INTENT DETECTION USING SEMANTICALLY ENRICHED WORD EMBEDDINGS

Joo-Kyung Kim[1*], *Gokhan Tur*[2†], *Asli Celikyilmaz*[3†], *Bin Cao*[4], *Ye-Yi Wang*[4]

[1]The Ohio State University, Columbus, Ohio, USA
[2]Google Research, Mountain View, California, USA
[3]Microsoft Research, Redmond, Washington, USA
[4]Microsoft, Bellevue, Washington, USA

ABSTRACT

State-of-the-art targeted language understanding systems rely on deep learning methods using 1-hot word vectors or off-the-shelf word embeddings. While word embeddings can be enriched with information from semantic lexicons (such as WordNet and PPDB) to improve their semantic representation, most previous research on word-embedding enriching has focused on improving intrinsic word-level tasks such as word analogy and antonym detection. In this work, we enrich word embeddings to force semantically similar or dissimilar words to be closer or farther away in the embedding space to improve the performance of an extrinsic task, namely, intent detection for spoken language understanding. We utilize several semantic lexicons, such as WordNet, PPDB, and Macmillan Dictionary to enrich the word embeddings and later use them as initial representation of words for intent detection. Thus, we enrich embeddings outside the neural network as opposed to learning the embeddings within the network, and, on top of the embeddings, build bidirectional LSTM for intent detection. Our experiments on ATIS and a real log dataset from Microsoft Cortana show that word embeddings enriched with semantic lexicons can improve intent detection.

Index Terms— word embeddings, semantic lexicons, LSTM, intent detection, spoken language understanding

1. INTRODUCTION

Word embeddings represent words as real-valued vectors and they have been widely used as the inputs to neural network based models for NLP tasks. Word embedding models such as word2vec (skip-gram and continuous bag-of-words (CBOW)) [1, 2], and GloVe [3] generate word vectors based on the distributional hypothesis [4], which assumes that the meaning of each word can be represented by the context of the word. However, word embeddings trained only with the neighboring contexts may place words improperly in vector spaces. For example, even though antonyms are semantically opposite, because their contexts are similar in many cases, antonym vectors can be quite close in vector spaces.

To deal with this issue, semantic lexicons such as WordNet [5] and the Paraphrase Database (PPDB) [6, 7], which contain semantic relations among words, have been used to enrich word embeddings. In [8], each word vector is adjusted to be in the middle between its initial vector and the average of its synonymous words. In [9], each word vector is adjusted with a max-margin approach letting synonyms be more similar and antonyms be more dissimilar while maintaining the similarities among initial neighboring words. In [10, 11, 12, 13], skip-gram is jointly trained to incorporate word relation information from semantic lexicons. For more fine-grained semantic representations, semantic intensity scales can be utilized to enrich word embeddings [14]. Hypernym and hyponym relations can also be used to enrich word embeddings regarding semantic hierarchies [12, 15].

However, these enrichment approaches showed improved performance only on word-level tasks such as word similarity [8, 9, 11], antonym detection [10, 11, 16, 13], semantic hierarchies [12, 15], and semantic scale inference [14]. More critical language understanding tasks such as intent detection and slot filling in spoken language understanding [17] require dealing with phrases and sentences.

Recently, gated recurrent neural network models such as Long Short-term Memory (LSTM) [18] and Gated Recurrent Unit (GRU) [19] have been widely used for sentence-level NLP tasks since they can utilize long temporal dependencies, which are important in dealing with long sentences. Specifically, models for intent detection and slot filling have been jointly or independently trained with GRU, LSTM [20, 21], bidirectional LSTM [22], or bidirectional LSTM with attention mechanism in [23]. Those models showed good performances without exploiting external semantic resources. By utilizing semantically enriched word embeddings, we can further improve the models for the spoken language understanding tasks.

In this work, we extend the word embedding enrichment in [9] by incorporating the enrichment mechanism via related words from Macmillan Dictionary, and build a bidirectional

[*]Work done during the author's internship at Microsoft.
[†]Work done at Microsoft.

LSTM on top of the enriched word embeddings. By evaluating on a publicly available language understanding benchmark corpus, ATIS, and a real log dataset about places from Microsoft Cortana, we show that word embeddings enriched with semantic lexicons can improve the performance of intent detection. Also, we show that fixing the enriched word embeddings during the training of the model is more effective than allowing updating of the word embeddings when the training sets are small.

2. ENRICHING WORD EMBEDDINGS

We use 200 dimensional GloVe word vectors, which showed competitive performance in various NLP tasks [24], as the baseline word vectors, and enrich them with the enrichment method used in [9]. The method adjusts word vectors to make 1) synonym word vectors to be more similar, 2) antonym word vectors farther apart while 3) keeping the adjusted word vectors' similarities to their initial neighboring word vectors. These three criteria are formulated as three max-margin objective functions whose linear combination is minimized with stochastic gradient descent. While this approach deals with only synonyms and antonyms, we added a new objective function making related words in Macmillan dictionary to be more similar. Those max-margin objective functions are formulated as follows.

2.1. Enriching with antonyms

Word vectors are adjusted so that the cosine similarity between a word and each of its antonyms will be zero or lower:

$$AF\left(V\right)=\sum_{(u,w)\in A}\tau\left(\cos\left(v_u,v_w\right)\right), \tag{1}$$

where $\tau\left(x\right)=\max\left(0,x\right)$, V is the vocabulary matrix, A is the set of antonym pairs, and v_i is the i-th row of V (i-th word vector). The antonym pairs consist of the antonyms from WordNet and *Exclusion* relations from PPDB.

2.2. Enriching with synonyms

The cosine similarity between a word and each of its synonyms is increased:

$$SC\left(V\right)=\sum_{(u,w)\in S}\tau\left(\delta-\cos\left(v_u,v_w\right)\right), \tag{2}$$

where S is the set of synonym pairs and δ is 1. The synonym pairs consist of the *Equivalence* relations from PPDB. Since δ is 1, synonym vectors are encouraged to be adjusted to be maximally similar.

2.3. Regularizing by keeping the similarity to the initial neighboring words

If we adjust word vectors too much, we may loose information about their distributional semantics coming from the initial word vectors. This may degrade the quality of word vectors. To deal with this issue, we added a regularization term to the objective function by keeping the cosine similarity between the initial vectors of a word and each of its neighboring words higher than or equal to the current cosine similarity between them as follows:

$$KN\left(V,V^0\right)=\sum_{i=1}^{N}\sum_{j\in N(i)}\tau\left(\cos\left(v_i,v_j\right)-\cos\left(v_i^0,v_j^0\right)\right), \tag{3}$$

where V^0 is the initial vocabulary matrix, N is the vocabulary size, and $N\left(i\right)$ is the set of the initial neighbors of the i-th word. For each word, other words with cosine similarities higher than or equal to 0.8 are regarded as its neighbors in the formulation.
The objective function for the word vector enrichment is represented as the sum of the following three terms:

$$C\left(V,V^0\right)=AF\left(V\right)+SC\left(V\right)+KN\left(V,V^0\right) \tag{4}$$

This function is minimized by stochastic gradient descent with learning rate 0.1 for maximum of 20 epochs.

2.4. Enriching with clusters of related words

The objective functions described so far are from [9] and they only utilize synonyms and antonyms to enrich word vectors. If we can also use information of related words, we can further improve the word embeddings. For example, "commute" and "ride" are not synonyms in WordNet but they are related as belonging to a same category of "traveling in a vehicle." In this case, even though "commute" and "ride" are not synonyms, it is helpful for representing word semantics to make the corresponding word vectors sufficiently similar. Therefore, we also adjust word vectors so that the cosine similarity between those related words from Macmillan Dictionary[1] is higher than or equal to 0 using:

$$RC\left(V\right)=\sum_{(u,w)\in R}\tau\left(\delta-\cos\left(v_u,v_w\right)\right), \tag{5}$$

where R is the set of related words and δ is 0. Different from Equation (2), which maximizes the cosine similarity between synonyms, we set δ to 0 so that related word vectors whose cosine similarity is already higher than or equal to 0 are not adjusted. This helps prevent related words from being more similar than synonyms in embedding spaces. Since verbs and

[1] Available from `http://www.macmillandictionary.com/us`

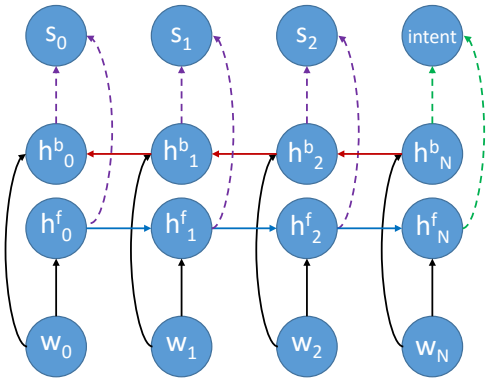

Fig. 1. The bidirectional LSTM architecture used in this work. w_i, h_i^f, h_i^b, and s_i denote the word embedding layer, the forward hidden unit layer, the backward hidden unit layer, and the slot output layer at i-th time step. $intent$ is the intent output layer of the current sentence.

nouns are important content words in intent detection, we tried both cases of using related verbs and nouns and using all the related words.

3. INTENT DETECTION

Intent detection is a query/utterance classification task that can be formulated as:

$$y' = \arg\max_y p\left(y|w_1, ..., w_n\right), \tag{6}$$

where w_i is an i-th word of a sentence and y is the intent.

Using the enriched word embeddings as the initial word representations, we build bidirectional LSTM, where we train an end-to-end model that jointly learns the slot and intent classes from the training data. Figure 1 shows the architecture of the model.

In Figure 1, w_i denotes the word embedding output of the i-th word in the current sentence. w_0 and w_N denote the word embedding outputs of the beginning (BOS) and the ending (EOS) of the sentence, respectively. h_i^f and h_i^b denote the LSTM hidden layer outputs in forward direction and backward direction, respectively. s_i denotes the slot output and $intent$ is the intent output of the sentence. For the regularization, we normalize word vectors after each update and we insert a dropout layer [25] with drop rate 0.5 between hidden layers and the output layers.

In [22], it was shown that the joint training of both intent detection and slot filling can perform better than training only for intent detection. Therefore, similar to the joint model architectures in [22], we jointly train intent detection and slot filling, where the nodes of 0 to $(N-1)$-th time steps are used for slot filling, and the nodes of N-th (EOS) step are used for the intent detection. However, different from the models in [22], we have dedicated connections from hidden nodes to the slot filling and other connections from hidden nodes to the intent detection since intent labels are never targeted in slot filling and vice versa. Also, while only forward hidden nodes are used for intent detection in [22], we detect intents using both forward and backward hidden nodes.

Since we jointly train both intent detection and slot filling, we utilize both intent errors and slot filling errors for updating the model. Because there can be more than one slot while there is only one intent in a sentence, as a gradient normalization, we divide the gradients from slot filling errors by the number of slots in the current sentence. Then, since the gradients from slot filling errors are shrinked, the model is trained with more focus on correcting the intent detection errors.

We try both cases of fixing and updating the word embedding layer during the training to see if the fixing can show better performance depending on the size of the training sets.

	ATIS	Places
Words in the vocabulary	637	13,640
Intent labels	22	35
Train set sentences	4,978	10,000
Dev set sentences	-	10,000
Test set sentences	893	10,000

Table 1. The vocabulary sizes, the numbers of the intent labels, the train set sizes, the development set sizes, and the test set sizes of the evaluation datasets.

	ATIS	Places
WordNet antonyms	53	530
PPDB antonyms	9	137
PPDB synonyms	67	1,268
Related verbs & nouns from Macmillan	269	5,389
All related words from Macmillan	458	6,417

Table 2. The numbers of adjusted words by different semantic lexicons.

4. EVALUATION

We show the accuracies of intent detection given different methods of word embedding enrichment and different training set sizes for both cases of word embedding fixing or updating during the training.

4.1. Evaluation datasets

We evaluate intent detection on ATIS and Places from Microsoft Cortana. Table 1 shows the description of the two datasets. We change words occurring only once in the training set to <unk>, and we map words in the test set but not existing in the training set to <unk>.

ATIS is one of the most widely used datasets for the evaluation of spoken language understanding tasks. In ATIS, for

ATIS Intent Detection Accuracy (%)		Train set coverage				
		10%	20%	40%	80%	100%
word embeddings fixed	1-hot ($d = 637$)	84.77	89.03	92.05	93.73	94.62
	GloVe	83.99	89.14	91.60	95.07	94.85
	GloVe+VerbCls n-hot ($d = 661$)	84.88	90.03	92.39	94.62	94.85
	GloVe adjusted w/ syn & ant	88.69	91.16	93.51	**95.97**	95.97
	GloVe adjusted w/ related verbs&nouns	88.69	91.04	94.06	95.52	95.86
	GloVe adjusted w/ syn & ant & related verbs&nouns	88.80	91.38	**94.51**	95.41	96.30
	GloVe adjusted w/ all related words	**88.91**	**91.60**	93.28	95.74	**96.53**
	GloVe adjusted w/ syn & ant & all related words	86.90	91.49	94.06	95.30	**96.53**
word embeddings updated	1-hot ($d = 637$)	**88.02**	90.26	93.62	95.41	94.62
	GloVe	87.23	**91.71**	92.72	94.29	95.63
	GloVe+VerbCls n-hot ($d = 661$)	87.68	90.48	94.29	94.85	95.86
	GloVe adjusted w/ syn & ant	84.66	90.37	94.18	95.86	96.30
	GloVe adjusted w/ related verbs&nouns	85.55	89.36	92.72	**96.08**	96.53
	GloVe adjusted w/ syn & ant & related verbs&nouns	85.55	90.15	**94.85**	94.96	95.86
	GloVe adjusted w/ all related words	86.56	89.03	94.74	95.07	95.97
	GloVe adjusted w/ syn & ant & all related words	87.46	90.03	93.73	95.52	**97.31**

Table 3. Intent detection accuracies on ATIS.

example, the intents of "I need a flight tomorrow from Columbus to Minneapolis," "what is the seating capacity of a Boeing 767", and "show me the ground transportation in Denver" are "flight," "capacity," and "ground_service," respectively.

In addition, we also evaluated intent detection performance on a real log dataset about places from Microsoft Cortana. Some examples of the dataset are provided below:

i want some hot wings tonight. (find-place)
where is the nearest planet fitness ? (find-place)
display map of ocala drive (get-route)
how far is my home to here ? (get-distance)
skype call now to mcdonalds (make-call)

The focus here is on audiovisual media in the places domain. The user interacts via voice with a system (in our case it was the Windows Phone) that can perform a variety of tasks such as browsing and searching. We used crowd-sourcing to collect and annotate queries with semantic entities. The dataset contains several thousand training and evaluation queries. We sampled 10,000 sentences from the places logs to compile our training, development, and test data. (see details in Table 1).

4.2. Semantic lexicons for enriching word embeddings

As described in Section 2, we enrich word embeddings with WordNet antonyms, PPDB antonyms, PPDB synonyms, and related words from Macmillan Dictionary. Table 2 shows how many words in the vocabulary were adjusted by specific semantic lexicons for each training set.

4.3. Experiment results

On top of the enriched word embeddings, we build a bidirectional LSTM model for the joint intent detection and slot filling. We optimize the model with ADAM[2] [26]. Given the enriched word embeddings and the training set, we train the model for 100 epochs and choose the parameters showing the best intent detection accuracy on the development set and evaluate it on the test set. Since there is no development set in ATIS, we randomly sampled 20% of the training set to use as the development set. In the cases that 100% of ATIS training set is used, we fix the number of epochs to be the epoch showing the best intent accuracy for the development set when the model is trained with 80% of the training set.

Tables 3 and 4 show the intent detection accuracies on different proportions of the training set and different adjustment methods for the word embeddings. In these tables, "1-hot" denotes the case representing each word as the vectors whose dimensionality is the vocabulary size. In the vectors, the element corresponding to the word's index in the vocabulary is set to 1 and all other elements are set to 0. "GloVe" denotes that the initial GloVe vectors are used without any adjustments. We used off-the-shelf GloVe vectors[3] that are trained with 2014 Wikipedia dump and English Gigaword Fifth Edition for ATIS, but we trained GloVe vectors with the entire training set for Places since many of the words in Places are missed in the off-the-shelf GloVe vectors. "GloVe+VerbCls n-hot" denotes that each GloVe vector is concatenated with an n-hot vector representing the classes that the word is belonging to in Macmillan Dictionary. For example, "arrive" belongs to the classes of "General Words Meaning to Happen," "To Succeed in Doing Something," "Pregnancy and Having a Baby," and so on. These belongings can be represented by setting 1 to the corresponding elements in a vector representing

[2]learning rate=0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e - 8$.

[3]Available from `http://nlp.stanford.edu/projects/glove/`

Places Intent Detection Accuracy (%)		Train set coverage				
		10%	20%	40%	80%	100%
word embeddings fixed	GloVe	88.64	90.78	92.70	93.57	94.03
	GloVe adjusted w/ syn & ant	88.56	90.42	92.54	93.63	93.87
	GloVe adjusted w/ related verbs & nouns	88.51	90.61	92.49	93.62	94.23
	GloVe adjusted w/ syn & ant & related verbs&nouns	**88.70**	**91.38**	92.84	**94.00**	**94.36**
	GloVe adjusted w/ all related words	88.47	91.18	**93.03**	93.95	94.17
	GloVe adjusted w/ syn & ant & all related words	88.68	90.53	92.74	93.84	94.15
word embeddings updated	GloVe	86.44	90.71	92.56	93.88	94.08
	GloVe adjusted w/ syn & ant	86.50	**91.08**	92.34	94.03	**94.44**
	GloVe adjusted w/ related verbs & nouns	86.09	90.30	92.49	93.92	94.27
	GloVe adjusted w/ syn & ant & related verbs&nouns	86.58	90.70	92.39	93.83	94.20
	GloVe adjusted w/ all related words	**87.23**	90.59	92.47	94.06	94.31
	GloVe adjusted w/ syn & ant & all related words	86.53	89.97	**92.69**	**94.10**	94.18

Table 4. Intent detection accuracies on Places.

the classes while setting 0 to other elements. By attaching this vector to the GloVe vector, we can use the information of verb relations without adjusting GloVe vectors. However, they did not show better performance than the adjusted word vectors for ATIS. "Syn & ant" denotes that semantic lexicons (WordNet antonyms, PPDB synonyms, and PPDB antonyms) that were used in [9] are used for the adjustment, "related verb & nouns" denotes that the related verbs and nouns from Macmillan Dictionary are used, and "syn & ant & related verbs & nouns" denotes that all the synonyms, antonyms, related verbs and nouns are used for the adjustment.

Table 3 shows the experiment results on ATIS. For this dataset, in the cases just using initial GloVe without enrichment, fixing the word embeddings during the training is worse than updating the word embeddings. However, fixing the word embeddings enriched with semantic lexicons show better performance than updating GloVe during the training in most cases. In many of the cases where enriched word embeddings are updated during the training, the performances of enriched word embeddings are worse than just using GloVe vectors if 20% or smaller proportions of the training set are used though the enrichments show better performances if 40% or larger proportions of the training set are used. This is mainly because a certain subset of word embeddings can be substantially modified during the model training.

Table 4 shows the results on Places. For this dataset, fixing word embeddings that are enriched with synonyms, antonyms, and related verbs and nouns also showed better performances than using GloVe vectors for all the training set proportions regardless of fixing or updating during the training. However, for some enrichment methods, updating enriched GloVe vectors during the training showed better performance when using larger proportions of the training set. Since the vocabulary size of Places is large, we did not evaluate 1-hot and GloVe+VerbCls n-hot cases on Places.

In sum, for both ATIS and Places, using fixed word embeddings that are enriched with semantic lexicons showed better performance than using the original GloVe vectors in most cases. Specifically, if smaller proportions of the training set are used, fixing the enriched word embeddings was better than updating the enriched word embeddings in most cases.

5. CONCLUSION AND FUTURE WORK

In this work, we showed that enriching word embeddings with semantic lexicons can be helpful not only for word-level tasks but also intent detection, which is a sentence-level downstream task. Evaluating on two datasets, ATIS and Places from Microsoft Cortana, using the fixed enriched word embeddings as the input layer of the bidirectional LSTM models showed better performances of intent detection than using the original GloVe vectors. Fixing the enriched word embeddings was more effective when the training set is small.

Following [9]'s work, we've tried semantic lexicons and related words from Macmillan Dictionary to enrich the word embeddings. For places domain task, using synonyms, antonyms, and the related verbs and nouns have show good performance when the word embeddings are fixed. For ATIS, using all the related words has yielded good performance in several, but not all, cases.

As future work, we can first try more complex models such as stacked bidirectional LSTM [27] and LSTM with Memory Networks [28, 29]. Since having only small training sets is a more critical issue for such more complex models, providing well enriched word embeddings can be more helpful. We can also try different lexicons such as hypernym/hyponym relations and relational database entries for adjusting word embeddings.

6. REFERENCES

[1] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient Estimation of Word Representations in Vector

Space," in *ICLR workshop*, 2013.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *NIPS*, 2013, pp. 3111–3119.

[3] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *EMNLP*, 2014, pp. 1532–1543.

[4] Z. Harris, "Distributional structure," *Word*, vol. 10, pp. 146–162, 1954.

[5] C. Fellbaum, *WordNet: An electronic lexical database*, MIT Press, 1998.

[6] J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "PPDB: The Paraphrase Database," in *NAACL-HLT*, 2013, pp. 758–764.

[7] E. Pavlick, P. Rastogi, J. Ganitkevich, B. Van Durme, and C. Callison-Burch, "PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification," in *ACL*, 2015, pp. 425–430.

[8] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting Word Vectors to Semantic Lexicons," in *NAACL*, 2015, pp. 1606–1615.

[9] N. Mrki, D. O. Saghdha, B. Thomson, M. Gai, L. Rojas-Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, "Counter-fitting Word Vectors to Linguistic Constraints," in *NAACL*, 2016, pp. 142–148.

[10] M. Ono, M. Miwa, and Y. Sasaki, "Word Embedding-based Antonym Detection using Thesauri and Distributional Information," in *NAACL*, 2015, pp. 984–989.

[11] N. T. Pham, A. Lazaridou, and M. Baroni, "A multitask objective to inject lexical contrast into distributional semantics," in *ACL*, 2015, pp. 21–26.

[12] Q. Liu, H. Jiang, S. Wei, Z.-H. Ling, and Y. Hu, "Learning semantic word embeddings based on ordinal knowledge constraints," in *ACL*, 2015, pp. 1501–1511.

[13] K. A. Nguyen, S. S. im Walde, and N. T. Vu, "Integrating Distributional Lexical Contrast into Word Embeddings for Antonym-Synonym Distinction," in *ACL*, 2016, pp. 454–459.

[14] J.-K. Kim, M.-C. de Marneffe, and E. Fosler-Lussier, "Adjusting Word Embeddings with Semantic Intensity Orders," in *ACL 2016 workshop on Representation Learning for NLP (RepL4NLP)*, 2016, pp. 62–69.

[15] R. Fu, J. Guo, B. Qin, W. Che, H. Wang, and T. Liu, "Learning semantic hierarchies via word embeddings," in *ACL*, 2014, pp. 1199–1209.

[16] Z. Chen, W. Lin, Q. Chen, X. Chen, S. Wei, H. Jiang, and X. Zhu, "Revisiting Word Embedding for Contrasting Meaning," in *ACL*, 2015, pp. 106–115.

[17] G. Tur and R. de Mori, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, New York, NY: John Wiley and Sons, 2011.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] K. Cho, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *EMNLP*, 2014, pp. 1724–1734.

[20] S. Ravuri and A. Stolcke, "A Comparative Study of Neural Network Models for Lexical Intent Classification," in *ASRU*, 2015, pp. 368–374.

[21] B. Liu and I. Lane, "Attention-based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling," in *Interspeech*, 2016.

[22] D. Hakkani-Tür, G. Tur, A. Celikyilmaz, Y.-N. Chen, J. Gao, L. Deng, and Y.-Y. Wang, "Multi-Domain Joint Semantic Frame Parsing using Bi-directional RNN-LSTM," in *Interspeech*, 2016.

[23] X. Zhang and H. Wang, "A Joint Model of Intent Determination and Slot Filling for Spoken Language Understanding," in *IJCAI*, 2016, pp. 2993–2999.

[24] S. Ghannay, B. Favre, Y. Estève, and N. Camelin, "Word Embeddings Evaluation and Combination," in *LREC*, 2016, pp. 300–305.

[25] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves Recurrent Neural Networks for Handwriting Recognition," in *ICFHR*, 2014, pp. 285–290.

[26] D. P. Kingma and J. L. Ba, "ADAM: A Method for Stochastic Optimization," in *ICLR*, 2015.

[27] A. Graves, N. Jaitly, and A. rahman Mohamed, "Hybrid Speech Recognition with Deep Bidirectional LSTM," in *ASRU*, 2013, pp. 273–278.

[28] J. Weston, S. Chopra, and A. Bordes, "Memory Networks," in *ICLR*, 2015.

[29] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End–To–End Memory Networks," in *NIPS*, 2015.