

---

# Learning to Compute Word Embeddings On the Fly

---

**Dzmitry Bahdanau**

MILA, Université de Montréal  
ElementAI  
bahdanau@iro.umontreal.ca

**Tom Bosc**

MILA, Université de Montréal

**Stanisław Jastrzębski**

Jagiellonian University, Krakow

**Edward Grefenstette**

DeepMind, London

**Pascal Vincent**

MILA, Université de Montréal  
CIFAR

**Yoshua Bengio**

MILA, Université de Montréal  
CIFAR

## Abstract

Words in natural language follow a Zipfian distribution whereby some words are frequent but most are rare. Learning representations for words in the “long tail” of this distribution requires enormous amounts of data. Representations of rare words trained directly on end-tasks are usually poor, requiring us to pre-train embeddings on external data, or treat all rare words as out-of-vocabulary words with a unique representation. We provide a method for predicting embeddings of rare words on the fly from small amounts of auxiliary data with a network trained against the end task. We show that this improves results against baselines where embeddings are trained on the end task in a reading comprehension task, a recognizing textual entailment task, and in language modelling.

## 1 Introduction

Natural Language Understanding is a particularly difficult branch of artificial intelligence research. This is due in part to the fact that being able to comprehend, reason, and generate fluent natural language is the hallmark of human-level intelligence, but there are practical challenges as well. Although we can operate at finer levels of granularity, such as characters or phonemes, the base unit of language is frequently taken to be the word. This medium presents particular difficulties for machine learning because words are sparse features, for which there may be a paucity of data from which to learn good representations. Indeed, Natural Language yields a Zipfian distribution (Zipf, 1949) which tells us that a core set of words (at the head of the distribution) are frequent and ubiquitous, while a significantly larger number (in the long tail) are rare.

The typical solution to deal with this problem is to learn embeddings for some proportion of the head of the distribution, possibly shifted towards the domain-specific vocabulary of the dataset or task at hand, and to treat all other words as out-of-vocabulary (OOV), replacing them with an unknown word “UNK” token with a shared embedding. This essentially heuristic solution is inelegant, as words from technical domains, names of people, places, institutions, and so on will lack a specific representation unless sufficient data are available to justify inclusion in the vocabulary. This forces model designers to rely on overly large vocabularies, as observed by (Mi et al., 2016; Sennrich et al., 2015), which are parametrically expensive, or to employ vocabulary selection strategies (L’Hostis et al., 2016). In both cases, we face the issue that words in the tail of the Zipfian distribution will typically still be too rare to learn good representations for through standard embedding methods. Some models, such as in the work of Ling et al. (2015), have sought to deal with the open vocabulary

problem by obtaining representations of words from characters. This is successful at capturing the semantics of morphological derivations (e.g. “running” from “run”) but puts significant pressure on the encoder to capture semantic distinctions amongst syntactically similar but semantically unrelated words (e.g. “run” vs. “rung”). Additionally, nothing about the spelling of named entities, e.g. “The Beatles”, tells you anything about their semantics (namely that they are a rock band).

In this paper, after reviewing related work in Section 2, we propose a new method for computing embeddings “on the fly”, which jointly addresses the large vocabulary problem and the paucity of data for learning representations in the long tail of the Zipfian distribution. This method, which we illustrate in Figure 1 and explain in Section 3 can be summarized as follows: at a high level, instead of directly learning representations for all words in a potentially unbounded vocabulary, we learn a network which predicts the representations of words based on auxiliary data. Such auxiliary data need only satisfy the general requirement that it describe some aspect of the semantics of the word for which a representation is needed. Examples of such data could be dictionary definitions, Wikipedia infoboxes, linguistic descriptions of named entities obtained from Wikipedia articles, or something as simple as the spelling of a word. We will refer to the content of auxiliary data as “definitions” throughout the paper, regardless of the source. Several sources of auxiliary data can be used simultaneously as input to a neural network that will compute an associated representation. These representations can then be used as the alternative representation for out-of-vocabulary words, or combined with within-vocabulary word embeddings directly trained on the task of interest or obtained from an external data source (Mikolov et al., 2013; Pennington et al., 2014). In the present paper, we will focus on a subset of these approaches and auxiliary data sources, restricting ourselves to producing out-of-vocabulary words embeddings from dictionary data, spelling, or both. Crucially, the auxiliary data encoders are trained jointly with the objective, ensuring the preservation of semantic alignment with representations which are directly learned from the data.

In Section 4, we aim to demonstrate that this “on the fly” embedding method provides improvements over models which learn embeddings directly from data. The obvious use case for this method would be datasets and tasks where there are many rare terms such as technical writing or bio/medical text (Deléger et al., 2016). On such datasets, attempting to learn global vectors—for example GloVe embeddings (Pennington et al., 2014)—from external data, would only provide coverage for common words and would be unlikely to be exposed to sufficient (or any) examples of domain-specific technical terms to learn good enough representations. However, there are no (or significantly fewer) established neural network-based baselines on these tasks, which makes it harder to validate baseline results. Instead, we present results on a trio of tasks, namely reading comprehension, recognizing textual entailment, and a variant on language modelling. For each task, we compare baseline models with embeddings trained against the objective to those same models with our on the fly embedding method. Additionally, we report results for those models with pretrained GLoVe vectors as input, which we do not update. We aim to show how the gap in results between the baseline and the data-rich GLoVe-based models can be partially but substantially closed merely through the introduction of relatively small amounts of auxiliary definitions. Quantitative results show that auxiliary data improves performance. Qualitative evaluation indicates our method allows models to draw and exploit connections defined in auxiliary data, along the lines of synonymy and semantic relatedness. We complete the paper in Section 5 by a discussion of the results across tasks, and suggesting further research directions using this model enhancement technique.

## 2 Related Work

Arguably, the most popular approach for representing rare words is by using word embeddings trained on very large corpora of raw text. (Mikolov et al., 2013; Pennington et al., 2014). Such embeddings are typically explicitly or implicitly based on word co-occurrence statistics. Being a big step forward from the models that are trained from scratch only on the task at hand, the approach can be criticized for being extremely data-hungry<sup>1</sup>. Obtaining the necessary amounts of data may be difficult, e.g. in technical domains. Besides, auxiliary training criteria used in the pretraining approaches are not guaranteed to yield representations that are useful for the task at hand.

There have been a number of attempts to achieve out-of-vocabulary generalization by relying on the spelling. (Ling et al., 2015) used a bidirectional LSTM to read the spelling of rare words and showed

---

<sup>1</sup>The GLoVe embeddings used are computed using 840 billion words of English text.

that this can be helpful for language modeling and POS tagging. We too will investigate spelling as a source of auxiliary data. In this respect, the approach presented here subsumes theirs, and can be seen as a generalization to other types of definitions.

The closest to our work is the study by Hill et al. (2016), in which a recurrent network is trained to produce an embedding of a dictionary definition that is close to the embedding of the headword. The network is shown to be an effective reverse dictionary and a crossword solver. Our approach is different in that we train a dictionary reader in an end-to-end fashion for a specific task, side-stepping the potentially suboptimal auxiliary ranking cost that was used in that earlier work. Their method also relies on the availability of high-quality pretrained embeddings which might not always be the case. Another related work by Long et al. (2016) uses dictionary definitions to provide initialization to a database embedding method, which is different from directly learning to use the definitions like we do.

At a higher level our approach belongs to the a broad family of methods for conditioning neural networks on external knowledge. For example, Larochelle et al. (2008) propose to add classes to a classifier by representing them using their “descriptions”. By description they meant, for example, a canonical picture of a printed character, that would represent all its possible handwritten versions. Their idea to rely on descriptions is similar to our idea to rely on definitions, however we focus on understanding complex inputs instead of adding new output classes.

Enhancing word embeddings with auxiliary data from knowledge bases (including wordnet) has a long tradition (Xu et al., 2014; Faruqui et al., 2015). Our work differs from previous approaches in essential ways. First, we use a textual form and are not restricted to knowledge represented as a graph. Second, we learn in an end to end fashion, allowing the model to pick useful information for the task of interest.

Finally, this work bears some relation, albeit distant, to the fast weights literature in the subfield of metalearning. In particular, Schmidhuber (1992) predicts updates to the weights of a “fast” network from the output of a “slow” network (which is trained by backpropagation) instead of updating them through gradient descent. Relatedly, Ha et al. (2016) explore architectures wherein an auxiliary network predicts the weights of a “slow” network directly. There are two key differences with the work presented here: first, the fast and slow networks observe the same input, whereas we explore scenarios whereby embeddings for one network are obtained from the output of another which observes auxiliary data; second, the auxiliary network predicts only embeddings, which can be seen as a subset of model parameters of the primary network, but leave the other parameters of the network to be learned as usual by gradient descent.

### 3 On the Fly Embeddings

In general, a neural network processes a language input by replacing its elements  $x_i$ , most often words, with the respective vectors  $e(x_i)$ , often called embeddings Bengio et al. (2003). Embeddings are typically either trained from scratch or pretrained. When embeddings are trained from scratch, a restricted vocabulary  $V_{train} = \{w_1, \dots, w_n\}$  is defined, usually based on training set frequency. Words not in  $V_{train}$  are replaced by a special token  $UNK$  with a trainable embedding  $e(UNK)$ . Unseen test-time words  $w \notin V_{train}$  are then represented by  $e(UNK)$ , which effectively means the specific meaning of this word is lost. Even if  $w$  had been included in  $V_{train}$  but was very rare, its learned embedding  $e(w)$  would likely not be very informative.

The approach proposed in this work, described in Figure 1, is to use definitions from auxiliary data, such as dictionaries, to compute embeddings of rare words. More specifically, this involves fetching a definition  $d(w) = (x'_1, \dots, x'_k)$  and feeding it into a network  $f$  that produces an embedding  $e_d(w) = f(e'(x'_1), \dots, e'(x'_k))$ . We will refer to  $e_d(w)$  a *definition embedding* produced by a *definition reader*  $f$ . One can either use the same embeddings  $e' = e$  when reading the dictionary or train different ones. Likewise, one can either stick to a shared vocabulary  $V_{dict} = V_{train}$ , or consider two different ones. When a word  $x'_i \notin V_{dict}$  is encountered, it is replaced by  $UNK$  and the respective trainable embedding  $e'(UNK)$  is used. For the function  $f$  we can consider two choices: a simple mean pooling  $e_d(w) = \sum_{i=1}^k e'(x'_i)/k$  or using the last state of an LSTM (Hochreiter and Schmidhuber, 1997),  $e_d(w) = LSTM(e'(x'_1), \dots, e'(x'_k))$ . Many words have multiple dictionary definitions. We combine embeddings for multiple definitions using mean pooling. We include all

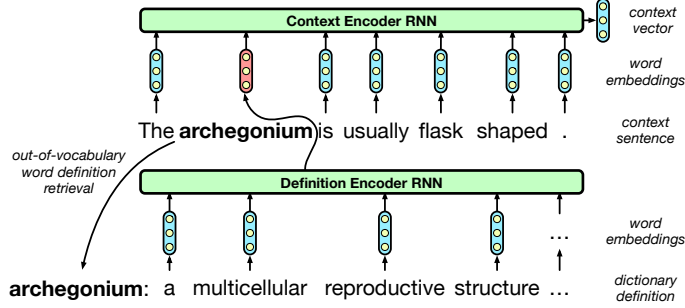


Figure 1: An example of how we deal with out-of-vocabulary words (indicated in bold). We obtain representations by retrieving external information (e.g. a dictionary definition) and embedding it with another LSTM-RNN, instead of using a catch-all “UNK” representation for out-of-vocabulary items.

definitions whose headwords match  $w$  or any possible lemma of a lower-cased  $w^2$ . To simplify the notation, the rest of the paper assumes that there is only one definition for each word.

While the primary purpose of definition embeddings  $e_d(w)$  is to inform the network about the rare words, they might also contain useful information for the words are in  $V_{train}$ . When we use both, we combine the information coming from the embeddings  $e(w)$  and the definition embeddings  $e_d(w)$  by computing  $e_c(w) = e(w) + We_d(w)$ , where  $W$  is a trainable matrix, or just by simply summing the two,  $e_c(w) = e(w) + e_d(w)$ . Alternatively it is possible to just use  $e_c(w) = e(w)$  for  $w$  from  $V_{train}$  and  $e_c(w) = e_d(w)$  otherwise. When no definition is available for a word  $w$ , we posit that  $e_d(w)$  is a zero vector. A crucial observation that makes an implementation of the proposed approach feasible is that the definitions  $d(x_i)$  of all words  $x_i$  from the input can be processed in parallel<sup>3</sup>.

We note, that there are least two aspects of our approach in which it can be considered very simplistic. First, we do not consider definition of word combinations, such as phrasal verbs like "give up" and geographical entities like "San Francisco". Second, our definition reader could better handle the unknown words  $w \notin V_{dict}$  by using their definition embeddings  $e_d(w)$  instead  $e'(UNK)$ , thereby implementing a form of recursion. We will investigate both in our future work.

## 4 Experiments

We worked on extractive question answering, semantic entailment classification and language modelling. For each task, we picked as baseline model and architecture from the literature we knew would provide sensible results, to explore how augmenting it with on the fly embeddings would affect performance. We explored two complementary sources of auxiliary data. First, we used word definitions from WordNet (Miller, 1995). While WordNet is mostly known for its structured information about synonyms, it does contain natural language definitions for all its 147306 lemmas (this also includes multi-word headwords which we do consider in this work)<sup>4</sup>. Second, we experimented with the character-level spelling of words as auxiliary data. To this end, in order to fit in with our use of dictionaries, we added fake definitions of the form “Word”  $\rightarrow$  “W”, “o”, “r”, “d”.

In order to measure the performance of models in “data-rich” scenarios where large amount of unlabelled language data is available for the training of word representations, we used as pretrained word embeddings 300-dimensional GLoVe vectors trained on 840 billion words (Pennington et al., 2014). We compared our auxiliary data-augmented on the fly embedding technique to baselines and models with fixed GLoVe embeddings to measure how well our technique closes the gap between a data-poor and data-rich scenario.

<sup>2</sup>We used the wordnet-based lemmatizer from NLTK.

<sup>3</sup>The same applies for all the words from a mini-batch of examples. By composing huge batches of up 10000 definitions from a mini-batch of examples, we were able to process them all in a reasonable time on GPUs.

<sup>4</sup>Advantages of using WordNet include its free availability and the ease of parsing, e.g., we used the NLTK (Bird, 2006) interface to extract the definitions.

#### 4.1 Question Answering

We used the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) that consists of approximately 100000 human-generated question-answer pairs. For each pair, a paragraph from Wikipedia is provided that contains the answer as a continuous span of words.

Our basic model is a simplified version of a coattention proposed in (Xiong et al., 2016). First, we represent the context of length  $n$  and the question of length  $m$  as matrices  $C \in \mathbb{R}^{n,d}$  and  $Q \in \mathbb{R}^{m,d}$  by running them through an LSTM and a linear transform. Next, we compute the affinity scores  $L = CQ^T$ . By normalizing  $L$  with row-wise and column-wise softmaxes we construct a context-to-question and question-to-context attention maps  $A_C$  and  $A_Q$ . These are used to construct a joint question-document representation  $U_0$  as a concatenation along the feature axis of the matrices  $C$ ,  $A_C Q$  and  $A_Q A_C^T C$ . We transform  $U_0$  with a bidirectional LSTM and another ReLU (Glorot et al., 2011) layer to obtain the final context-document representation  $U_2$ . Finally, two linear layers followed by a softmax assign to each position of the document probabilities of it being the beginning and the end of the answer span. We refer the reader to the work of Xiong et al. (2016) for more details. Compared to their model, the two main simplifications that we applied is skipping the iterative inference procedure and using usual ReLU instead of the highway-maxout units.

Our baseline is a model with the embeddings trained purely for scratch. We found that it performs best with a very small vocabulary of 3k most common words from the training set. This can be explained by a rather moderate size of the dataset: all the models we tried tended to overfit severely. In addition to using the spelling and the dictionary definitions we tried mixing the dictionary definitions with the spelling. The last model in our comparison is trained with the GLoVe embeddings. In this round of experiments we only used definitions for the out-of-vocabulary words.

The results are reported in Table 1. We report the exact match ratio as computed by the evaluation tools provided with the dataset, which is basically the accuracy of selecting the right answer. For the development set we report the average over four runs, for the test set we could evaluate only one model because the test was not released and evaluation was done manually by the dataset authors. Looking at the development set results one can see that the basic model (B) performs quite poorly, and adding any external information boosts the accuracy significantly (7.5% - 13.5%). Adding the spelling (S) helps more than adding a dictionary (D), but the model that has both (SD) has a 1.3% advantage over the model that uses just the spelling (S), demonstrating that combining several forms of auxiliary data allows the model to exploit the complementary information they provide. The model with GLoVe embeddings (G) is still ahead with a 3.3% margin, but the gap has been more than halved. Finally, the test set results confirm the main takeaway that SD performs the best, even though the relative order of S and D is different.

Table 1: Exact match (EM) ratio for different models on SQuAD development and tests set. The test results for GLoVe embeddings are missing because evaluation servers did not have enough memory to run our model.

model	EM dev	EM test
baseline (B)	47.06	-
spelling (S)	56.05	52.63
dict (D)	53.67	54.76
spelling+dict (SD)	<b>57.28</b>	<b>55.83</b>
glove (G)	60.54	-

To understand how our the model uses the dictionary and what prevents it from using it better we conducted a qualitative investigation on selected examples. Namely, we considered examples on which SD selected the correct answer span and S did not, and from them we chose the ones with the largest difference of log-likelihoods that S and SD assigned to the correct answer. Figure 2 shows the attention maps  $A_C$  for both models. We can see that S failed to match “autumn” and “season”, likely because their spelling is completely different. On the other hand SD does match the two words and our dictionary defines “autumn” as “the *season* when the leaves fall from the trees”. We saw the SD model match words “Tuesday” and “day”, “Cambridge” and “city”, “images” and “pictures”, “religion” and “Taoism”.

Furthermore, we compared the models G and SD in a similar way. We found that often SD simply missed a definition. For a example, it was not able to match “XBox” and “console”, “Coronation”<sup>5</sup> and “show”, “most-watched” and “most watched”. We also saw cases where the definition was available but was not used, seemingly because the key word in the definition was outside  $V_{train} = V_{dict}$ . For example, “arrow” was defined “a projectile with a straight thin shaft”, and the word “projectile” was quite rare in the training corpus. As a consequence, the model had no chances to understand that an arrow is a weapon and match the word “arrow” in the context with the word “weapon” in the question. Finally, we saw cases where inferring important aspects of meaning from the dictionary would be non-trivial, for example, guessing that “historian” is a “profession” from the definition “a person who is an authority on history and who studies it and writes about it” would involve serious common sense reasoning.

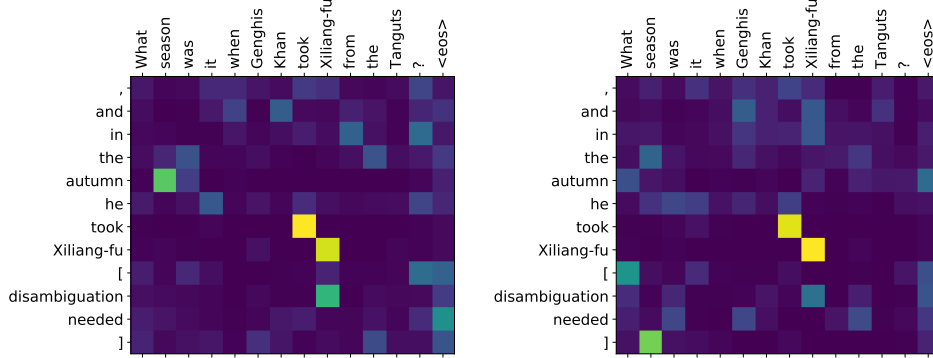


Figure 2: The attention maps  $A_C$  of the models with (on the left) and without the dictionary (on the right). One can how the dictionary helps to match the words “autumn” and “season”.

## 4.2 Entailment prediction

We used Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015), which consists of around 500k pairs of sentences (hypothesis and premise) and the task is to predict the logical relation (contradiction, neutral or entailment) between them. Our first model (denoted as “simple” in Table 2) is a slightly improved baseline from (Bowman et al., 2015). This model first computes an embedding of the two sentences by summing word vectors and concatenates them forming  $\mathbf{h} \in \mathbb{R}^{2D}$  embedding. The resulting representation is fed to a 2 layer ReLU MLP normalized by Batch Normalization (Ioffe and Szegedy, 2015).

In order to validate that improvements translate to state of the art models we implemented a variant (replacing TreeLSTM by biLSTM) of Enhanced Sequential Inference Model (ESIM) (Chen et al., 2016) that achieves close to SOTA accuracy. Similarly as in the model used in SQuAD experiments, this model represents hypothesis and premise as  $H \in \mathbb{R}^{n,d}$  and  $P \in \mathbb{R}^{m,d}$  matrices by encoding them using bidirectional LSTM. Analogously, alignment matrices  $A_H$  and  $A_P$  are computed by normalizing affinity scores. The alignment matrices are used to form joint hypothesis-premise representation. For hypothesis we compute and concatenate  $H$ ,  $A_H P$ ,  $H - A_H P$  and  $H \odot A_H P$  and repeat the same operations for premise forming  $\mathbf{h} \in \mathbb{R}^{n,4D}$  sentence embedding for each. These sentence representations are processed in parallel by bidirectional LSTM and the final states of the LSTMs are concatenated and used to predict entailment after processing by single layer Tanh MLP. Dropout is applied on word embeddings in the case of all models (with tuned rate), similarly as in SQuAD experiments we use small vocabulary of size 3000 or 5000. We never compute definition embeddings for top 1k words. Definition and word embeddings are combined using simple sum. All runs were repeated 3 times and scores are averaged.

Results of the simple and ESIM models are presented in Table 2. We include as baseline model using only embeddings trained from scratch and dictionary composed of word spellings. Results are directionally similar to results on SQuAD, auxiliary definitions bring 0.19% and 1.2% margin

<sup>5</sup>“Coronation Street” is a popular British TV show.

Table 2: Results on SNLI of ESIM and simple model.

	ESIM (valid / test)	Simple (valid / test)
baseline	83.39/82.84	80.00/79.63
spelling dict	83.78/82.89	81.86/81.17
dict	84.41/83.73	81.84/81.09
dict + spelling	<b>84.63/84.05</b>	<b>82.21/81.36</b>
glove	87.20/86.39	83.80/83.24

when used with ESIM and baseline models respectively<sup>6</sup>. One difference is that spelling was not as important for good performance on SNLI (spelling showed a smaller margin, and adding it to the dictionary definitions was also less helpful).

In order to aid understanding of the SNLI models performance we first plot t-SNE (van der Maaten and Hinton, 2008) of word embeddings computed using definitions, see Figure 3. We can see, for instance, that “bomber”<sup>7</sup> is close both to words like “jet” and “peach”, which would be challenging to learn using standard word embeddings. We also note that, as expected, dictionary-enabled models significantly outperform baseline models at sentences containing rare words, see Figure 3.

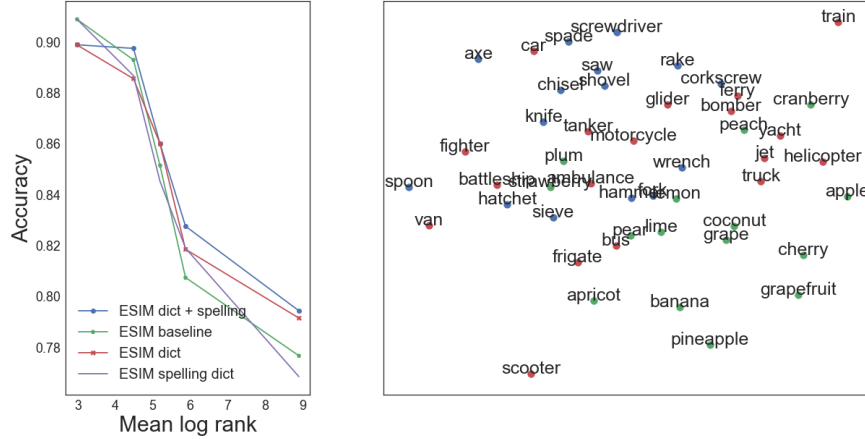


Figure 3: The left plot reports performance split by mean rank of word in the sentence of ESIM models. As expected, model using dictionary outperforms baselines on rare words. Right plot shows tSNE of word embeddings for 3 random categories from BLESS dataset (Baroni and Lenci, 2011) (denoted by color). Embeddings are computed using simple model on SNLI with full dictionary.

### 4.3 Language Modelling

In our first two experiments we used datasets of moderate size. To get an idea of how useful our auxiliary data sources will be on larger datasets, we conduct an experiment on the One Billion Words language modeling task (Chelba et al., 2014). Similarly to prior work on using the spelling (Ling et al., 2015) we restrict the softmax output layer to only predict probabilities of the 10k most common words, however, we do not impose such a constraint when the model reads the words. While such language models are not, strictly speaking, generative models, one can still study if having a definition of an observed word helps the model to predict the following ones from a restricted vocabulary.

Our baseline model is an LSTM with 500 units and it has trainable input embeddings for 10k input words. This covers around 90.24% of all the word occurrences. We consider predicting the embeddings of the other input words using GLoVe vector (G), spelling (S) and dictionary definitions (D-\*). Dictionary definitions were only available for 63.35% of the rest of occurrences whereas GloVe

<sup>6</sup>If the model selection is done using the dev set performance.

<sup>7</sup>A bomber is both an aircraft and a type of sandwich.

Table 3: Perplexities on the language modeling task. See Section 4.3 for the abbreviations.

model	ppl	ppl after defs	ppl after UNK
baseline	53.39	-	80.38
with spelling (S)	47.26	34.39	34.47
with glove (G)	46.69	34.50	34.58
with spelling (S-R)	48.51	35.48	41.63
with glove (G-R)	48.41	42.53	42.55
with trainable embeddings (T-R)	47.38	-	53.11
with dictionary (D-shared)	49.60	43.72	46.33
with dictionary (D-separate)	48.73	38.29	42.19
with dictionary (D-separate vocab.)	48.24	35.92	41.21

covers 97.43% of the rest and spelling is available everywhere. Thus, in order to compare how helpful the different auxiliary sources of information are, we tried restricted variants of GLoVe and spelling experiments (S-R, G-R, T-R), in which the embeddings or the spelling were only provided where a definition was available. We tried three varieties of models with the dictionary. In first one (D-shared), the same LSTM was used for reading the text and the dictionaries. In the second one (D-separate), separate LSTMs were used for these purposes, but they still shared the word embeddings. We also tried building a separate vocabulary to better cover the words from the dictionary definitions (D-separate vocab.). Lastly, we trained a model that had trainable embeddings for all the words with definitions as a reference.

We report the validation perplexities measured at iteration 200k in Table 3. Unsurprisingly, using external information to compute embeddings of unknown words helps in all cases. We observe a significant gain (3.6 units) even when for D-shared, which is remarkable as this model has the same parameters as the baseline. The perplexities of the best dictionary-equipped (D-shared) model and the restricted variants of other models (S-R, G-R) ended up being close (from 48.24 to 48.51) and not far from the reference (T-R). Thus, for the same coverage, the 3 models with additional information are comparable in performance. The regular perplexity is heavily influenced by how well the models handle the frequent words. To zoom in on how the models deal with rare words, we look at the perplexities of the words that (a) follow right after a word outside of the vocabulary (b) follow right after a word for which a definition is available. Interestingly, GLoVe embeddings were less helpful in situation (b) (42.53 vs 35.92). We note the strong performance of the models using spelling according to all our metrics. In our nearest future work we will try combine spelling with dictionary similarly to the first two experiments.

## 5 Discussion

We showed how different sources of auxiliary information, such as the spelling and a dictionary of definitions can be used to produce on the fly useful embeddings for rare words. While it was known before that adding the spelling information to the model is helpful, it is often hard or not possible to infer the meaning directly from the characters. Our more general approach offers endless possibilities of adding other data sources and learning end-to-end to extract the relevant bits information from them. Our experiments with a dictionary of definitions show the feasibility of the approach, as we report improvements over using just the spelling on question answering and semantic entailment classification tasks. Our qualitative investigations on the question answering data confirms our intuition on where the improvement comes from. It is also clear from them that adding more auxiliary data would help, and that it would probably be also useful to add definitions not just for words, but also for phrases (see “Coronation Street” from Section 4.1). We are planning to add more data sources (e.g. first sentences from Wikipedia articles) and better use the available ones (WordNet has definitions of phrasal verbs like “come across”) in our future work.

An important question that we did not touch in this paper is how to deal with rare words in the auxiliary information, such as dictionary definitions. Based on our qualitative investigations (see the example with “arrow” and “weapon” in Section 4.1), we believe that better handling rare words in the auxiliary information could substantially improve the proposed method. It would be natural to use on the fly embeddings similarly to the ones that we produce for words from the input, but a



straight-forward approach of computing them on request would be very computation and memory hungry. One would furthermore have to resolve cyclical dependencies, which are unfortunately common in dictionary data (when e.g. “entertainment” is defined using “diverting” and “diverting” is defined using “entertainment”). In our future work we want to investigate asynchronous training of on the fly embeddings and the main model.

## 6 Conclusion

In this paper, we have shown that introducing relatively small amounts of auxiliary data and a method for computing embeddings on the fly from these data bridges the gap between data-poor setups, where embeddings need to be learned directly from the end task, and data-rich setups, where embeddings can be pretrained and sufficient external data exists to ensure in-domain lexical coverage. A large representative corpus to pretrain word embeddings is not always available and our method is applicable when one has access only to limited auxiliary data. Learning end-to-end from auxiliary sources can be extremely data efficient when these sources represent compressed relevant information about the word, as dictionary definitions do. A related desirable aspect of our approach is that it may partially return the control over what a language processing system does into the hands of engineers or even users: when dissatisfied with the output, they may edit or add auxiliary information to the system to make it perform as desired. Furthermore, domain adaptation with our method could be carried out simply by using other sources of auxiliary knowledge, for example definitions of domain-specific technical terms in order to understand medical text. Overall, the aforementioned properties of our method make it a promising alternative to the existing approaches to handling rare words.

## Acknowledgements

We thank the developers of Theano (Theano Development Team, 2016) and Blocks (van Merriënboer et al., 2015) for their great work. We thank NVIDIA for giving access to their DGX-1 computers used in this work. Stanisław Jastrzębski was supported by Grant No. DI 2014/016644 from Ministry of Science and Higher Education, Poland. We thank Çağlar Gülçehre and Alexandre Lacoste for useful discussions.

## References

- Baroni, M. and Lenci, A. (2011). How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, GEMS ’11, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.
- Bird, S. (2006). Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.
- Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.
- Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2014). One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.
- Chen, Q., Zhu, X., Ling, Z., Wei, S., and Jiang, H. (2016). Enhancing and combining sequential and tree LSTM for natural language inference. *ArXiv*, abs/1609.06038.
- Deléger, L., Bossy, R., Chaix, E., Ba, M., Ferré, A., Bessieres, P., and Nédellec, C. (2016). Overview of the bacteria biotope task at bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop. Berlin: Association for Computational Linguistics*.
- Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E. H., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1606–1615.

- Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275.
- Ha, D., Dai, A., and Le, Q. V. (2016). Hypernetworks. *arXiv preprint arXiv:1609.09106*.
- Hill, F., Cho, K., Korhonen, A., and Bengio, Y. (2016). Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456.
- Larochelle, H., Erhan, D., and Bengio, Y. (2008). Zero-data learning of new tasks. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 646–651.
- Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015). Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.
- Long, T., Lowe, R., Cheung, J. C. K., and Precup, D. (2016). Leveraging lexical resources for learning entity embeddings in multi-relational data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.
- L’Hostis, G., Grangier, D., and Auli, M. (2016). Vocabulary selection strategies for neural machine translation.
- Mi, H., Wang, Z., and Ittycheriah, A. (2016). Vocabulary manipulation for neural machine translation. *arXiv preprint arXiv:1605.03209*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of ACM*, 38(11):39–41.
- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.
- Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.
- Schmidhuber, J. (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139.
- Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.
- Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.
- van der Maaten, L. and Hinton, G. (2008). Visualizing high-dimensional data using t-sne. *JMLR*.
- van Merriënboer, B., Bahdanau, D., Dumoulin, V., Serdyuk, D., Warde-Farley, D., Chorowski, J., and Bengio, Y. (2015). Blocks and fuel: Frameworks for deep learning. *arXiv:1506.00619 [cs, stat]*.
- Xiong, C., Zhong, V., and Socher, R. (2016). Dynamic coattention networks for question answering. In *International Conference on Learning Representations*.
- Xu, C., Bai, Y., Bian, J., Gao, B., Wang, G., Liu, X., and Liu, T.-Y. (2014). Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM ’14*, pages 1219–1228, New York, NY, USA. ACM.
- Zipf, G. K. (1949). Human behavior and the principle of least effort: An introduction to human ecology.