexts the quality of the learned OOV embeddings is naturally lower, and inaccuracies in the embeddings add noise which hurts the performance of the downstream NER tagger. This highlights the need for a sufficient number of occurrences to effectively learn word embeddings, even with models specifically designed to handle the lack of data, and that using inaccurate embeddings can lower the performance of the entire downstream system. Our findings corroborate contemporary research which suggests that random embeddings can perform comparably to pre-trained and contextual embeddings on benchmark tasks (Arora et al., 2020).

[2]Hu et al. (2019) do not distinguish between different number of shots/contexts per word in their results.

4.3 Informative Contexts

Apart from comparing different meta-learning approaches for word embedding learning, we seek to find which contexts are invariably most informative to a word's meaning. For this purpose, we return to the Chimera dataset. Considering the 2-shot case, we rank passages by the performance of the chimera embeddings inferred from them. That is, we infer the chimera embeddings, obtain the cosine similarities against the probe embeddings, and calculate the Spearman correlation against the human scores. We then calculate the pairwise Spearman correlation between these rankings across random seeds and methods experimented with in this paper, i.e. HiCE, HiCE with MAML, and HiCE with Leap. The average Spearman correlation is 0.89 ± 0.0047 for a 95% confidence interval, indicating that the rankings of passages are largely similar across methods. Thus we conclude that the contexts which each method finds most useful to infer a chimera's meaning are largely invariable.

Ordering passages by their cumulative rank across methods, we observe that passages which consistently perform poorly are composed of sentences with more ambiguity and fewer content words. The top section in table 3 gives the lowest and highest performing passages as examples. The lowest performing passage contains little in the way of content words indicating the meaning of the chimera 'refrigerator/closet' besides the presence of 'freezers', and the second sentence is highly ambiguous. Naturally, human annotators would also struggle to pinpoint the semantic meaning of this chimera based on the two sentences given. In contrast the highest performing passage very clearly relates to organic produce and cooking, providing far more hints as to the semantic meaning of the chimera 'broccoli/spinach'.

We also look at which passages perform best for a given chimera. The bottom section in table 3 gives the lowest and highest performing passages for the 'drum/tuba' chimera. We observe the same trend within the chimera, with the lowest

chimera	probes	shot 1	shot 2	score
refrigerator / closet	cupboard, basement, mixer, dishwasher, ladle, boat	23144 security materials that may adversely affect hpc components shall not be stored in the ___ or freezers	if there is any difference from the first there are fewer of those ___	-0.68
broccoli / spinach	celery, radish, grape, salamander, budgie, pot	all manner of produce fill the fields including exotic vegetables like pumpkins spring onions ___ and asparagus	discard any bruised or yellow leaves and soak remaining leaves in a basin of cold water with ___	0.98
drum / tuba	bagpipe, harmonica, whistle, shotgun, bear, bouquet	i could nt yet hear the ___ that told you where to go on any particular day	is probably at mirage poker room income from week he borrowed was swimming ___	-0.63
drum / tuba	bagpipe, harmonica, whistle, shotgun, bear, bouquet	is there a better way to start a record or a song than with a thumpin ___ intro	they add considerably to the tone of the ___ however when used on their low notes	0.91

Table 3: Each section gives a higher and lower performing 2-shot passage with its average Spearman correlation across all methods and random seeds. The nonce word is replaced by '___'.

performing passage consisting of more ambiguous sentences, while the highest performing passage contains many content words which hint at the semantics of the chimera, such as 'song'. These two passages differ greatly in their performance, with the lowest passage averaging a Spearman correlation of -0.63 while the highest passage achieves an average of 0.91. However, if we look across methods the difference in performance for each of these passages is no higher than 0.2. This suggests that one of the most important factors to inferring an OOV word's embedding is the choice of contexts, perhaps more so than the meta-learning employed. To quantify this further, we calculate the Spearman correlation between the proportion of content words in a passage and its score, which we find to be 0.12 ± 0.0083 for a 95% confidence interval. We do this by using a standard part-of-speech tagger (Honnibal and Montani, 2017) on each passage; taking all nouns, adjectives, verbs and adverbs to be content words. While the correlation is weak, it is significant at a 95% confidence interval and further suggests that context informativeness is a suitable future area of work.

5 Conclusion

We investigated the use of meta-learning for few-shot learning of OOV word embeddings. We built on the work of Hu et al. (2019), formulating OOV word embedding learning as a few-shot regression problem and training their proposed architecture HiCE to predict OOV embeddings given K contexts and morphological features. We proposed the use of Leap as a meta-learning algorithm to adapt HiCE to a new semantic domain and compared it to the popular MAML as used by Hu et al. (2019). Experiments on a benchmark dataset show that Leap is more stable and achieves comparably higher performance than MAML in the context of OOV embedding learning. Further experimentation shows that there is little variation in which contexts perform well across both random seeds and meta-learning approaches, and a qualitative analysis indicates that performance is lower on ambiguous sentences with fewer content words. Our findings suggest a future avenue of work which focuses on the selection of contexts from which to learn an OOV embedding, such as prioritising contexts based on a notion of informativeness.

References

Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2019. How to train your MAML. In *International Conference on Learning Representations*.

Simran Arora, Avner May, Jian Zhang, and Christopher

Ré. 2020. Contextual embeddings: When are they worth it? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2663, Online. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kenneth W. Church. 2000. Empirical estimates of adaptation: The chance of two noriegas is closer to p/2 than p2. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*.

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 73–78, Geneva, Switzerland. COLING.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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. Association for Computational Linguistics.

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*, ICML'17, page 1126–1135. JMLR.org.

Sebastian Flennerhag, Pablo Garcia Moreno, Neil Lawrence, and Andreas Damianou. 2019. Transferring knowledge across learning processes. In *International Conference on Learning Representations*.

Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135.

Nicolas Garneau, Jean-Samuel Leboeuf, and Luc Lamontagne. 2018. Predicting and interpreting embeddings for out of vocabulary words in downstream tasks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 331–333, Brussels, Belgium. Association for Computational Linguistics.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. 2020. Generalized inner loop meta-learning. In *International Conference on Learning Representations*.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 304–309, Copenhagen, Denmark. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Few-shot representation learning for out-of-vocabulary words. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4102–4112, Florence, Italy. Association for Computational Linguistics.

Prathusha Kameswara Sarma, Yingyu Liang, and Bill Sethares. 2018. Domain adapted word embeddings for improved sentiment classification. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 51–59, Melbourne. Association for Computational Linguistics.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Melbourne, Australia. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2741–2749. AAAI Press.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive Science*, 41 Suppl 4.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*

- *Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms.

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.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark. Association for Computational Linguistics.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc.

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.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger B. Grosse. 2018. Understanding short-horizon bias in stochastic meta-optimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.