Definition Modeling: Learning to define word embeddings in natural language

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey
Department of Electrical Engineering & Computer Science
Northwestern University, Evanston IL 60208, USA
{nor, chenliang2013}@u.northwestern.edu, {l-birnbaum, d-downey}@northwestern.edu

Abstract
Distributed representations of words have been shown to capture lexical semantics, as demonstrated by their effectiveness in word similarity and analogical relation tasks. But, these tasks only evaluate lexical semantics indirectly. In this paper, we study whether it is possible to utilize distributed representations to generate dictionary definitions of words, as a more direct and transparent representation of the embeddings’ semantics. We introduce definition modeling, the task of generating a definition for a given word and its embedding. We present several definition model architectures based on recurrent neural networks, and experiment with the models over multiple data sets. Our results show that a model that controls dependencies between the word being defined and the definition words performs significantly better, and that a character-level convolution layer designed to leverage morphology can complement word-level embeddings. Finally, an error analysis suggests that the errors made by a definition model may provide insight into the shortcomings of word embeddings.

1 Introduction
Distributed representations of words, or word embeddings, are a key component in many natural language processing (NLP) models (Turian, Ratino, and Bengio 2010; Huang et al. 2014). Recently, several neural network techniques have been introduced to learn high-quality word embeddings from unlabeled textual data (Mikolov et al. 2013a; Pennington, Socher, and Manning 2014; Yogatama et al. 2015). Embeddings have been shown to capture lexical syntax and semantics. For example, it is well-known that nearby embeddings are more likely to represent synonymous words (Landauer and Dumais 1997), or words in the same class (Downey, Schoenmackers, and Etzioni 2007). More recently, the vector offsets between embeddings have been shown to reflect analogical relations (Mikolov, Yih, and Zweig 2013). However, tasks such as word similarity and analogy only evaluate an embedding’s lexical information indirectly.

In this work, we study whether word embeddings can be used to generate natural language definitions of their corresponding words. Dictionary definitions serve as direct and explicit statements of word meaning. Thus, compared to the word similarity and analogical relation tasks, definition generation can be considered a more transparent view of the syntax and semantics captured by an embedding. We introduce definition modeling: the task of estimating the probability of a textual definition, given a word being defined and its embedding. Specifically, for a given set of word embeddings, a definition model is trained on a corpus of word and definition pairs. The models are then tested on how well they model definitions for words not seen during the training, based on each word’s embedding.

The definition models studied in this paper are based on recurrent neural network (RNN) models (Elman 1990; Hochreiter and Schmidhuber 1997). RNN models have established a new state-of-the-art performance on many sequence prediction and natural language generation tasks (Cho et al. 2014; Sutskever, Vinyals, and Le 2014; Karpathy and Fei-Fei 2014; Wen et al. 2015a). An important characteristic of dictionary definitions is that only a subset of the words in the definition depend strongly on the word being defined. For example, the word “woman” in the definition of “feminine” in Table 1 depends on the word being defined than the rest. To capture the varying degree of dependency, we introduce a gated update function that

| Word       | Generated definition          |
|------------|--------------------------------|
| brawler    | a person who fights            |
| butterfish | a marine fish of the atlantic coast |
| continually | in a constant manner          |
| creek      | a narrow stream of water       |
| feminine   | having the character of a woman |
| juvenility | the quality of being childish  |
| mathematical | of or pertaining to the science of |
| mathematics |                            |
| negotiate  | to make a contract or agreement |
| prance     | to walk in a lofty manner      |
| resent     | to have a feeling of anger or dislike |
| similar    | having the same qualities      |
| valueless  | not useful                     |

Table 1: Selected examples of generated definitions. The model has been trained on occurrences of each example word in running text, but not on the definitions.
is trained to control information of the word being defined used for generating each definition word. Furthermore, since the morphemes of the word being defined plays a vital role in the definition, we experiment with a character-level convolutional neural network (CNN) to test whether it can provide complementary information to the word embeddings. Our best model can generate fluent and accurate definitions as shown in Table 1. We note that none of the definitions in the table exactly match any definition seen during training.

Our contributions are as follows: (1) We introduce the definition modeling task, and present a probabilistic model for the task based on RNN language models. (2) In experiments with different model architectures and word features, we show that the gate function improves the perplexity of a RNN language model on definition modeling task by \( \sim 10\% \), and the character-level CNN further improves the perplexity by \( \sim 5\% \). (3) We also show that the definition models can be use to perform the reverse dictionary task studied in previous work, in which the goal is to match a given definition to its corresponding word. Our model achieves an 11.8% absolute gain in accuracy compared to previous state-of-the-art by Hill et al. (2016). (4) Finally, our error analysis shows that a well-trained set of word embeddings pays significant role in the quality of the generated definitions, and some of error types suggest shortcomings of the information encoded in the word embeddings.

## 2 Previous Work

Our goal is to investigate RNN models that learns to define word embeddings by training on examples of dictionary definitions. While dictionary corpora have been utilized extensively in NLP, to the best of our knowledge none of the previous work has attempted create a generative model of definitions. Early work focused on extracting semantic information from definitions. For example, Chodorow (1985), and Klavans and Whitman (2001) constructed a taxonomy of words from dictionaries. Dolan et al. (1993) and Vanderwende et al. (2005) extracting semantic representations from definitions, to populate a lexical knowledge base.

In distributed semantics, words are represented by a dense vector of real numbers, rather than semantic predicates. Recently, dictionary definitions have been used to learn such embeddings. For example, Wang et al. (2015) used words in definition text as a form of “context” words for the Word2Vec algorithm (Mikolov et al. 2013b). Hill et al. (2016) use dictionary definitions to model compositionality, and evaluate the models with the reverse dictionary task. While these works learn word or phrase embeddings from definitions, we only focus on generating definitions from existing (fixed) embeddings. Our experiments show that our models outperform those of Hill et al. (2016) on the reverse dictionary task.

Our work employs embedding models for natural language generation. A similar approach has been taken in a variety of recent work on tasks distinct from ours. Dinu and Baroni (2014) present a method that uses embeddings to map individual words to longer phrases denoting the same meaning. Likewise, Li et al. (2015) study how to encode a paragraph or document as an embedding, and reconstruct the original text from the encoding. Other recent work such as the image caption generation (Karpathy and Fei-Fei 2014) and spoken dialog generation (Wen et al. 2015a) are also related to our work, in that a sequence of words is generated from a single input vector. Our model architectures are inspired by sequence-to-sequence models (Cho et al. 2014) and Sutskever, Vinyals, and Le (2014), but definition modeling is distinct, as it is a word-to-sequence task.

## 3 Dictionary Definitions

In this section, we first investigate definition content and structure through a study of existing dictionaries. We then describe our new data set, and define our tasks and metrics.

### 3.1 Definition Content and Structure

In existing dictionaries, individual definitions are often comprised of genus and differentiae (Chodorow, Byrd, and Heidorn 1985; Montemagni and Vanderwende 1992). The genus is a generalized class of the word being defined, and the differentiae is what makes the word distinct from others in the same class. For instance,

- **Phosphorescent**: emitting light without appreciable heat as by slow oxidation of phosphorous

- “emitting light” is a genus, and “without applicable heat ...” is a differentiae. Furthermore, definitions tend to include common patterns such as “the act of ...” or “one who has ...” (Markowitz, Ahlswede, and Evens 1986). However, the patterns and styles are often unique to each dictionary.

- The genus + differentiae (G+D) structure is not the only pattern for definitions. For example, the entry below exhibits distinct structures.

  **Eradication**: the act of plucking up by the roots; a rooting out; extirpation; utter destruction

- This set of definitions includes a synonym (“extirpation”), a reverse of the G+D structure (“utter destruction”), and an uncategorized structure (“a rooting out”).

### 3.2 Corpus: Preprocessing and Analysis

Dictionary corpora are available in a digital format, but are designed for human consumption and require preprocessing before they can be utilized for machine learning. Dictionaries contain non-definitional text to aid human readers, e.g. the entry for “gradient” in Wordnik contains fields (“Mathematics”) and example usage (“as, the gradient line of a railroad.”). Further, many entries contain multiple definitions, usually (but not always) separated by “;”.

We desire a corpus in which each entry contains only a word being defined and a single definition. We parse dictionary entries from GCIDE and preprocess WordNet’s glosses, and the fields and usage are removed. The parsers and preprocessing scripts can be found at https://github.com/northanapon/dict-definition.

---

1https://www.wordnik.com/words/gradient
2http://gcide.gnu.org.ua/
Table 2: Basic statistics of the common word definitions corpus. Splits are mutually exclusive in the words being defined.

| Split   | train | valid | test  |
|---------|-------|-------|-------|
| #Words  | 20,298| 1,127 | 1,129 |
| #Entries| 146,486| 8,087 | 8,352 |
| #Tokens | 1,406,440| 77,948 | 79,699 |
| Avg length | 6.60 | 6.64 | 6.54 |

Table 3: The number of manually labeled structures of dictionary definitions in WordNet (WN) and GCIDE (GC). G+D is genus followed by differentiae, and D+G is the reverse. Syn is a synonym. The words defined in the example column are “lamine”, “fawn”, “activity”, “diagonally”, and “absolutely”.

| Label | WN   | GC   | Example                                                                 |
|-------|------|------|-------------------------------------------------------------------------|
| G+D   | 85%  | 50%  | to divide into thin plates                                             |
| D+G   | 7%   | 9%   | a young deer                                                            |
| Syn   | 1%   | 32%  | energy                                                                  |
| Misc  | 4%   | 8%   | in a diagonal direction                                                |
| Error | 3%   | 1%   | used as intensifiers                                                    |
| Total | 256  | 424  |                                                                         |

To create a corpus of reasonable size for machine learning experiments, we sample around 20k words from the 50k most frequent words in the Google Web 1T corpus (Brants and Franz 2006), removing function words. In addition, we limit the number of entries for each word in a dictionary to three before the splitting by “;” (so that each word being defined may repeat multiple times in our corpus). After cleaning and pruning, the corpus has a vocabulary size of 29k. Other corpus statistics are shown in Table 2.

We analyze the underlying structure of the definitions in the corpus by manually labeling each definition with one of four structures: G+D, D+G, Syn (synonym), and Misc. In total, we examine 680 definitions from 100 randomly selected words. The results are shown in Table 3. We reaffirm earlier studies showing that the G+D structure dominates in both dictionaries. However, other structures are also present, highlighting the challenge inherent in the dictionary modeling task. Further, we observe that the genus term is sometimes general (e.g., “one” or “that”), and other times specific (e.g., “an advocate”).

### 3.3 Dictionary Definition Tasks

In the definition modeling (DM) task, we are given an input word $w^*$, and output the likelihood of any given text $D$ being a definition of the input word. In other words, we estimate $P(D|w^*)$. We assume our definition model has access to a set of word embeddings, estimated from some corpus other than the definition corpus used to train the definition model.

DM is a special case of language modeling, and as in language modeling the performance of a definition model can be measured by using the perplexity of a test corpus. Lower perplexity suggests that the model is more accurate at capturing the definition structures and the semantics of the word being defined.

Besides perplexity measurement, there are other tasks that we can use to further evaluate a dictionary definition model including definition generation, and the reverse and forward dictionary tasks. In definition generation, the model produces a definition of a test word. In our experiments, we evaluate generated definitions using both manual examination and BLEU score, an automated metric for generated text. The reverse and forward dictionary tasks are ranking tasks, in which the definition model ranks a set of test words based on how likely they are to correspond to a given definition (the reverse dictionary task) or ranks a set of test definitions for a given word (the forward dictionary task) (Hill et al. 2016). A dictionary definition model achieves this by using the predicted likelihood $P(D|w^*)$ as a ranking score.

### 4 Models

The goal of a definition model is to predict the probability of a definition $D = \{w_1, ..., w_T\}$ given a word being defined $w^*$. Our model assumes that the probability of generating the $t$th word $w_t$ of a definition text depends on both the previous words and the word being defined (Eq 1). The probability distribution is usually approximated by a softmax function (Eq 2).

$$p(D|w^*) = \prod_{t=1}^T p(w_t|w_1, ..., w_{t-1}, w^*)$$  (1)

where $W_j$ is a matrix of parameters associated with word $j$, $h_t$ is a vector summarizing inputs so far at token $t$, and $\tau$ is a hyper-parameter for temperature, set to be 1 unless specified. Note that in our expression, the word being defined $w^*$ is present at all time steps as an additional conditioning variable.

The definition models explored in this paper are based on a recurrent neural network language model (RNNLM) (Mikolov et al. 2010). An RNNLM is comprised of RNN units, where each unit reads one word $w_t$ at every time step $t$ and outputs a hidden representation $h_t$ for Eq 2.

$$h_t = g(v_{t-1}, h_{t-1}, v^*)$$  (3)

where $g$ is a recurrent nonlinear function, $v_t$ denotes the embedding (vector representation) of the word $w_t$, and $v^*$ is likewise the embedding of the word being defined.

### 4.1 Model Architectures

A natural method to condition an RNN language model is to provide the network with the word being defined at the first step, as a form of “seed” information. The seed approach has been shown to be effective in RNNs for other tasks (Kalchbrenner and Blunsom 2013, Karpathy and Fei-Fei 2014). Here, we follow the simple method of Sutskever et al., (2011), in which the seed is added at the beginning of the text. In our case, the word being defined is added to the beginning of the definition. Note
that we ignore the predicted probability distribution of the seed itself at test time.

Section 4.2 shows that definitions exhibit common patterns. We hypothesize that the word being defined should be given relatively more important for portions of the definition that carry semantic information, and less so for patterns or structures comprised of function and stop words. Further, Wen et al. (2015) show that providing constant seed input at each time step can worsen the overall performance of spoken dialog generators.

Thus, inspired by the GRU update gate (Cho et al. 2014), we update the output of the recurrent unit with GRU-like update function as:

\[
\begin{align*}
    z_t &= \sigma(W_z[v^*; h_{t-1}] + b_z) \\
    r_t &= \sigma(W_r[v^*; h_{t-1}] + b_r) \\
    \tilde{h}_t &= \tanh(W_h[(r_t \odot v^*); h_{t-1}] + b_h) \\
    h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{align*}
\]

where \(\sigma\) is the sigmoid function, \([a; b]\) denotes vector concatenation, and \(\odot\) denotes element-wise multiplication. \(h_t\) from Eq 3 is updated as given in Eq 7. At each time step, \(z_t\) is an update gate controlling how much the output from RNN unit changes, and \(r_t\) is a reset gate controlling how much information from the word being defined is allowed. We name this model Gated (G).

In the rest of this subsection, we present three baseline model architectures that remove portions of Gated. In our experiments, we will compare the performance of Gated against the baselines in order to measure the contribution of each portion of our architecture. First, we reduce the model into a standard RNNLM, where

\[
h_t = g(v_{t-1}, h_{t-1})
\]

The standard model only receives information about \(w^*\) at the first step (from the seed). We refer to this baseline as Seed (S).

A straightforward way to incorporate the word being defined throughout the definition is simply to provide its embedding \(v^*\) as a constant input at every time step. We refer to this model as Input (I):

\[
h_t = g([v^*; v_{t-1}], h_{t-1})
\]

(Mikolov and Zweig 2012). Alternatively, the model could utilize \(v^*\) to update the hidden representation from the RNN unit, named Hidden (H). The update function for Hidden is:

\[
h_t = \tanh(W_h[v^*; h_{t-1}] + b_h)
\]

where \(W_h\) is a weight matrix, and \(b_h\) is the bias. In Hidden we update \(h_t\) from Eq 3 using Eq 10. This is similar to the GRU-like architecture in Eq 8 without the gates (i.e. \(r_t\) and \(z_t\) are always vectors of 1s).

4.2 Other Features

In addition to model architectures, we explore whether other word features derived from the word being defined can provide complementary information to the word embeddings. We focus on two different features: affixes, and hypernym embeddings. To add these features within DM, we simply concatenate the embedding \(v^*\) with the additional feature vectors.

Affixes Many words in English and other languages consist of composed morphemes. For example, a word “capitalist” contains a root word “capital” and a suffix “-ist”. A model that knows the semantics of a given root word, along with knowledge of how affixes modify meaning, could accurately define any morphological variants of the root word. However, automatically decomposing words into morphemes and deducing the semantics of affixes is not trivial.

We attempt to capture prefixes and suffixes in a word by using character-level information. We employ a character-level convolution network to detect affixes (LeCun et al. 1990). Specifically, \(w^*\) is represented as a sequence of characters with one-hot encoding. A padding character is added to the left and the right to indicate the start and end of the word. We then apply multiple kernels of varied lengths on the character sequence, and use max pooling to create the final features (Kim et al. 2016). We hypothesize that the convolution input, denoted as CH, will allow the model to identify regularities in how affixes alter the meanings of words.

Hyponym Embeddings As we discuss in Section 3, dictionary definitions often follow a structure of genus + differrentia. We attempt to exploit this structure by providing the model with knowledge of the proper genus, drawn from a database of hyponym relations. In particular, we obtain the hyponyms from WebLSA database (Seitner et al. 2016) which employs Hearst-like patterns (Hearst 1992) to extract hyponym relations from the Web. We then provide an additional input vector, referred to as HE, to the model that is equal to the weighted sum of the top \(k\) hyponyms in the database for the word being defined. In our experiments \(k = 5\) and the weight is linearly proportional to the frequency in the resource. For example, the top 5 hyponyms and frequencies for “fawn” are “color”:149, “deer”:135, “animal”:132.0, “wildlife”:82.0, “young”:68.0.

5 Experiments and Results

We now present our experiments evaluating our definition models. We train multiple model architectures using the train set and evaluate the model using the test set on all of the three tasks described in Section 3.3. We use the valid set to search for the learning hyper-parameters. Note that the words being defined are mutually exclusive across the three sets, and thus our experiments evaluate how well the models generalize to new words, rather than to additional definitions or senses of the same words.

All of the models utilize the same set of fixed, pre-trained word embeddings from the Word2Vec project and a 2-layer LSTM network as an RNN component (Hochreiter and Schmidhuber 1997). The embedding and LSTM hidden layers have 300 units each. For the affix detector, the character-level CNN has kernels of length 2-6 and size \{10, 30, 40, 40, 40\} with a stride of 1. During training, we maximize the log-likelihood objective using Adam, a variation of stochastic gradient decent (Kingma and Ba 2014). The learning rate is 0.001, and
5.1 Definition Modeling

First, we compare our different methods for utilizing the word being defined within the models. The results are shown in the first section of Table 4. We see that the gated update (S+G) improves the performance of the Seed, while the other architectures (S+I and S+H) do not. The results are consistent with our hypothesis that the word being defined is more relevant to some words in the definition than to others, and the gate update can identify this. We explore the behavior of the gate further in Section 6.

Next, we evaluate the contribution of the linguistic features. We see that the affixes (S+G+CH) further improves the model, suggesting that character-level information can complement word embeddings learned from context. Perhaps surprisingly, the hypernym embeddings (S+G+CH+HE) have an unclear contribution to the performance. We suspect that the average of multiple embeddings of the hypernym words may be a poor representation of the genus in a definition. More sophisticated methods for harnessing hypernyms are an item of future work.

5.2 Definition Generation

In this experiment, we evaluate the quality of the definitions generated by our models. We compute BLEU score between the output definitions and the dictionary definitions to measure the quality of the generation. The decoding algorithm is simply sampling a token at a time from the model’s predicted probability distribution of words. We sample 40 definitions for each word being defined, using a temperature (τ in Eq 2) that is close to a greedy algorithm (0.05 or 0.1, selected from the valid set) and report the average BLEU score. For help in interpreting the BLEU scores, we also report the scores for three baseline methods that output definitions found in the training or test set. The first baseline, Inter, returns the definition of the test set word from the other dictionary. Its score thus reflects that of a definition that is semantically correct, but differs stylistically from the target dictionary. The other baselines (NE-WN and NE-GC) return the definition from the training set for the embedding nearest to that of the word being defined. In case of a word having multiple definitions, we micro-average BLEU scores before averaging an overall score.

Table 5 shows the BLEU scores of the generated definitions given different reference dictionaries. AVG and Merge in the table are two ways of aggregating the BLEU scores. AVG averages the BLEU scores by using each dictionary as the ground truth. The Merge computes score by using union of the two dictionaries. First, we can see that the baselines have low BLEU scores when evaluated on definitions from the other dictionary (Inter and NE-). This shows that different dictionaries use different styles. However, despite the fact that our best model S+G+CH is unaware of which dictionary it is evaluated against, it generates definitions that strike a balance between both dictionaries, and achieves higher BLEU scores overall. As in the earlier experiments, the Gated model improves the most over the Seed model. In addition, the affixes further improves the performance while the contribution of the hypernym embeddings is unclear on this task.

It is worth noting that many generated definitions contain a repeating pattern (i.e. “a metal, or other materials, or other materials”). We take the definitions from the language model (Seed) and our full system (S+G+CH+HE), and clean the definitions by retaining only one copy of the repeated phrases. We also only output the most likely definition for each word. The BLEU score increases by 2 (Seed* and S+G+CH+HE*). We discuss about further analysis and common error types in Section 6.

5.3 Reverse and Forward Dictionary

In the dictionary tasks, the models are evaluated by how well they rank words for given definitions (RVD) or definitions for words (FWD). We compare against models from previous work on the reverse dictionary task (Hill et al. 2016). The previous models read a definition and output an embedding, then use cosine similarity between the output embedding and the word embedding as a ranking score. There are two ways to compose the output embedding: BOW w2v cosine uses vector addition and linear projection, and KNN w2v...
Table 6: Model performance on Reverse (RVD) and Forward (FWD) Dictionary tasks

| Model       | #Params | RVD @1 | RVD @10 | RVD R-Prec | FWD @1 | FWD @2 | FWD @3 | FWD @4 | Avg |
|-------------|---------|--------|---------|------------|--------|--------|--------|--------|-----|
| BOW w2v cosine | 0.09m   | 0.106  | 0.316   | -          | -      |        |        |        |     |
| RNN w2v cosine | 1.82m   | 0.190  | 0.452   | -          | -      |        |        |        |     |
| Seed        | 10.2m   | 0.175  | 0.465   | 0.163      | -      |        |        |        |     |
| S+I         | 10.6m   | 0.187  | 0.492   | 0.169      | -      |        |        |        |     |
| S+H         | 10.4m   | 0.286  | 0.573   | 0.282      | -      |        |        |        |     |
| S+G         | 10.8m   | 0.293  | 0.581   | 0.282      | -      |        |        |        |     |
| S+G+CH      | 11.1m   | 0.307  | 0.600   | 0.298      | -      |        |        |        |     |
| S+G+CH+HE*  | 11.7m   | 0.308  | 0.608   | 0.304      | -      |        |        |        |     |

Table 7: Percentage of times a definition is manually ranked in each position (@k), and average rank (Avg).

| Choices | @1 | @2 | @3 | @4 | Avg |
|---------|----|----|----|----|-----|
| Dictionary | 58.3 | 21.9 | 7.72 | 10.1 | 1.64 |
| NE | 16.3 | 22.8 | 27.85 | 37.0 | 2.75 |
| Seed* | 6.8 | 23.5 | 35.23 | 35.1 | 2.92 |
| S+G+CH+HE* | 18.7 | 31.8 | 29.19 | 17.8 | 2.41 |

The figure shows that the gate makes the output distribution more effective. We investigate this advantage by plotting the average gate activation ($z$ and $r$ in Eq 4 and 5) in Figure 1.

The $r$ gate is split into 3 parts corresponding to the embedding, character information, and the hypernym embedding. The figure shows that the gate makes the output distribution more dependent on the word being defined when predicting content words, and less so for function words. The hypernym embedding does not contribute to the performance and its gate activation is relatively constant. Additional examples can be found in the supplementary material.

Finally, we present a comparison of definitions generated from different models to gain a better understanding of the models. Table 8 shows the definitions of three words from Table 1. The Random Embedding method does not generate good definitions. The nearest embedding method NE returns a similar definition, but often makes important errors (e.g., “feminine” vs “masculine”). The models using the gated update function generate better definitions, and the character-level information is often informative for selecting content words (e.g. “mathematics” in “mathematical”).

### 6.2 Error Analysis

In our manual error analysis of 200 labeled definitions, we find that 140 of them contain some degree of error. Table 9 shows the primary error types, with examples. Types (1) to (3) are fluency problems, and likely stem from the definition
model, rather than shortcomings in the embeddings.

We believe the other error types stem more from semantic gaps in the embeddings than from limitations in the definition model. Our reasons for placing the blame on the embeddings rather than the definition model itself are twofold. First, we perform an ablation study in which we train \( S+G+CH \) using randomized embeddings, rather than the learned Word2Vec ones. The performance of the model is significantly worsened (the test perplexity is 100.43, and the BLEU scores are shown in Table 5), which shows that the good performance of our definition models is in significant measure due to the embeddings. Secondly, the error types (4) - (6) are plausible shortcomings of embeddings, some of which have been reported in the literature. These erroneous definitions are partially correct (often the correct part of speech), but are missing details that may not appear in contexts of the word due to reporting bias (Gordon and Van Durme 2013). For example, the word “captain” often appears near the word “ship”, but the role (as a leader) is frequently implicit. Likewise, embeddings are well-known to struggle in capturing antonym relations (Argerich, Torre Zaffaroni, and Cano 2016), which helps explain the opposite definitions output by our model.

### 7 Conclusion

In this work, we introduce the definition modeling task, and investigate whether word embeddings can be used to generate definitions of the corresponding words. We evaluate different architectures based on a RNN language model on definition generation and the reverse and forward dictionary tasks. We find the gated update function that controls the influence of the word being defined on the model at each time step improves accuracy, and that a character-level convolutional layer further improves performance. Our error analysis shows a well-trained set of word embeddings is crucial to the models, and that some failure modes of the generated definitions may provide insight into shortcomings of the word embeddings. In future work, we plan to investigate whether definition models can be utilized to improve word embeddings or standard language models.

### 8 Acknowledgments

This work was supported in part by NSF Grant IIS-1351029 and the Allen Institute for Artificial Intelligence. We thank Chandra Sekhar Bhagavatula for helpful comments.

### References

[Argerich, Torre Zaffaroni, and Cano 2016] Argerich, L.; Torre Zaffaroni, J.; and Cano, M. J. 2016. Hash2Vec, Feature Hashing for Word Embeddings. ArXiv e-prints.

[Brants and Franz 2006] Brants, T., and Franz, A. 2006. Web 1t 5-gram, ver. 1. LDC2006T13.

[Cho et al. 2014] Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder.
[Wang, Mohamed, and Hirst 2015] Wang, T.; Mohamed, A.; and Hirst, G. 2015. Learning lexical embeddings with syntactic and lexicographic knowledge. In ACL 2015, 458–463. Association for Computational Linguistics.

[Wen et al. 2015a] Wen, T.-H.; Gasic, M.; Mrki, N.; Su, P.-H.; Vandyke, D.; and Young, S. 2015a. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In EMNLP 2015, 1711–1721. Association for Computational Linguistics.

[Wen et al. 2015b] Wen, T.-H.; Gasic, M.; Kim, D.; Mrksic, N.; Su, P.-H.; Vandyke, D.; and Young, S. 2015b. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In SIGDIAL 2015, 275–284. Association for Computational Linguistics.

[Yogatama et al. 2015] Yogatama, D.; Manaal, F.; Chris, D.; and Noah A., S. 2015. Learning word representations with hierarchical sparse coding. In Proceedings of The 32nd International Conference on Machine Learning, volume 37 of ICML ’15. Journal of Machine Learning Research.