Learning to Define Terms in the Software Domain

Vidhisha Balachandran **Dheeraj Rajagopal** **Rose Catherine** **William Cohen**
School of Computer Science
Carnegie Mellon University, Pittsburgh, USA
{vbalacha,dheeraj,rosecatherinek,wcohen}@cs.cmu.edu

Abstract

One way to test a person's knowledge of a domain is to ask them to define domain-specific terms. Here, we investigate the task of automatically generating definitions of technical terms by reading text from the technical domain. Specifically, we learn definitions of software entities from a large corpus built from the user forum Stack Overflow. To model definitions, we train a language model and incorporate additional domain-specific information like word co-occurrence, and ontological category information. Our approach improves previous baselines by 2 BLEU points for the definition generation task. Our experiments also show the additional challenges associated with the task and the short-comings of language-model based architectures for definition generation.

1 Introduction

Dictionary definitions have been previously used in various Natural Language Processing (NLP) pipelines like knowledge base population (Dolan et al., 1993), relationship extraction, and extracting semantic information (Chodorow et al., 1985). Creating dictionaries in a new domain is time consuming, often requiring hand curation by domain experts with significant expertise. Developing systems to automatically learn and generate definitions of words can lead to greater time-efficiency (Muresan and Klavans, 2002). Additionally, it helps accelerate resource-building efforts for any new domain.

In this paper, we study the task of generating definitions of domain-specific entities. In particular, our goal is to generate definitions for technical terms with the freely available Stack Overflow[1] (SO) as our primary corpus. Stack Overflow is a technical question-and-answer forum aimed at supporting programmers in various aspects of computer science. Each question is tagged with associated entities or "tags", and the top answers are ranked based on user upvotes (de Souza et al., 2014). Figure 1 shows a screenshot from the forum of a question and the entities tagged with the question. Our work explores the challenge of generating definitions of entities in SO using the background data of question-answer pairs and their associated tags.

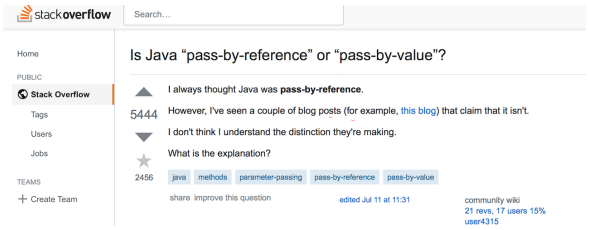

Figure 1: Screenshot from Stack Overflow showing questions and corresponding tags in blue boxes

Our base definition generation model is adapted from Noraset et al. (2017), a Recurrent Neural Network (RNN) language model to learn to generate definitions for common English words using embeddings trained on Google News Corpus. Over this base model, we leverage the distributed word information via the embeddings trained on domain specific Stack Overflow corpus. We improve this model to additionally incorporate domain-specific information such as co-occurring entities and domain ontology in the definition generation process. Our model also uses an additional loss function to reconstruct the entity word representation from the generated sequence. Our best model can generate definitions in software domain with a BLEU score of 10.91, improving upon the baseline by 2 points.

In summary, our contributions are as follows,

1. We propose a new dataset of entities in the

[1] https://stackoverflow.com

software domain and their corresponding definitions, for the definition generation task.

2. We provide ways to incorporate domain-specific knowledge such as co-occurring entities and ontology information into a language model trained for the definition generation task.

3. We study the effectiveness of the model using the BLEU (Papineni et al., 2002) metric and present the results and discussion about our results.

Section 4 of this paper presents the dataset. Section 5 discusses the model in detail. In Section 6 and 7 we present the experimental details and results of our experiments. Section 8 provides an analysis and discussion of the results and the generated definitions.

2 Related Work

Definition modeling: The closest work related to ours is Noraset et al. (2017) who learn to generate definitions for general English words using a RNN language model initialized with pre-trained word embeddings. We adapt the method proposed by them and use it in a domain-specific construct. We aim to learn definitions of entities in the software domain. Hill et al. (2015) learn to produce distributed embeddings for words using their dictionary definitions as a means to bridge the gap between lexical and phrase semantics. Similarly, Tissier et al. (2017) use lexical definitions to augment the Word2Vec algorithm by adding an objective of reconstructing the words in the definition. In contrast, we focus solely on generating the definitions of entities. We add an objective of reconstructing the embedding of the word from the generated sequence. Also, all the above work focus on lexical definitions of general English words, while we focus on closed domain software terms. Dhingra et al. (2017) present a dataset of cloze-style queries constructed from definitions of software entities on Stack Overflow. In contrast to their work, we focus on generating the entire definition of entities.

RNN Language Models : We use RNN based language models, conditioned on the term to be defined and its ontological category, to generate definitions. Such neural language models have been shown to successfully work for image-captioning tasks (Karpathy and Fei-Fei, 2015; Kiros et al., 2014), concept to text generation (Lebret et al., 2016; Mei et al., 2015), machine translation (Luong et al., 2014; Bahdanau et al., 2014) and conversations and dialog systems (Shang et al., 2015; Wen et al., 2015).

Reconstruction Loss Framework : We also build an explicit loss framework to reconstruct the term by reducing the cosine distance between the embedding of the term and the embedding of the reconstructed term. We adapt this approach from Hill et al. (2015) who apply it to learn word representations using dictionaries. Inan et al. (2016) propose a loss framework for language modeling to minimize the distribution distance between the prediction distribution and the true data distribution. Though we use a different loss framework, we use a similar type of parameter tying in our implementation.

3 Definitions

Dictionary definitions represent a large source of our knowledge of meaning of words (Amsler, 1980). Definitions are composed of a 'genus' phrase and a 'differentia' phrase (Amsler, 1980). The 'genus' phrase identifies the general category of the defined word. This helps derive an 'Is A' relationship between the general category and the word being defined. The 'differentia' phrase distinguishes this instance of the general category from other instances. Definitions can have further set of differentia to imply more granular explanation. For example, Merriam-Webster [2] defines the word 'house' as 'a building that serves as living quarters for one or a few families'. Here the phrase 'a building' is the genus that denotes that a house is a building. The phrases 'that serves as living quarters' and 'for one or a few families' are differentia phrases which help identify house from other buildings. Our model of definitions adapts this interpretation. From a modeling perspective, we hypothesize that using language models would learn the template structure of definitions, and incorporating entity-entity co-occurrence as well as ontological category information would help us fill the specific differentia and genus concepts to the template structure.

4 Dataset

In SO, users can associate a question with a 'tag', such as 'Java' or 'machine learning', to help other users find and answer it. These tags are nearly always names of domain specific entities. Each tag

[2] https://www.merriam-webster.com/dictionary/house

Software Tag	Definition	Category
hmac	in cryptography hmac hash-based message authentication code is a specific construction for calculating a message authentication code mac involving a cryptographic hash-function in combination with a secret-key	authentication
persistence	persistence in computer programming refers to the capability of saving data outside the application memory .	database
ndjango	ndjango is a port of the popular django template-engine to .net .	framework
intellisense	intellisense is microsoft s implementation of automatic code-completion best known for its use in the microsoft visual-studio integrated development-environment .	compiler

Table 1: Sample definitions from the dataset

	train	val	test
# samples	22334	1240	1240
# avg definition length	16.54	16.53	16.69

Table 2: Dataset Statistics

has a definition on SO. For the definitions, we created a dataset of 25K software entities (tags from SO) and their definitions on SO. The data collection and pre-processing for the task is similar to cloze-style software questions collected in Dhingra et al. (2017). The definitions dataset was built from the definitional "excerpt" entry for each tag (entity) on Stack Overflow. For example, the excerpt for the "java" tag is, "Java is a general purpose object-oriented programming language designed to be used in conjunction with the Java Virtual Machine (JVM)." The dataset statistics can be seen in Table 2. This dataset is used for training our definition generation models. Examples of definitions in the dataset are shown in Table 1.

We use a background corpus of top 50 threads [3] tagged with every entity on Stack Overflow (Dhingra et al., 2017) and attempt to learn definitions of entities from this data. We use this background corpus for training word embeddings and to give us tag co-occurrences. In SO, a particular question can have multiple tags associated with it, which we call 'co-occurring tags'. We extracted the top 50 question posts for each tag, along with any answer-post responses and metadata (tags, authorship, comments) using Scrapy [4]. From each thread, we used all text that is not marked as code and segmented them into sentences. Each sentence is truncated to 2048 characters, lower-cased and tokenized using a custom tokenizer compatible with special characters in software terms (e.g. .net, c++). The background corpus for our task consists of 27 million sentences.

5 Model

5.1 Definition generation as language modeling

We model the task of generating definitions as a language modeling task. The architecture of the model is shown in Figure 2. We model the problem of generating a definition $D = w_1, w_2 ... w_T$ given the entity w^*, where the probability of generating a token $p(w_t)$ is given by $P(w_t|w_1..w_{t-1}, w^*)$ and the probability of generating the entire definition is given by:

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

We model this using LSTM language models (Mikolov et al., 2010; Hochreiter and Schmidhuber, 1997). An LSTM unit is composed of three multiplicative gates which control the proportions of information to forget and to pass on to the next time step. During training, the input to the LSTM are the word embeddings of the gold definition sequence. At test time, the input is the embedding of the input entity and previously generated words.

[3] A question along with the answers provided by other users is collectively called a thread. The threads are ranked in terms of votes from the community.

[4] https://scrapy.py

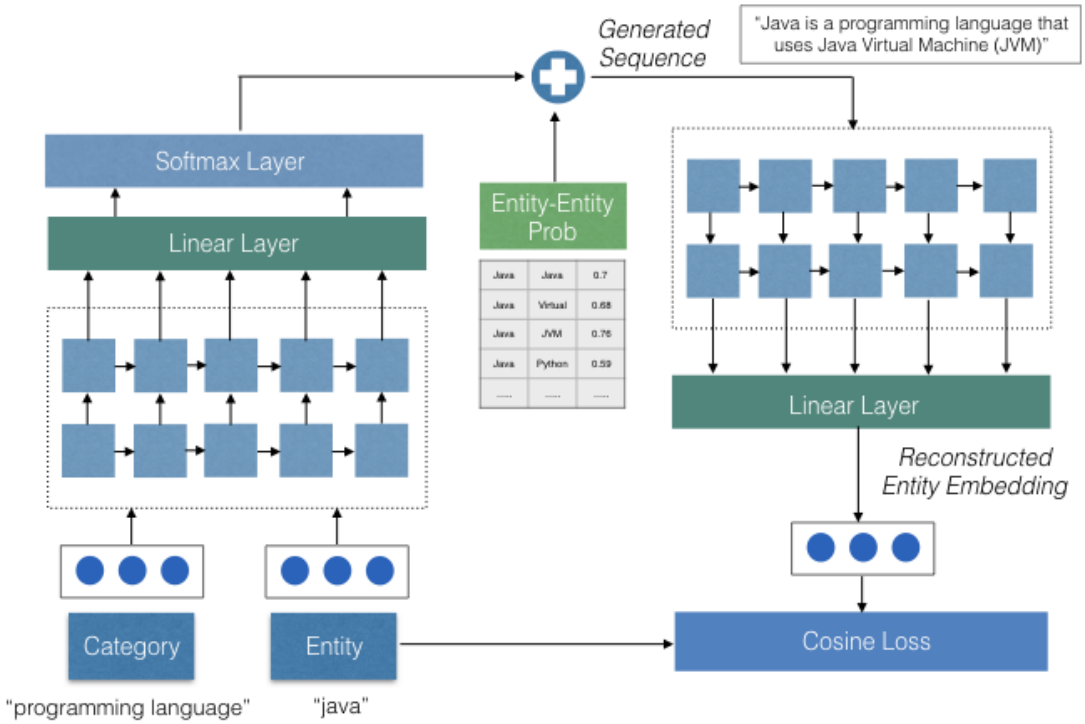

Figure 2: Model Architecture

We outline the functions in our model below :

$$x'_t = E[w_t]$$
$$h_t = LSTM(x'_t, h_{t-1})$$
$$P_{LM}(w_t|w_1...w_{t-1}, w^*) = sm(W_k * h_t) \quad (1)$$

where E is an embedding matrix, initialized with pre-trained embeddings and W_k is a weight matrix. sm is the softmax function.

Adapting the baselines from Noraset et al. (2017), we explore two variants of providing the model with the input entity :

Seed Model: The input entity is given as the first input token to the RNN as a seed. The loss of predicting the start token, $< sos >$, given the word is not taken into account.

Concat Model: Along with being given as a seed to the model, the input entity is concatenated with the input token of the RNN at every timestep. We use these as baselines for our approach as well.

5.2 Incorporating Entity-Entity Co-occurrence

We propose an extended model to incorporate entity-entity association information from the tags to generate the final definition. We define a co-occurrence based probability measure for every entity, w_e as :

$$P_{EE}(w_e|w^*) = \begin{cases} \frac{c(w_e,w^*)}{c(w^*)} + \epsilon, & \text{if } w_e \text{ is an entity} \\ \epsilon, & \text{otherwise} \end{cases}$$

where c is count function and w_e is defined as any software entity which is not the entity being defined. $c(w_e, w^*)$ is the count of sentences for which entities w_e and w^* were tagged together. This probability is smoothed for non-entity words with an ϵ value. We use $\epsilon = 0.0001$ for all cases. To incorporate this probability into our model, we interpolate it with the language model probability defined in Equation 1 as follows :

$$P(w_t|w_1..w_{t-1}, w^*) = \lambda_t * P_{EE}(w_t|w^*) + (1 - \lambda_t) * P_{LM}(w_t|w_1..w_{t-1}, w^*)$$

where λ_t is a learned parameter, given as follows :

$$z_t = tanh(W_p * h_{t-1} + W_q * E(w^*) + b)$$
$$\lambda_t = \sigma(W_r * z_t)$$

where W_p, W_q, W_r, b are learn-able parameters. The λ_t parameter learns to switch between the contextual language model probability when generating tokens forming the structure of the definition and the entity-entity probability when generating tokens which are themselves entities.

5.3 Modeling Ontological Category Information

We use a pre-defined set of 86 categories which are the ontological categories for the software domain proposed as part of the GNAT [5] project. For each category, we compute a distributed representation vector by taking the mean of every dimension of the constituent tokens in the category name. We term this the *average category embedding*. We map every entity to its closest category in embedding space using cosine distance. Examples of category mappings to the terms are shown in Table 1.

We explore two ways of using the average category embedding :
1) Adding the category embedding vector **(ACE)** to the word vector of the entity to extract a new vector which we hope is closer to words defining the entity. This is inspired from the idea of additive compositionality of vectors as shown in (Mikolov et al., 2013b).

$$x'_1 = E[w^*] + E'[c(w^*)]$$

2) Concatenating the category embedding vector **(CCE)** with the input embeddings at every timestep of the LSTM.

$$x'_t = E[w_t]; E'[c(w^*)]$$

where E is the word embedding matrix, E' is the category embedding matrix and $c(x)$ is a function that maps every entity to its corresponding ontological category.

5.4 Loss Augmentation

To enforce the model to condition on the entity and generate definitions, we propose to augment the standard cross entropy loss with a loss framework that focuses on reconstructing the entity from the generated sequence. This additionally constrains the model to generate tokens close to the tokens in the definition. We introduce a second LSTM model which reverse encodes the output text sequence of the forward mode and projects the encoded sequence into an embedding space of the same dimension as the term being defined.

$$h'_t = LSTM(y'_1....y'_T) \quad (2)$$
$$e^*_{w_r} = W_b * h'_t$$

where $y'_1...y'_T$ is the generated definition sequence and W_b is a weight matrix.

We add an additional objective to the model to minimize the cosine distance between the projected vector and the embedding of the input term:

$$loss_{cosine}(w^*, e^*_{w_r}) = 1 - \cos(E(w^*), e^*_{w_r})$$

where, w^* is the input term, and $e^*_{w_r}$ is the reconstructed term vector.

The resulting network can be trained end-to-end to minimize the cross entropy loss between the output and target sequence $L(y, y')$ in addition to the reconstruction loss between the input and reconstructed input vector $L(w^*, e^*_{w_r})$. Since the decode step is a greedy decode step, gradients cannot propagate through it. To solve this, we share the parameters between the two LSTM networks and forward and reconstruction linear layers (Chisholm et al., 2017). To generate definitions at test time, the backward network does not need to be evaluated.

6 Experimental Setup

To train the model, we use the 25k definitions dataset built as described in Section 4. We split the data randomly into 90:5:5 train, test and validation sets as shown in Table 2. The words being defined are mutually exclusive across the three sets, and thus our experiments evaluate how well the models generalize to new words.

All of the models utilize the same set of fixed word embeddings from two different corpora. The first set of vectors are trained entirely on the Stack Overflow background corpus and the other set are pre-trained open domain word embeddings[7]. Both these embeddings are concatenated and we use this as the representation for each word. For the embeddings trained on Stack Overflow corpus, we use the Word2Vec (Mikolov et al., 2013a) implementation of Gensim[8] toolkit. In the corpus, we prepend to every sentence in a question-answer pair, every tag it is associated with. We further eliminated stopwords from the corpus and set a larger context window size of 10.

For the model, we use a 2 layer LSTM with 500 hidden unit size for both the forward and reconstruction layers of the models. The size of the em-

[5] http://curtis.ml.cmu.edu/gnat/software/

[6] Our implementation of baselines from (Noraset et al., 2017) using greedy decode approach

[7] https://code.google.com/archive/p/word2vec/

[8] https://radimrehurek.com/gensim/models/word2vec.html

Model	BLEU
Seed + SO Emb*	8.90
Seed + SO Emb + Eng Emb**	9.01
Concat Model	9.44
Concat Model + Entity-Entity Model	9.28
Concat Model + Category Model (CCE)	10.19
Concat Model + Category Model (ACE)	**10.86**
Concat Model + Category Model (ACE) + Cosine Loss Model	**10.91**

Table 3: Experimental results for our different models. *SO Emb = Embeddings learned on Stack Overflow. **Eng Emb = Embeddings learned on general open domain English dataset

Model	BLEU
Seed (Noraset et al., 2017)	26.69
Concat (Noraset et al., 2017)	28.44
Seed (English words) [6]	32.79
Concat (English words)	36.37

Table 4: Experimental results of our baselines on common English words dataset (Noraset et al., 2017)

bedding layer is set to 300 dimension. For training, we minimize the cross-entropy and reconstruction loss using Adam (Kingma and Ba, 2014) optimizer with a learning rate of 0.001 and gradient clip of 5. We evaluate the task using BLEU (Papineni et al., 2002).

7 Results

The results for the different models are summarized in Table 4. The first section are results of the baselines as reported by Noraset et al. (2017). The second section shows results of our implementation of the baselines on common English word definitions dataset. We report BLEU scores on definitions generated using a greedy approach. The remaining two sections are results from our proposed models on software entity definitions dataset.

In comparison, on the software entity definitions dataset, the same baselines do not generate any reasonable definitions, giving low BLEU scores. This demonstrates that using a language-model with embeddings trained on general purpose large-scale, domain-specific corpora is inadequate when the definitions are longer and more domain-specific.

The Seed model, instantiated with both the Stack Overflow embeddings as well as the open domain Google News English embeddings, shows better performance than the model that only uses Stack Overflow embeddings. For all further models, we adopt the concatenated Stack Overflow embeddings and open domain English embeddings. We see from the table, that the Concat Model performs better than the Seed Model. We choose the best model among Seed and Concat and perform additional experiments to evaluate the additional proposed changes to the model.

Surprisingly, adding the entity-entity relationships to the Concat Model does not provide any gains. Although, providing the category information from the ontology by adding it to the input word vector (ACE) provides us a higher BLEU score of 10.86.

Further, adding the reconstruction loss objective to the ACE model provides us small additional gains and achieves a 10.91 BLEU score. Although we see incremental improvements on the task, overall our results show that language models are inadequate to model definitions, as empirically shown by the low overall BLEU scores.

8 Discussion

Noraset et al. (2017) showed that RNN language models can be used to learn to generate definitions for common English words. On adapting their techniques for closed domain software entities, we find that the language models generate definitions which follow the template of definitions, but have incorrect terms in the genus and differentia components.

Table 5 shows some reference definitions and definitions generated from our best model. In the generated definition of "virtual-pc", we observe that the model generates a definition which has a distinguishable genus and differentia, but the genus is not the right ontological category for the entity. The differentia is incorrect as well. Similarly in the definition of "esper", we observe that the model generates 'open source software' as the genus, while the reference genus is 'open source

Entity	Reference	Generated
windows-server-2000	microsoft windows-2000 server is an operating-system for use on server computers.	microsoft s UNK is a free open-source content-management-system ..
esper	UNK is a lightweight open-source library for cep complex-event-processing and esp event stream processing applications.	UNK is a open-source software for the java programming language.
virtual-pc	virtual-pc is a virtualization program that simulates a hardware environment using software.	the UNK is a commercial operating-system for the windows operating-system.

Table 5: Reference and generated sentences

library' which shows that the model is able to learn the right genus for the entity but generates an incorrect differentia component. We see that the reference differentia is quite long with many non-entity tokens which would be hard to model. This explains our results of obtaining lower BLEU scores using the entity-entity co-occurrence models as most differentia terms consist of many non-entity tokens. We also observed that the genus and differentia components for technical definitions have longer and very specific phrases compared to common English words. These phrases also tend to be very sparse in the vocabulary, making the task even more challenging.

The general English definitions dataset presented by Noraset et al. (2017) has 20K most common English words and their definitions. The English words for which the definitions are generated, also tend to appear in the corpus very frequently, thereby having better distributed representations. We presume that the higher BLEU scores for common English words are reflective of that. In contrast, entities in closed domains are much less frequent in background corpora increasing the difficulty of the task. Also, the average definition length in common English words definition corpus is 6.6 while the average length of definitions for software entities is 16.54, which adds additional complexity in generating these definitions. We hypothesize that due to low expressiveness of word representations of entities in comparisons to common English words, language models are unable to learn relations between entities and their genus and differentia components. The addition of the ontological category information alleviates the problem by a small margin, but is still insufficient for the model to learn to generate close-to-perfect definitions.

Through our observations, we find that RNN language models initialized with distributed word representations of entities is inadequate to generate definitions from scratch. We envision that future models should be able to learn better associations between entities and its genus and differentia phrases. Also, the model should ensure it has adequate long term memory to generate definitions that are longer in length.

9 Conclusion and Future Work

In this paper, we present our initial work in the task of definition generation for software entities or terms. We introduced different approaches for the task, where we explore ways of incorporating ontology information and entity-entity co-occurrence relationships. We also present the results and analysis for the same. Given the complexity of the task, we achieve around 2 BLEU improvements over baselines. We demonstrate that the current models are inadequate to automatically learn to generate complex definitions for entities.

As an immediate next step, we would like to approach the task from an encoder-decoder perspective by collecting external data about the word being defined and using it to guide the generation process. Our hypothesis is that providing external information about an entity and it's usage in various contexts, would help us better identify the genus and differentia for the entity. Currently, we give only the immediate parent category as an input from the ontology, we would also like to explore how to leverage on the entire ontology structure for definition generation.

Acknowledgments

This work was funded by NSF under grant CCF-1414030. Additionally, the authors would like to thank Kathryn Mazaitis for her help with data collection and analysis.

References

Robert Alfred Amsler. 1980. The structure of the merriam-webster pocket dictionary.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *arXiv preprint arXiv:1702.06235*.

Martin S Chodorow, Roy J Byrd, and George E Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 299–304. Association for Computational Linguistics.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

William Dolan, Lucy Vanderwende, and Stephen D Richardson. 1993. Automatically deriving structured knowledge bases from on-line dictionaries. In *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*, pages 5–14.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *arXiv preprint arXiv:1504.00548*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Smaranda Muresan and Judith Klavans. 2002. A method for automatically building and evaluating dictionary resources. In *LREC*, volume 2, pages 231–234.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *AAAI*, pages 3259–3266.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Lucas Batista Leite de Souza, Eduardo Cunha Campos, and Marcelo de Almeida Maia. 2014. Ranking crowd knowledge to assist software development. In *ICPC*.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec: Learning word embeddings using lexical dictionaries. In *Conference on*

Empirical Methods in Natural Language Processing (EMNLP 2017), pages 254–263.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.