Linguistic Knowledge Transfer for Enriching Vector Representations

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the

Graduate School of The Ohio State University

By

Joo-Kyung Kim, B.E., M.S.

Graduate Program in Computer Science & Engineering

The Ohio State University

2017

Dissertation Committee:

Prof. Eric Fosler-Lussier, Advisor

Prof. Alan Ritter

Prof. Michael White

ABSTRACT

Many state-of-the-art neural network models utilize a huge number of parameters, where a large number of labeled training examples are necessary for sufficient training of the models. Those models may not be properly trained if there are not enough training examples for target tasks. This dissertation focuses on transfer learning methods, which improve the performance of the target tasks in such situations by leveraging external resources or models from other tasks. Specifically, we introduce transfer learning methods for enriching word or sentence vector representations of neural network models by transferring linguistic knowledge.

Usually, the first layer of the neural networks for Natural Language Processing (NLP) is a word embedding layer. Word embeddings represent each word as a real-valued vector, where semantically or syntactically similar words tend to have similar vector representations in vector spaces. The first part of this dissertation is mainly about word embedding enrichment, which is categorized as an inductive transfer learning methodology. We show that word embeddings can represent semantic intensity scales like "good" < "great" < "excellent" on vector spaces, and semantic intensity orders of words can be used as the knowledge sources to adjust word vector positions to improve the semantics of words by evaluating on word-level semantics tasks. Also, we show that word embeddings that are enriched with linguistic knowledge can be used to improve the performance of the Bidirectional

Long Short-Term Memory (BLSTM) model for intent detection, which is a sentence-level downstream task especially when only small numbers of training examples are available.

The second part of this dissertation concerns about sentence-level transfer learning for sequence tagging tasks. We introduce a cross-domain transfer learning model for dialog slot-filling, which is an inductive transfer learning method, and a cross-lingual transfer learning model for Part-of-Speech (POS) tagging, which is a transductive transfer learning method. Both models utilize a common BLSTM that enables knowledge transfer from other domains/languages, and private BLSTMs for domain/language-specific representations. We also use adversarial training and other auxiliary objectives such as representation separations and bidirectional language models to further improve the transfer learning performance. We show that those sentence-level transfer learning models improve sequence tagging performances without exploiting any other cross-domain or cross-lingual knowledge.

To my family

ACKNOWLEDGMENTS

Completing a PhD would mean being an independent researcher, but no one would be able to do it alone without help of many people.

First and foremost, I would like to express my sincere gratitude to my advisor, Eric Fosler-Lussier. He incredibly helped me develop my crude initial ideas into sensible ones and guided me focus on meaningful research rather than work on toy problems while providing me great research opportunities and encouraging me find my own research topics. I will always be thankful to have worked with and learnt from him.

I would like to thank Marie-Catherine de Marneffe. Nearly as a co-advisor, she helped me initiate the work in the early chapters of this dissertation and safely land at the NLP research area. I am very grateful for her help with deep linguistic insight. I am also thankful to have Alan Ritter and Michael White as my dissertation committee members. They gave me insightful feedback on many of my work during my PhD study.

I would also like to thank the former and the current lab mates of the SLaTe lab – Rohit Prabhavalkar, Preethi Jyothi, Preethi Raghavan, Yanzhang (Ryan) He, Yi Ma, Chaitanya Shivade, Andy Plummer, Sirui Xi, Deblin Bagchi, Denis Newman-Griffis, Adam Stiff, Prashant Serai, and Manirupa Das. They were great mentors, colleagues, and friends.

I also thank my family in South Korea who always supported me so that I can focus on my PhD work for such a long time. Last, but not least, I cannot thank my wife, Irang enough for her devotion, encouragement, and support to me.

VITA

2005	B.E., Computer Science Sogang University, Seoul, South Korea
2008	M.S., Computer Science and Engineering Seoul National University, Seoul, South Korea

PUBLICATIONS

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier, "Cross-Lingual Transfer Learning for POS Tagging without Cross-Lingual Resources," *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang, "Intent Detection using Semantically Enriched Word Embeddings," *IEEE Workshop on Spoken Language Technology (SLT)*, pp. 414-419, 2016.

Joo-Kyung Kim, Marie-Catherine de Marneffe, and Eric Fosler-Lussier, "Adjusting Word Embeddings with Semantic Intensity Orders," *ACL 2016 Workshop on Representation Learning for NLP (RepL4NLP)*, pp. 62-69, 2016.

Joo-Kyung Kim, Marie-Catherine de Marneffe, and Eric Fosler-Lussier, "Neural word embeddings with multiplicative feature interactions for tensor-based compositions," *NAACL 2015 Workshop on Vector Space Modeling for NLP (VSM)*, pp. 143-150, 2015.

Joo-Kyung Kim and Marie-Catherine de Marneffe, "Deriving adjectival scales from continuous space word representations," *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1625-1630, 2013.

Joo-Kyung Kim and Byoung-Tak Zhang, "Evolving hypernetworks for pattern classification," *IEEE Congress on Evolutionary Computation (CEC)*, pp. 1856-1862, 2007.

Joo-Kyung Kim, Byung Soo Kim, Oh Hyuk Kwon, Seung Kon Hwang, Jung-Woo Ha, Chan-Hoon Park, Duck Jin Chung, Chong Ho Lee, Jaehyun Park, and Byoung-Tak Zhang, "A DNA computing-inspired silicon chip for pattern recognition," *13th International Meeting on DNA Computing (DNA)*, 2007.

Byoung-Tak Zhang and **Joo-Kyung Kim**, "DNA hypernetworks for information storage and retrieval," *12th International Meeting on DNA Computing (DNA)*, pp. 298-307, 2006.

FIELDS OF STUDY

Major Field: Computer Science and Engineering

Studies in Artificial Intelligence: Prof. Eric Fosler-Lussier

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Neural network models transform input patterns to vector forms with zero to multiple layers of nonlinear transformations. Thanks to the expressivity of the models, neural network models are showing the state-of-the-art performance for many different Natural Language Processing (NLP) tasks such as Part-of-Speech (POS) Tagging (Plank et al., 2016), Sentiment Analysis (Kokkinos and Potamianos, 2017), Textual Entailment (Wang et al., 2017), and Machine Translation (Gehring et al., 2017; Vaswani et al., 2017). However, since those models usually consist of a large number of parameters, the model cannot be sufficiently trained if there are only a small number of training examples. One way to address the data insufficiency issue is using transfer learning methods, which can help improve the performance of the target task by utilizing the models or datasets for other tasks. Pan and Yang (2010) defined transfer learning as follows:

Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$ or $T_S \neq T_T$.[1]

In addition, Pan and Yang (2010) categorized three transfer learning cases as Table 1.1.

In neural network models, sharing the input layers for multiple tasks is an inductive transfer learning case, and sharing the output layers is a transductive transfer learning case. Neural network models for NLP usually consist of word and/or character embedding layers

[1]In neural network models, we can regard domains as the input spaces and tasks as the output spaces.

	Source and Target Domains	Source and Target Tasks
Inductive Transfer Learning	The same	Different but related
Transductive Transfer Learning	Different but related	The same
Unsupervised Transfer Learning	Different but related	Different but related

Table 1.1: Three transfer learning categories (Pan and Yang, 2010).

as the input layers, zero to multiple intermediate layers, and output layers for the final model outputs. Therefore, using pretrained word embeddings, which are widely adopted in many NLP models (Turian et al., 2010; Collobert et al., 2011), is an inductive transfer learning since the word embeddings are pretrained using different corpora with different training objectives but on the same input space.

We can also classify transfer learning methods for NLP as either resource-based or model-based (Yang et al., 2017). Resource-based transfer learning transfers knowledge from external linguistic annotations such as dictionaries, thesauruses, or word alignments. In contrast, model-based transfer learning utilizes models trained for the source tasks to improve the performance of the target models.

We first address word-level semantic representations in vector spaces. We demonstrate the representations of semantic intensity scales in word embedding spaces and word embedding enrichment with semantic intensity orders as well as using thesauruses, which improve several word-level NLP tasks related to semantic intensities.

Also, we show that word embeddings enriched with thesauruses can be utilized to improve the performance of Bidirectional Long Short-Term Memory (BLSTM)–models (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) for intent detection on ATIS (He and Young, 2003) and a real log dataset especially when only small numbers of training examples are given.

We then move to a sentence-level inductive transfer learning model: a cross-domain transfer learning model for slot-filling. Our model has a common BLSTM that enables knowledge transfer from other domains, and private BLSTMs for domain-specific representations. The model is not only trained with the slot-filling prediction but also with domain-adversarial training as an auxiliary objective. We show that our model shows better transfer learning performance especially when we are given two source domain datasets.

As an opposite view, sharing the output layers with different word embedding spaces is a transductive transfer learning. As a transductive transfer learning case,[2] we address cross-lingual transfer learning for POS tagging, where the overall model architecture is similar to that of cross-domain transfer learning, but the output labels are universal POS tags and the input word embeddings are different for different languages. In the inductive transfer learning cases, it is relatively easier to obtain and utilize linguistic resources such as thesauruses and semantic intensity orders, but it is hard to collect cross-lingual resources over multiple languages. Therefore, we focus on the case where we do not utilize any cross-lingual knowledge or resources. We show that our model improves the POS tagging performance on Universal Dependencies corpora (Nivre et al., 2016) without exploiting any cross-lingual knowledge or resources.

In addition, as an appendix, we show an extension of the Continuous Bag-of-Words (CBOW) model reflecting the phrase composition of models for downstream tasks to the word embedding training.

[2]Since the character embeddings is shared among all the languages, our approach is also a partially inductive transfer learning method.

1.1 Contributions

The contribution for each chapter can be summarized as follows.

1.1.1 Representations of semantic intensity scales in vector spaces (Chapter 3)

Word vector representations extracted from neural network language models have been shown that these representations do capture syntactic and semantic regularities (Mikolov et al., 2013c). Here, we push the interpretation of word vector representations further by demonstrating that vector offsets can be used to derive adjectival scales (e.g., *okay* $<$ *good* $<$ *excellent*). We evaluate the scales on the indirect answers to *yes/no* questions corpus de Marneffe et al. (2010). Our approach outperforms previous results on this corpus and highlights the quality of the scales extracted, providing further support that the word vector representations are meaningful.

1.1.2 Enriching word embeddings with semantic intensity orders (Chapter 4)

Semantic lexicons such as WordNet (Fellbaum, 1998) and Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) have been used to improve the vector-based semantic representations of words by adjusting the word vectors. However, such lexicons lack semantic intensity information, inhibiting adjustment of vector spaces to better represent semantic intensity scales. We enrich word vectors using the semantic intensity information in addition to synonyms and antonyms from WordNet and PPDB, and show improved performance on judging semantic intensity orders of adjective pairs on three different human annotated datasets. The enrichment can be considered as knowledge transfer from semantic lexicons to word vector representations.

1.1.3 Enriching word embeddings for intent detection (Chapter 5)

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. 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 dataset of real user transactions from Microsoft Cortana show that word embeddings enriched with semantic lexicons can improve intent detection. The enrichment here is also regarded as knowledge transfer from semantic lexicons to word vectors, but we show that the word-level enrichment can also be helpful for sentence-level downstream tasks.

1.1.4 Cross-domain transfer learning with multi-source domain-adversarial training for slot-filling (Chapter 6)

We propose a transfer learning method separately representing domain-general information and domain-specific information applied to slot filling. We show the effectiveness of encouraging domain-general representations for all the domains to be domain-invariant and domain-specific representations to be less correlated to the domain-general representations on transfer learning for slot filling tasks. Our model can be effectively generalized to transfer learning cases with more than one source domain by using multi-source domain-adversarial training. We demonstrate that our approach can improve the performance of transfer learning for sequence models given multiple domains with different label spaces. Our model shows the best F1-scores compared to other recently introduced transfer learning models for slot filling.

1.1.5 Cross-lingual transfer learning for POS tagging without cross-lingual resources (Chapter 7)

Training a POS tagging model with cross-lingual transfer learning usually requires linguistic knowledge and resources about the relation between the source language and the target language. We introduce a cross-lingual transfer learning model for POS tagging without ancillary resources such as parallel corpora. The proposed cross-lingual model utilizes a common BLSTM that enables knowledge transfer from other languages, and private BLSTMs for language-specific representations. The cross-lingual model is trained with language-adversarial training and bidirectional language modeling as auxiliary objectives to better represent language-general information while not losing the information about a specific target language. Evaluating on POS datasets from 14 languages in the Universal Dependencies corpus (Nivre et al., 2016), we show that the proposed transfer learning model improves the POS tagging performance of the target languages without exploiting any external linguistic knowledge between the source language and the target language.

1.1.6 Word embeddings with multiplicative feature interactions for tensor-based compositions (Appendix A)

As an appendix, we show that the word embeddings from extended CBOW models using multiplication or tensor product between context words, reflecting the actual composition methods, can perform better than those from the baseline CBOW model in actual tasks of compositions with multiplication or tensor-based methods.

1.1.7 Contributions in transfer learning perspective

Our contributions can be described as transfer learning methods. Table 1.2 shows the type of each chapter in terms of transfer learning.

Chapter	Source Type	Transfer Type	Transfer Level	End-task Level
3	-	-	-	Word
4	Resource-based	Inductive	Word	Word
5	Resource-based	Inductive	Word	Sentence
6	Model-based	Inductive	Sentence	Sentence
7	Model-based	Inductive (Character), Transductive (Sentence)	Character, Sentence	Sentence
A	-	-	-	Phrase, Sentence

Table 1.2: Transfer learning category for each chapter. Transfer level denotes the level that knowledge transfer is made, source type denotes the type of the source knowledge (Yang et al., 2017), transfer type is either inductive or transductive (Pan and Yang, 2010), and end-task level denotes the level of the evaluated end-task.

1.2 Outline

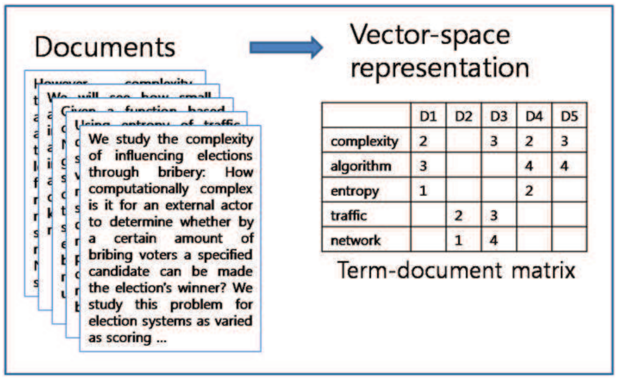
The outline of this dissertation is as follows. In Chapter 2, we describe different types of word embedding methods, BLSTM and Convolutional Neural Network (CNN) models for sentence representations, and adversarial training used for cross-domain and cross-lingual transfer learning.

We show semantic representations of words in vector spaces focusing on semantic intensity scales in Chapter 3. Chapter 4 describes enriching word embeddings with semantic intensity orders as well as thesauruses. In contrast to Chapters 3 and 4, which address word-level NLP tasks, Chapter 5 demonstrates the usage of enriched word embeddings for intent detection, a sentence-level downstream task, especially when the number of training examples is small. Chapter 6 describes cross-domain transfer learning for slot-filling focusing on multi source cases. Chapter 7 introduces cross-lingual transfer learning for POS tagging without using any cross-lingual knowledge or resources. Finally, Chapter 8 concludes this dissertation and discusses future work. Appendix A introduces an extension of the CBOW model, which uses different phrase composition methods.

CHAPTER 2: BACKGROUND

2.1 Word representations in vector spaces

Word semantics can be represented based on the distributional hypothesis, *a word is characterized by the company it keeps* (Firth, 1957). Within the hypothesis, each word can be represented as a vector of the context around the word. Co-occurrence based methods represent words as the frequencies of neighboring words within the context ranges. Figure 2.1 is an example that represents each term as a vector of the term frequency in different documents.



We study the complexity of influencing elections through bribery: How computationally complex is it for an external actor to determine whether by a certain amount of bribing voters a specified candidate can be made the election's winner? We study this problem for election systems as varied as scoring ...

	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Figure 2.1: An example of term-document matrix that represents each word as a vector from Yoo and Choi (2008).

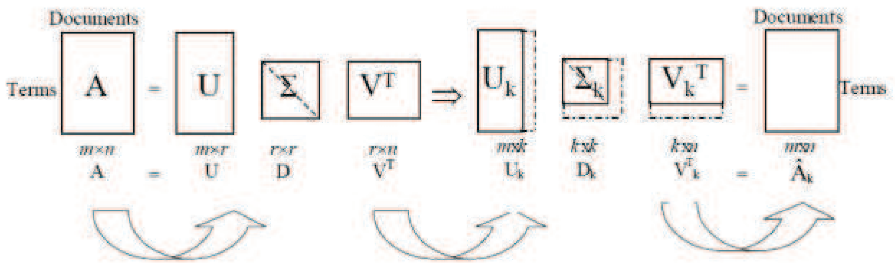


Figure 2.2: The process of decomposition (left), dimensionality reduction (middle), and recovery (right) of a term-document matrix from Wang (2010).

2.1.1 Latent semantic analysis (LSA)

Representing each word as the frequencies over all the documents is not very effective because the dimensionality can be very high and it can be sensitive to outlier documents. To address those issues, we can reduce the dimensionality with latent semantic analysis (LSA). Figure 2.2 illustrates the process of reducing the dimensionality of a term-document matrix and then recover the original dimensionality. Through this process, the term-document matrix is first decomposed into three matrices with singular value decomposition (SVD). Then, we can reduce the dimensionalities of the decomposed matrices by truncating the later part of the matrices. Finally, we can recover the dimensionality of original matrix by multiplying the reduced matrices. In the second step of the process, U_k has m rows which correspond to the vocabulary size and k columns which are the vector representations of the words. As it is known that the reduction by LSA is the best linear approximation to the original matrix by another low rank (Eckart and Young, 1936), U_k is an effective vector representation of words.

2.1.2 Neural word embeddings

In the previous approach, the vocabulary was represented as a V by d matrix, where V is the size of the vocabulary and d usually smaller than V.

Co-occurrence based models with LSA have been widely used to obtain word embeddings. In recent years, however, neural word embeddings are being actively researched along with a boom in deep learning. Neural word embeddings are the projection matrices of neural networks trained for certain tasks given one-hot encoded input word sequences. Compared to word embeddings from co-occurrence based models that can only represent linear feature interactions, neural word embeddings can be trained to represent nonlinear feature interactions via nonlinear activation functions like sigmoid functions. In addition, while the contexts of the co-occurrence based models are just consecutive neighboring words or words co-occurring in a same document without considering the word orders, neural word embeddings can use different types of contexts following the neural network architectures. Neural word embeddings can also be used for the pretraining of neural networks for downstream NLP tasks to boost the performance.

2.1.3 Word embeddings from language models

Neural word embeddings were first introduced as by-products of feed-forward neural network language models (FFNNLM) (Bengio et al., 2003; Schwenk, 2007). The left part of Figure 2.3 shows a FFNNLM architecture that predicts the current word given three previous words. Bengio et al. (2003) showed that FFNNLMs project semantically and syntactically similar words to have similar vector representations. This is one of the main ideas behind word embeddings. Following the distributional hypothesis mentioned in Section 2.1, if two words are syntactically and semantically similar, their context words are also similar. For example, assuming that “beautiful" and “gorgeous" are similar, they would appear to have similar contexts in corpora like:

She is very beautiful.

She is very gorgeous.

Such beautiful scenery.

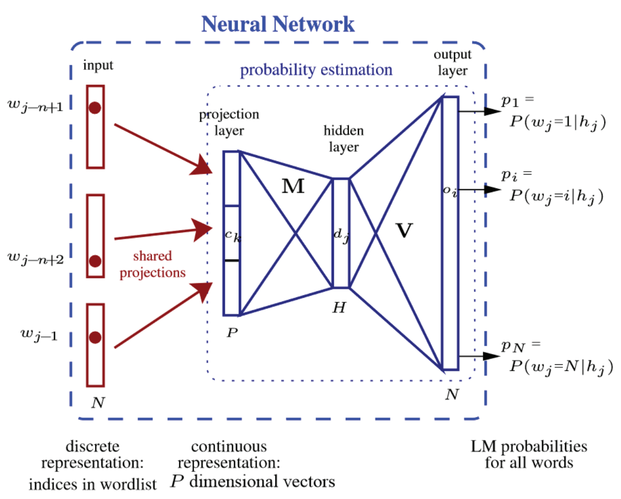
Such gorgeous scenery.

The objective function to be optimized can be modeled as follows:

$$\sum_{i} \ln p\left(t_i | c_i, p_i\right), \tag{2.1}$$

where t_i is the target, c_i is the context, and p_i is the word vector at the ith time step; if a FFNNLM is used, t_i is the following word, c_i is the concatenation of a fixed number of previous word vectors, and p_i is the word vector of the ith word in the corpus. From the example corpus, if p_j and p_k are the word vectors of “beautiful" and “gorgeous", respectively, then $c_j = c_k$ and $t_j = t_k$. Since the contexts are the same and the targets are the same for both “beautiful" vector and “gorgeous" vector, their error surfaces regarding the objective function with respect to the word vectors are also the same. As we usually initialize word vectors similarly with a small perturbation, both word vectors will be directed to the same local optimum of the objective function by gradient descent methods. In real corpora, the error surfaces would not be equivalent but still similar. Therefore, “beautiful" vector and “gorgeous" vector would also get close.

Because the input contexts of FFNNLMs are limited as the fixed number of previous words for targeting of the current word, their performance was not very significant. The right part of Figure 2.3 shows an architecture of RNNLMs. In contrast to FFNNLMs, recurrent neural network language models (RNNLMs) can use arbitrary number of previous



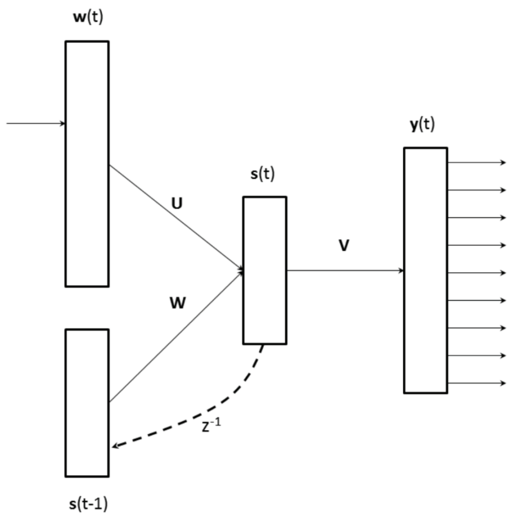


Figure 2.3: The architectures of the feed-forward neural language model (left) from Bengio et al. (2003) and the recurrent neural language model (right) from Mikolov et al. (2010).

words as the context via recurrences. Therefore, the language model performance has been improved compared to the feed-forward models.

In addition, it has been shown that the word embeddings from the RNNLM have syntactic and semantic regularities (Mikolov et al., 2013c). From (Mikolov et al., 2013c), Figure 2.4 shows illustrations that the words in the embedding space are semantically and syntactically consistent in terms of relative positions. The examples illustrate that each semantic or syntactic transformation of words can be done by adding the difference between a vector and another vector that is syntactically or semantically transformed from the original vector. The left panel of the figure shows that the gender of words can be transformed consistently by just adding a vector of the gender difference (blue arrow). The right panel shows that singular nouns can be transformed to plural nouns by adding another vector (read arrow). Also, by adding both the gender changing vector and the number changing vector to “king", we can get “queens".

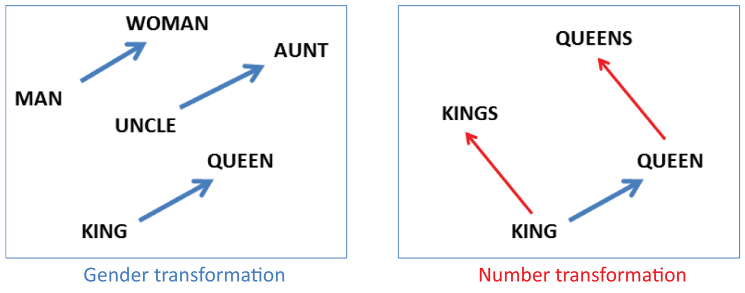


Figure 2.4: Examples of the semantic and syntactic regularities from Mikolov et al. (2013c).

2.1.4 Word2Vec

Although FFNNLMs and RNNLMs can be used to obtain word embeddings, the performance might not be the best because only the previous word contexts are used to predict the current word. If we are focusing on word embeddings rather than language modeling, we can design better neural network models dedicated to word embeddings.

Figure 2.5 shows the architectures of the continuous bag-of-words (CBOW) model and the skip-gram model (Mikolov et al., 2013a,b).[3] In contrast to FFNNLMs and RNNLMs, the CBOW model and the skip-gram model consider the context information from both previous words and following words. As we can use the context from both directions, their word embedding performance has been shown to be better for representing word relationships than those of FFNNLMs and RNNLMs. The CBOW model uses the average neighboring word vectors to predict the current word while the skip-gram model predicts the neighboring words given the current word.

[3]Although sum is used for the CBOW model in Mikolov et al. (2013a), the current version of `word2vec` implementation (`https://code.google.com/p/word2vec`) uses mean indeed.

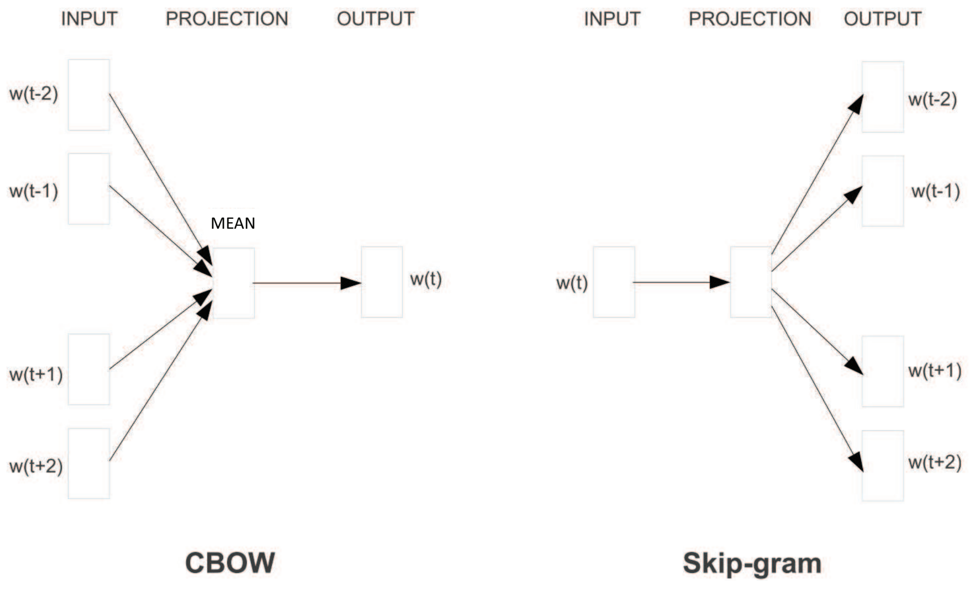


Figure 2.5: The architecture of the continuous bag-of-words model (left) and the skip-gram (right) from Mikolov et al. (2013a).

Given a word sequence $w_1, w_2, ..., w_T$, a corresponding word projection sequence $p_1, p_2, ..., p_T$, and a window size c, Equation 2.2 and 2.3 are the objective functions of the CBOW model and the skip-gram model, respectively.

$$\frac{1}{T}\sum_{t=1}^{T} \ln p\left(w_t \middle| \frac{1}{2c} \sum_{-c\leq j\leq c, j\neq 0} p_{t+j}\right) \tag{2.2}$$

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c\leq j\leq c, j\neq 0} \ln p\left(w_{t+j}|p_t\right) \tag{2.3}$$

Although their formulations are quite opposite, their word embedding performance has been shown to be comparable (Mikolov et al., 2013a).

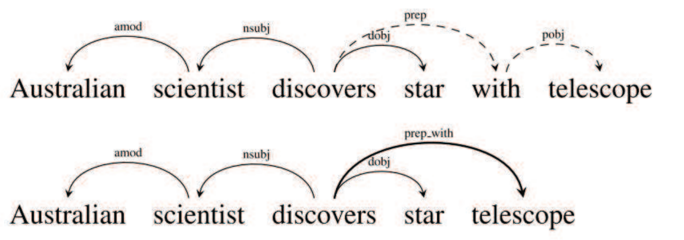
One issue of the CBOW model is that it uses the average vector of the context words to form the input for predicting the current word. In Appendix A, we demonstrate extensions of the CBOW model using different composition methods of the context words, which show better performances for certain phrase-level tasks (Kim et al., 2015a).

2.1.5 Dependency-based word embeddings

As Word2Vec models use bag-of-words contexts that are projections of the neighbor words, it cannot effectively use discontinuous contexts or syntactics of the input word sequences. We can deal with this issue by training the word embeddings using the dependency trees.

Word embeddings trained with the dependency trees to form the contexts show less topical but more functional similarities (Levy and Goldberg, 2014). For example, given “hogwarts", the similar words from the original skip-gram model were “dumbledore", “hallows", “half-blood", “malfoy", and “snape", which are topically related to “hogwarts". Differently, as “hogwarts" is a name of fictional school, the similar words from the model considering the dependency structures were “sunnydale", “collinwood", “calarts", “greendale", and “millfield", which are names of famous schools.

Figure 2.6 shows how previous work represented the targets and the input contexts given a sentence with dependency trees. From the original dependency tree at the top of the figure, relations including prepositions are collapsed resulting in the tree in the middle. Then, the targets and the contexts are extracted from the dependencies. Each context consists of the word and the relation. The inverse ($^{-1}$) marker is used to represent the direction of the dependencies. Then, the target and the contexts are included in the training so that the model encourages predicting the contexts given the target.


amod
nsubj
prep
dobj
pobj
Australian scientist discovers star with telescope
amod
nsubj
prep_with
dobj
Australian scientist discovers star telescope

WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Figure 2.6: An example of dependency trees of a sentence and the corresponding target/input contexts from Levy and Goldberg (2014).

2.1.6 GloVe: Global vectors for word representation

One drawback of Word2Vec models is that they are shallow window-based models, which do not utilize global co-occurrence statistics of given corpora. Pennington et al. (2014) introduced GloVe, a log-bilinear model to train word embeddings using global word co-occurrence statistics. The loss function of GloVe is formulated as follows:

$$\hat{J} = \sum_{i,j} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j - \log X_{ij}\right)^2, \tag{2.4}$$

where w_i is the i-th word vector, $\tilde{w}_j$ is the separate j-th context word vector, X_{ij} is the number of co-occurrences of the i-th word and the j-th context word. $f\left(x\right)$ is used for dealing with low frequency cases and defined as follows:

$$f\left(x\right) = \begin{cases} \left(x/x_{max}\right)^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise ,} \end{cases} \tag{2.5}$$

where $x_{max} = 100$ and $\alpha = 3/4$ in Pennington et al. (2014).

2.1.7 Embeddings from paraphrase pairs

While previous word embedding models depend on distributional hypothesis to obtain word vectors from given corpora, Wieting et al. (2015) utilized PPDB (Ganitkevitch et al., 2013), which can directly relate different sentences with the same semantics. The objective function is as follows:

$$\begin{aligned}\min_{W_w} \frac{1}{|X|} \Bigg(& \sum_{\langle x_1, x_2 \rangle \in X} \max\left(0, \delta - W_w^{(x_1)} \cdot W_w^{(x_2)} + W_w^{(x_1)} \cdot W_w^{(t_1)}\right) \\ & + \max\left(0, \delta - W_w^{(x_1)} \cdot W_w^{(x_2)} + W_w^{(x_1)} \cdot W_w^{(t_2)}\right) \Bigg) + \lambda W_w \left\| W_{w_{init}} - W_w \right\|^2 ,\end{aligned} \tag{2.6}$$

where X is the paraphrase database, W_w^x is the word vector of word x, t_1 and t_2 are randomly chosen words, $W_{w_{init}}$ is the initial word embeddings (skip-gram from Word2vec), and δ and λ are hyperparameters. Word embeddings obtained with this method showed better performance on word semantic analogy task like SimLex-999 (Hill et al., 2015) than word embeddings from Word2Vec models (Wieting et al., 2015). Also, it was used for improving dialog state tracking performance (Mrkšić et al., 2017).

2.2 Sentence representations in vector spaces

On top of the vector sequence of an input word sequence, we predict an output for each time step or for the entire sequence. For the former case, recent models usually use unidirectional or bidirectional LSTMs to obtain sequence outputs considering the contexts.

LSTM is formulated as follows:

$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
\tilde{c}_t &= \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \\
o_t &= \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \\
h_t &= o_t \cdot \tanh\left(c_t\right),
\end{aligned}
\tag{2.7}
$$

where f_t controls how much cell information from the previous time step will be maintained, i_t controls how much information from the current time step will be utilized, $\tilde{c}_t$ represents the information from the current time step, c_t represents the weighted summation of the information from the previous time step and the information from the current time step, o_t controls how much information will be passed to the output of the current time step, and h_t represents the LSTM hidden output.

To obtain a single vector representing an entire word sequence (sentence), which is necessary for sentence-level prediction tasks, there are following approaches.

2.2.1 Sum/Average vector

The simplest approach to obtain a single vector representing a sentence is the sum or the average vectors of the words in the sentence. Although this approach ignores word orders, it has been shown to be an effective baseline for the sentence representation (Wieting et al., 2016).

2.2.2 The last LSTM outputs

Since LSTM outputs are produced considering the context information, we can use the last LSTM output as the vector representing the entire sentence. Cho et al. (2014); Sutskever et al. (2014) introduced machine translation models using sequence-to-sequence architectures, which encode an entire input word sequence of a source language into a single vector using LSTMs or Gated Recurrent Units (GRU) and then decode the vector to a word sequence of a target language. We can also use the concatenation of both last outputs of the forward LSTM and the backward LSTM to represent an entire sentence. This BLSTM-based approach is utilized for intent detection in Chapter 5 and word vector representation from character embeddings in Chapter 7.

2.2.3 Convolutional neural networks (CNN)

CNNs have been shown to be effective in terms of obtaining a single vector over a given sequence for text classification tasks (Kim, 2014). In Kim (2014)'s model, the output of a convolution filter with size h is formulated as follows:

$$c_i = \tanh\left(w \cdot x_{i:i+h-1} + b\right), \tag{2.8}$$

where x_i:i+h-1 is the vector concatenation from the (i)-th word to the $(i+h-1)$-th word, w is the filter weight matrix, b is the bias term, and c_i is the output at the time step i. For a sentence with length n, the produced feature map is $c = [c_1, c_2, ..., c_{n-h+1}]$, and the max-pooling output is $\hat{c} = \max c$.

We use CNN similarly to Kim (2014) on top of BLSTM outputs to make a language discriminator in Chapter 7.

2.3 Adversarial training

Adversarial training was first introduced in generative adversarial networks (GANs) (Goodfellow et al., 2014), and shown to be effective for generating synthetic images that are hard to distinguish from real images. GAN formulation is as follows:

$$\mathbb{E}_{x \sim p_{data}(x)} \left[\log D\left(x\right) \right] + \mathbb{E}_{z \sim p_z(z)} \left[\left(1 - \log D\left(G\left(z\right)\right)\right) \right], \tag{2.9}$$

where D is a discriminator classifying whether the input is a real image or a synthetic image, G is a image generator, x is a real image, and z is a sampled synthetic image. GAN optimization encourages discriminator D to better classify the given input, while the generator is encouraged to generate synthetic images that are hard to distinguish from the real images by the discriminator D. Theoretically, the discriminator is trained to reduce Jensen-Shannon divergence between the distribution of real images and the distribution of generated images.

Text sequences can also be generated using adversarial training combined with text generation methods (Zhang et al., 2016b; Kusner and Hernández-Lobato, 2016; Yu et al., 2017; Li et al., 2017).

Ben-David et al. (2006, 2010) formulated that domain knowledge transfer can be done effectively when the input representations from different domains are indistinguishable to prediction models. Based on the indistinguishable representations obtainable from adversarial training and the theory of domain knowledge transfer, Ganin et al. (2016) introduced domain-adversarial training, which encourages the representations to be domain invariant regardless of their domains using adversarial training methods. Domain separation networks also showed that domain-adversarial training is effective for encouraging the domain-general representations to be domain invariant (Bousmalis et al., 2016).

Adversarial training was also applied for other NLP tasks such as semi-supervised text classification (Miyato et al., 2017), cross-lingual sentiment classification, (Chen et al., 2017), and aspect-dependent text classification (Zhang et al., 2017).

2.3.1 Adversarial training with Wasserstein distance

Goodfellow et al. (2014) showed that their GAN training model, which is formulated in Equation 2.9, reduces Jensen-Shannon divergence between the real data distribution and the generated data distribution. Jensen-Shannon divergence is formulated as follows:

$$JS\left(\mathbb{P}_r, \mathbb{P}_g\right) = KL\left(\mathbb{P}_r || \mathbb{P}_m\right) + KL\left(\mathbb{P}_g || \mathbb{P}_m\right), \tag{2.10}$$

where $\mathbb{P}_r$ is the real data distribution, $\mathbb{P}_g$ is the generated data distribution, and $\mathbb{P}_m$ is their mixture, $\left(\mathbb{P}_r + \mathbb{P}_g\right)/2$. KL denotes Kullback-Leibler (KL) divergence and formulated as follows:

$$KL\left(\mathbb{P}_r || \mathbb{P}_m\right) = \int \log\left(\frac{P_r\left(x\right)}{P_m\left(x\right)}\right) P_r\left(x\right) d\mu\left(x\right). \tag{2.11}$$

The problem of using KL divergence is that the divergence cannot be properly measured for certain distributions. For example, as shown in Arjovsky et al. (2017), if $\mathbb{P}_0$ and $\mathbb{P}_\theta$ are $(0, Z) \in \mathbb{R}^2$ and $(\theta, Z) \in \mathbb{R}^2$, respectively, where $Z \sim U\left[0, 1\right]$, then Jensen-Shannon divergence cannot properly reflect the difference between the two distributions as follows:

$$JS\left(\mathbb{P}_0, \mathbb{P}_\theta\right) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0. \end{cases} \tag{2.12}$$

Also, if the discriminator perfectly classify real data and generated data during the GAN training, then there is little gradients from the discriminator, thereby the generator cannot be properly trained. Therefore, in practice, the discriminator training and the generator training should be balanced so that the discriminator can produce sufficient gradients for the generator training.

Wasserstein GAN is one of the approaches resolving the GAN training problem by using Wasserstein distance instead of Jensen-Shannon divergence (Arjovsky et al., 2017). Wasserstein distance is formulated as follows:

$$W\left(\mathbb{P}_r, \mathbb{P}_\theta\right) = \underset{||f||_L \leq 1}{sup}\ \mathbb{E}_{x\sim\mathbb{P}_r}\left[f\left(x\right)\right] - \mathbb{E}_{x\sim\mathbb{P}_\theta}\left[f\left(x\right)\right], \tag{2.13}$$

where $f\left(x\right)$ is a 1-Lipschitz function. (Arjovsky et al., 2017) also showed that reducing Wasserstein distance in GAN can be formulated as follows:

$$\underset{w\in W}{max}\mathbb{E}_{x\sim\mathbb{P}_r}\left[f_w\left(x\right)\right] - \mathbb{E}_{z\sim p(z)_\theta}\left[f_w\left(g_\theta\left(z\right)\right)\right], \tag{2.14}$$

where $\left\{f_w\right\}_{w\in W}$ are K-Lipschitz for some K, which can be simply ensured by limiting the range of the parameters of the discriminator[4], and $g_\theta\left(z\right)$ denotes the generator. Since Wasserstein distance is measurable for any kinds of distributions, we can train GAN without concerns for the original GANs.

[4] Arjovsky et al. (2017) clipped the parameter values of the discriminator to be between -0.01 and 0.01.

CHAPTER 3: REPRESENTATIONS OF SEMANTIC INTENSITY SCALES IN VECTOR SPACES

We first discuss the semantics of word vector representations obtained from simple word embedding models without any other explicit knowledge transfer methods. Specifically, we address the representation of semantic scales in vector spaces, and show that word vectors from a recurrent neural network language model (RNNLM) can effectively represent semantic scales in vector spaces.

A portion of this chapter has been published as Kim and de Marneffe (2013).

3.1 Introduction

There has recently been a surge of interest for deep learning in natural language processing. In particular, neural network language models (NNLMs) have been used to learn distributional word vectors (Bengio et al., 2003; Schwenk, 2007; Mikolov et al., 2010): the models jointly learn an embedding of words into an n-dimensional feature space. One of the advantages put forth for such distributed representations compared to traditional n-gram models is that similar words are likely to have similar vector representations in a vector space, whereas the discrete units of an n-gram model do not exhibit any inherent relation with one another. It has been shown that the vector space representations improve performance in a variety of NLP tasks, such as POS tagging, semantic role labeling, named entity resolution, parsing (Collobert and Weston, 2008; Turian et al., 2010; Huang et al., 2012).

Mikolov et al. (2013c) show that there are some syntactic and semantic regularities in the word representations learned, such as the singular/plural relation (the difference of singular and plural word vectors are equivalent: *apple* $-$ *apples* $\approx$ *car* $-$ *cars* $\approx$ *family* $-$ *families*) or the gender relation (a masculine noun can be transformed into the feminine form: *king* $-$ *man* $+$ *woman* $\approx$ *queen*).

We extend Mikolov et al. (2013c)'s approach and explore further the interpretation of the vector space. We show that the word vectors learned by NNLMs are meaningful: we can extract scalar relationships between adjectives (e.g., *bad* $<$ *okay* $<$ *good* $<$ *excellent*), which can not only serve to build a sentiment lexicon but also be used for inference. To evaluate the quality of the scalar relationships learned by NNLMs, we use the indirect *yes/no* question answer pairs (IQAP) from de Marneffe et al. (2010), where scales between adjectives are needed to infer a *yes/no* answer from a reply without explicit *yes* or *no* such as *Was the movie good? It was excellent.* Our method reaches 72.8% accuracy, which is the best result reported so far as far as we know without using any other linguistic knowledge or resources.

3.2 Previous work

Our work is based upon the foundation of word vector representations from Mikolov et al. (2011), extracted from a recurrent neural network language model (RNNLM), whose three-layer architecture is represented in Figure 3.1.

In the input layer, $w(t)$ is the input word represented by one-hot encoding at time t when the vocabulary size is N. When there are M nodes in the hidden layer, the number of connections between the input layer and the hidden layer is NM and the connections can be represented by a matrix U.

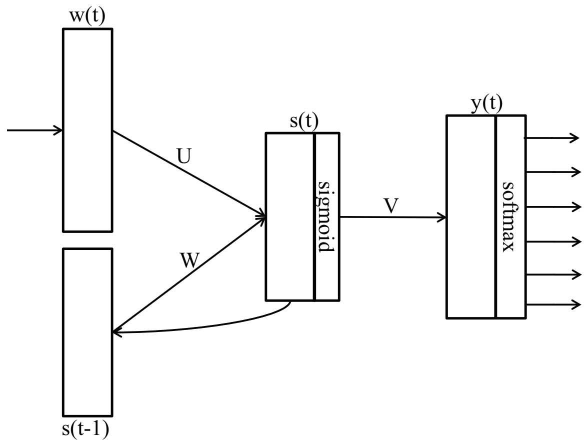


Figure 3.1: The architecture of the RNNLM.

The hidden layer is also connected recurrently to the context $s\left(t-1\right)$ at time $t-1$, with the initial state $s\left(0\right)$ initialized with small values like 0.1. The connections between the previous context and the hidden layer are represented by a matrix W. The dimensionality of the word representations is controlled by the size of W. The output of the hidden layer is $s\left(t\right)=f\left(Uw\left(t\right)+Ws\left(t-1\right)\right)$, where f is a sigmoid function.

Because the inputs of the hidden layer consist of the word $w\left(t\right)$ and the previous hidden layer output $s\left(t-1\right)$, the current context of the RNN is influenced by the current word and the previous context. Therefore, we can regard that the word vector representations from the RNNLM exploit the context implicitly considering the word sequence information (Mikolov et al., 2010).

V is a N by M matrix representing the connections between the hidden layer and the output layer. The final output is $y\left(t\right)=g\left(Vs\left(t\right)\right)$, where g is a softmax function to represent the probability distribution over all the words in the vocabulary.

When the RNN is trained by the back propagation algorithm, we can regard the ith column vector of U as the word vector representation of the ith word in the vocabulary since the column was adjusted correspondingly to the ith element of $w(t)$. Because the $s(t)$ outputs of two input words will be similar when they have similar $s(t-1)$ values, the corresponding column vectors of the words will also be similar.

Mikolov et al. (2013c) showed that constant vector offsets of word pairs can represent linguistic regularities. Let w_a and w_b denote the vectors for the words a and b, respectively. Then the vector offset of the word pair is $w_a - w_b$. If a and b are syntactically or semantically related, the vector offset can be interpreted as a transformation of the syntactic form or the meaning. The offset can also be added to another word vector c. The word vector nearest to $w_a - w_b + w_c$ would be related to word c with the syntactic or semantic difference as the difference between a and b, as it is the case for the *king*, *man*, and *woman* example, where *king* $-$ *man* $+$ *woman* would approximately represent *king* with feminine gender (i.e., *queen*). They also tried to use the vector representations generated by Latent Semantic Analysis (LSA) (Landauer et al., 1998). However, the results using LSA were worse because LSA is a bag-of-words model, in which it is difficult to exploit word sequence information as the context.

For all the experiments in this chapter, we use the precomputed word representations generated by the RNNLM from Mikolov et al. (2013c). Their RNN is trained with 320M words from the Broadcast News data (the vocabulary size is 82,390 words), and we used word vectors with a dimensionality of 1,600 (the highest dimensionality provided).[5] We

[5]We also experimented with smaller dimensions, but consistent with the analyses in Mikolov et al. (2013c), the highest dimensionality gave better results.

Input words	Words with highest cosine similarities to the mean vector			
good:best	**better: 0.738**	strong: 0.644	normal: 0.619	less: 0.609
bad:worst	terrible: 0.726	great: 0.678	horrible: 0.674	**worse: 0.665**
slow:slowest	**slower: 0.637**	sluggish: 0.614	steady: 0.558	brisk: 0.543
fast:fastest	**faster: 0.645**	slower: 0.602	quicker: 0.542	harder: 0.518

Table 3.1: Words with corresponding vectors closest to the mean of positive:superlative word vectors.

First word (-)		1st quarter		Half		3rd quarter		Second word (+)	
furious	1	angry	0.632	unhappy	0.640	pleased	0.516	**happy**	1
furious	1	angry	0.615	tense	0.465	quiet	0.560	**calm**	1
terrible	1	horrible	0.783	incredible	0.714	wonderful	0.772	**terrific**	1
cold	1	mild	0.348	warm	0.517	sticky	0.424	**hot**	1
ugly	1	nasty	0.672	wacky	0.645	lovely	0.715	**gorgeous**	1

Table 3.2: Adjectival scales extracted from the RNN: each row represent a scale, and for each intermediate point the closest word in term of cosine similarity is given.

standardized the dataset so that the mean and the variance of the representations are 0 and 1, respectively.[6]

3.3 Deriving adjectival scales

Here we explore further the interpretation of word vectors. Assuming that the transformation of form or meaning represented by the vector offset is linear, an intermediate vector between two word vectors would represent some “middle" form or meaning. For example, given the positive and superlative forms of an adjective (e.g., *good* and *best*), we expect that the word representation in the middle of them will correspond to the comparative form (i.e.,

[6]http://www.fit.vutbr.cz/~imikolov/rnnlm/word_projections-1600.txt.gz

better). To extract the "middle" word between two word vectors w_a and w_b, we take the vector offset $w_a - w_b$ divided by 2, and add w_b: $w_b + (w_a - w_b)/2$. The result corresponds to the midpoint between the two words. Then, we find the word whose cosine similarity to the midpoint is the highest.

Table 3.1 gives some positive:superlative pairs and the top four closest words to the mean vectors, where the distance metric is the cosine similarity. The correct comparative forms (in bold) are quite close to the mean vector of the positive and superlative form vectors, highlighting the fact that there is some meaningful interpretation of the vector space: the word vectors are constituting a scale.

We can extend this idea of extracting an ordering between two words. For any two semantically related adjectives, intermediate vectors extracted along the line connecting the first and second word vectors should exhibit scalar properties, as seen above for the positive-comparative-superlative triplets. If we take two antonyms (*furious* and *happy*), words extracted at the intermediate points x_1, x_2 and x_3 should correspond to words lying on a scale of happiness (from "less furious" to "more happy"), as illustrated in Figure 3.2.

Table 3.2 gives some adjectival scales that we extracted from the word vector space, using antonym pairs. We picked three points with equal intervals on the line from the first to the second word (1st quarter, half and 3rd quarter). The extracted scales look quite reasonable: the words form a continuum from more negative to more positive meanings.

Tables 3.1 and 3.2 demonstrate that the word vector space is interpretable: intermediate vectors between two word vectors represent a semantic continuum.

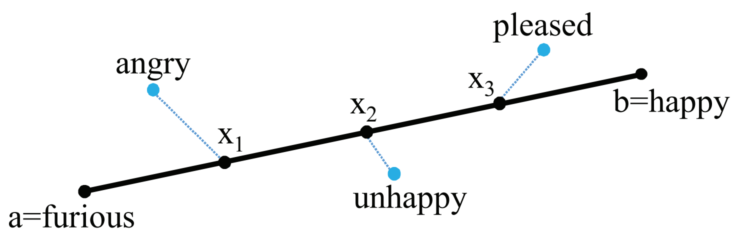


Figure 3.2: An example of vectors with the highest cosine similarity to intermediate points on the line between *furious* and *happy*.

3.4 Evaluation: Indirect answers to *yes/no* questions

To evaluate the quality of the adjective scales learned by the neural network approach, we use the corpus of indirect answers to *yes/no* questions created by de Marneffe et al. (2010), which consists of question-answer pairs involving gradable modifiers to test scalar implicatures. We focus on the 125 pairs in the corpus where both the question and answer contain an adjective: e.g., *Is Obama qualified? I think he's young.*[7] Each question-answer pair has been annotated via Mechanical Turk for whether the answer conveys *yes*, *no* or *uncertain*.

3.4.1 Method

The previous section showed that we can draw a line passing through an adjective and its antonym and that the words extracted along the line are roughly semantically ordered. To infer a *yes* or *no* answer in the case of the IQAP corpus, we use the following approach illustrated with the *Obama* example above (Figure 3.3). Using WordNet 3.1 (Fellbaum, 1998), we look for an antonym of the adjective in the question *qualified*: *unqualified* is retrieved. Since the scales extracted are only roughly ordered, to infer *yes* when the question

[7]These 125 pairs correspond to the ‘Other adjective’ category in de Marneffe et al. (2010).

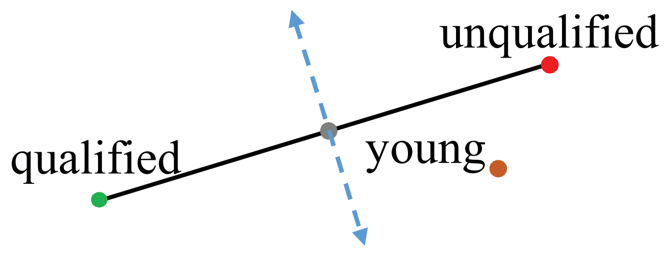


Figure 3.3: An example of the decision boundary given *qualified* as the question and *young* as the answer.

and answer words are very close, we set the decision boundary perpendicular to the line connecting the two words and passing through the midpoint of the line.

Since the answer word is *young*, we check whether *young* is in the area including *qualified* or in the other area. We infer a *yes* answer in the former case, and a *no* answer in the latter case. If *young* is on the boundary, we infer *uncertain*. If a sentence contains a negation (e.g., *Are you stressed? I am not peaceful.*), we compute the scale for *stressed-peaceful* and then reverse the answer obtained, similarly to what is done in de Marneffe et al. (2010).

Since a word can have multiple senses and different antonyms for the senses, it is important to select the most appropriate antonym to build a more accurate decision boundary. We consider all antonyms across senses[8] and select the antonym that is most collinear with the question and the answer. For the word vectors of the question w_q, the ith antonym w_{ant_i}, and the answer w_a, we select ant_i where $argmax_{ant_i}|cos(w_q - w_a, w_q - w_{ant_i})|$. Figure 3.4 schematically shows antonym selection when the question is *good* and the answer is *excellent*: *bad* and *evil* are the antonym candidates of *good*. Because the absolute cosine

[8]Antonyms in WordNet can be directly opposed to a given word or indirectly opposed via other words. When there are direct antonyms for the question word, we only consider those.

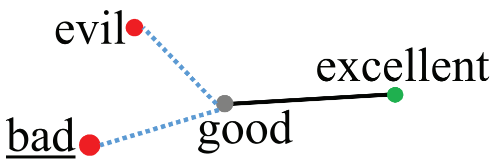


Figure 3.4: An example of antonym selection.

	Acc	Macro P	Macro R	Macro F1
de Marneffe (2010)	60.00	59.72	59.40	59.56
Mohtarami (2011)	–	62.23	60.88	61.55
RNN model	**72.80**	**69.78**	**71.39**	**70.58**

Table 3.3: Score (%) comparison on the 125 scalar adjective pairs in the IQAP corpus.

similarity of *good-excellent* to *good-bad* is higher than to *good-evil*, we choose *bad* as the antonym in this case.

3.4.2 Results and discussion

Table 3.3 compares our results with previous ones where adjectival scales are considered: de Marneffe et al. (2010) propose an unsupervised approach where scales are learned from distributional information in a Web corpus; Mohtarami et al. (2011)'s model is similar to ours but uses word representations obtained by LSA and a word sense disambiguation system (Zhong and Ng, 2010) to choose antonyms. To compare with Mohtarami et al. (2011), we use macro-averaged precision and recall for *yes* and *no*. For the given metrics, our model significantly outperforms the previous ones ($p < 0.05$, McNemar's test).

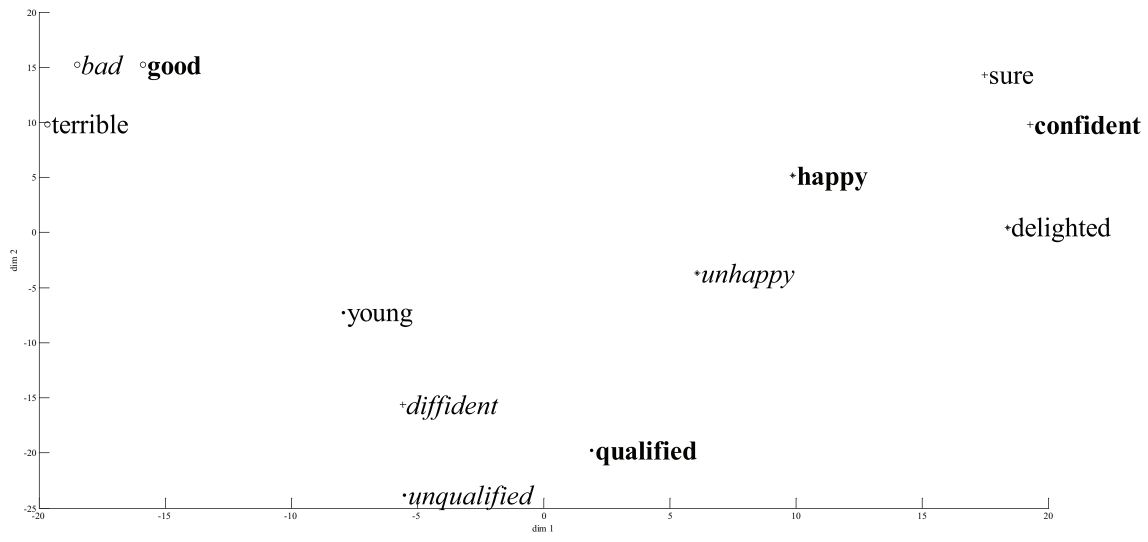


Figure 3.5: Question words (bold), their antonyms (italic), and answer words (normal) of four pairs from the IQAP dataset. The words are visualized by MDS.

Mohtarami et al. (2011) present higher numbers obtained by replacing the answer words with their synonyms in WordNet. However, that approach fails to capture orderings. Two words of different degree are often regarded as synonyms: even though *furious* means extremely angry, *furious* and *angry* are synonyms in WordNet. Therefore using synonyms, the system will output the same answer irrespective of the order in the pair. Mohtarami et al. (2012) also presented results on the interpretation of indirect questions on the IQAP corpus, but their method did not involve learning or using scalar implicatures.

Figure 3.5 gives a qualitative picture: the question words, antonyms and answer words for four of the IQAP pairs are visualized in 2D space by multidimensional scaling (MDS). Note that MDS introduces some distortion in the lower dimensions. Bullet markers correspond to words in the same pair. Question words, antonyms, and answer words are displayed by bold, italic, and normal fonts, respectively. In the *Obama* example previously mentioned (*Is Obama qualified? I think he's young.*), the question word is *qualified* and the answer word is *young*. In Figure 3.5, *qualified* is around (2,-20) while its antonym *unqualified* is around (-6,-24). Since *young* is around (-7,-8), we infer that *young* is semantically closer to *unqualified* which corroborates with the Turkers' intuitions in this case. *e.g.*happy, *e.g.*good and *e.g.*confident give the other examples displayed in Figure 3.5.

(3.1) A: Do you think she'd be *happy* with this book?

B: I think she'd be *delighted* by it.

(3.2) A: Do you think that's a *good* idea?

B: It's a *terrible* idea.

(3.3) A: The president is promising support for Americans who have suffered from this hurricane. Are you *confident* you are going to be getting that?

B: I'm not so *sure* about my insurance company.

In *e.g.*happy, *delighted* is stronger than *happy*, leading to a *yes* answer, whereas in *e.g.*good, *terrible* is weaker than *good* leading to a *no* answer. In *e.g.*confident, the presence of a negation will reverse the answer inferred, leading to *no*.

3.5 Conclusion

This chapter provides further evidence that the relationships in the word vector space learned by recurrent neural network models are interpretable. We show that using vector offsets, we can successfully learn adjectival scales, which are useful for scalar implicatures, as demonstrated by the significantly better results we obtain on the IQAP corpus compared to those of other baseline models.

One problem of using word embeddings for representing word semantics is that antonym words can often be close in vector spaces since they appear within similar contexts in many cases.[9] For example, in Table 3.1, “slower” was the second closest word to the midpoint between “fast” and “fastest”, which is semantically inappropriate. In the next chapter, we discuss the methods dealing with this issue.

[9]In other words, antonym words are in the same semantic class.

CHAPTER 4: ENRICHING WORD EMBEDDINGS WITH SEMANTIC INTENSITY ORDERS

In the previous chapter, we discussed the semantics of word vectors focusing on semantic intensity scales. The word vectors obtained from a RNNLM worked significantly better than previous approaches. However, word vectors obtained from conventional word embedding models have the following issues:

Unclear context scope It is difficult to set the proper scope of the neighbors to be the context. For example, assume that we have two sentences, “The tiger we saw yesterday was very cruel” and “the cat we saw yesterday was very cute”, and we are using five nearest neighbors as the context of the current word. Then, the context words of both “tiger” and “cat” are the same as “the”, “we”, “saw”, ‘yesterday”, “was”, and “very”. In this case, as the contexts are equivalent, we cannot differentiate “tiger” and “cat”, thereby the vectors of “tiger” and “cat” are not correctly trained. Although we can increase the scope by increasing the window size, using LSTMs, or using dependency relations, they might not be sufficient yet if we need to cover the entire paragraph or document to capture the contexts properly.

Incorrect context composition Even if the model’s context covers all the necessary context words, the context might not be best represented yet since it depends on the model’s composition methods such as addition, multiplication, and concatenation. In the neural word embedding models, there is no perfect composition methods. In addition, conventional compositional model cannot deal with non-compositional semantics. For example, the actual meaning of “kick the bucket” is significantly different from the literal meaning, which is difficult to represent with conventional compositional models.

Same semantic classes For better representation of semantic scales, antonyms should be far apart in the vector space. Also, words with different semantic intensities should not be very close. However, because antonyms are in the same semantic class and words with different semantic intensities are near-synonyms, their contexts might not be sufficiently different, thereby the trained vectors might not be far enough as well. Therefore, using the farness information from semantic lexicons would help the words that should be far apart can actually be distant in vector spaces.

Dealing with those issues, one solution is using linguistic resources such as thesauruses to enrich word vectors from word embedding models so that the word vectors can be positioned to represent word semantics better. This approach can be considered as a word-level inductive transfer learning. In this chapter, we specifically focus on the enrichment with semantic intensity orders as well as thesauruses.

A portion of this chapter has been published as Kim et al. (2016a).

4.1 Introduction

Word embedding models that represent words as real-valued vectors have been directly used in word-level NLP tasks such as word similarity (Mikolov et al., 2013c), antonym detection (Ono et al., 2015; Pham et al., 2015; Chen et al., 2015), knowledge relations

(Toutanova et al., 2015; Socher et al., 2013a; Bordes et al., 2013), and semantic scale inference (Kim and de Marneffe, 2013). Word embedding models such as Word2Vec (continuous bag-of-words (CBOW) and skip-gram) (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014), widely used to generate word vectors, are trained following the distributional hypothesis, which assumes that the meaning of words can be represented by their context (Harris, 1954).

However, word embedding models based solely on the distributional hypothesis often place words improperly in vector spaces. For example, in a vector space, a word and its antonym should be sufficiently far apart, but they can be quite close because they can have similar contexts in many cases.

For better semantic representations, different approaches using semantic lexicons as well as lexical knowledge to adjust word vectors have recently been introduced. Faruqui et al. (2015) adjusted each word vector to be in the middle between the initial position and its synonymous words. Mrkšić et al. (2016) used max-margin approaches to adjust each word vector with synonyms and antonyms while keeping the relative similarities to the neighbors. While these two approaches are post-processing models that adjust preexisting word vectors, Ono et al. (2015), Pham et al. (2015), and Liu et al. (2015) jointly train models that augment the skip-gram (Mikolov et al., 2013a) objective function to include knowledge from semantic lexicons. The common goal in these approaches is to make semantically close words closer and semantically distant words farther apart while keeping each word vector not to be too far from the original position. Although the joint training models can even indirectly adjust words that are not listed in the semantic lexicons (Pham et al., 2015), the post-processing models are much more efficient and can be applied to word vectors from any kinds of models, which can eventually perform better than the joint training models (Mrkšić et al., 2016).

Although Faruqui et al. (2015), Mrkšić et al. (2016), Ono et al. (2015), Pham et al. (2015), and Liu et al. (2015)'s adjustment approaches have been shown to represent word semantics better in vector spaces, their coarse modeling of words as synonyms or antonyms may be insufficient for modeling words lying along a semantic intensity scale. For example, assume that "great" is erroneously between "bad" and "good" in a vector space ("bad" should be closer to "good" than "great"). Since semantic lexicons such as WordNet (Fellbaum, 1998) and the Paraphrase Database (PPDB) (Pavlick et al., 2015) only inform us that "good" and "great" are semantically similar and "good" is semantically opposite to "bad", adjusting word vectors with those semantic lexicons does not permit to retrieve the appropriate semantic intensity ordering: bad $<$ good $<$ great.

Accurate representation of such semantic intensity scales can help correct processing in downstream tasks that require robust textual understanding. For instance, given an assertion such as *the movie is outstanding*, statements that contain a semantically weaker expression (e.g., *the movie is good*, *the movie is okay*) are entailed, whereas *the movie is okay* does not entail that *the movie is outstanding*. Similarly, correct information about semantic scales can also provide accurate inferences: when answers to a yes/no question that contains a gradable adjective does not explicitly contain a *yes* or a *no*, we can derive the intended answer by figuring out whether the answer entails or implicates the question (Horn, 1972; Hirschberg, 1985; de Marneffe et al., 2010). For example, for the question *Was the talk good?*, if the answer is *It was excellent*, the answer entails "yes", but if the answer is *It was okay*, "no" will be implied.

To deal with the representation of semantic intensity scales, we infer semantic intensity orders with de Melo and Bansal (2013)'s approach and then use the intensity orders to adjust the word vectors. Evaluating on three different human annotated datasets, we show that the adjustment with intensity orders in addition to adjustments with synonyms and antonyms performs best in representing semantic intensities.

4.2 Adjusting word embeddings with semantic lexicons

In this study, we start from one of three different off-the-shelf word vector types as a baseline for our studies: GloVe, CBOW, and Paragram-SL999 (Wieting et al., 2015); we adjust each of these sets of vectors with a variety of contrastive methods. Our first contrastive system is a baseline using synonyms and antonyms ("syn&ant") following Mrkšić et al. (2016)'s approach, which adjusts word vectors so that the sum of the following three max-margin objective functions are minimized.

Adjusting with antonyms We adjust word vectors so that the cosine similarity between each word and its antonyms is zero or lower:

$$AF(V) = \sum_{(u,w)\in A} \tau\left(\cos\left(v_u, v_w\right)\right), \tag{4.1}$$

where $\tau(x) = \max(0, x)$, 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 word pairs.

Adjusting with synonyms We let the cosine similarities between each word and its synonyms be increased:

$$SC(V) = \sum_{(u,w)\in S} \tau\left(1 - \cos\left(v_u, v_w\right)\right), \tag{4.2}$$

where S is the set of synonym pairs. The synonym pairs consist of the *Equivalence* relations from PPDB word pairs.

Keeping the similarity to the initial neighboring words We encourage the cosine similarity between the initial vectors of each word and a neighbor word to be equal to or higher than the current cosine similarity between them:

$$KN\left(V, V^0\right) = \sum_{i=1}^{N} \sum_{j \in N(i)} \tau\left(\cos\left(v_i, v_j\right) - \cos\left(v_i^0, v_j^0\right)\right), \tag{4.3}$$

where V^0 is the initial vocabulary matrix, N is the vocabulary size, and $N(i)$ is the set of the initial neighbors of the i-th word. Word pairs with cosine similarities equal to or higher than 0.8 are regarded as neighbors.

The objective function for the word vector adjustment is represented as the sum of the three terms:

$$C\left(V, V^0\right) = AF(V) + SC(V) + KN\left(V, V^0\right) \tag{4.4}$$

This function is minimized with stochastic gradient descent with learning rate 0.1 for 20 iterations.

4.3 Adjusting word embeddings with semantic intensity orders

In order to better model semantic intensity ordering, we augment the synonym and antonym adjusted model with semantic intensity information to adjust word vectors. We first cluster semantically related words, infer semantic intensity orders of words in each cluster, and then adjust word vectors based on the intensity orders.

4.3.1 Clustering words for intensity ordering

de Melo and Bansal (2013) used WordNet dumbbells (Gross and Miller, 1990), each of which consists of an adjective antonym pair and each adjective's synonyms, to define a set of words along a semantic intensity scale. Words in each half of a dumbbell form a cluster. This clustering is effective since synonyms are semantically highly related but their intensities may be different. However, this approach can only cluster words listed in WordNet.

Shivade et al. (2015) clustered word vectors from the CBOW model with k-means++ clustering (Arthur and Vassilvitskii, 2007). This approach depends on the current word vector placement and does not require semantic lexicons. However, a word can only belong to one cluster since k-means++ is a hard clustering, thus causing issues with polysemous words. For example, "hot" is both on the temperature scale (e.g., *It's hot today*) and on the interestingness scale (e.g., *It's a hot topic*). If "hot" is adjusted for the former scale, "hot" may not properly be placed on the latter scale. Another issue of using clustering algorithms is that unrelated or antonymous words can belong to a cluster, which may hinder correct intensity ordering.

We evaluated both clustering approaches and their combination to cluster words for intensity orders. In Table 4.3, by default, WordNet dumbbells and *Equivalence* relations of PPDB word pairs are used as the intensity clusters. "kmeans only" denotes that only clusters from k-means++ are used, and "+kmeans" means that WordNet, PPDB, and clusters from k-means++ are used altogether. Following Shivade et al. (2015), when clustering with k-means++, we set k to be 5,798, which is the number of all observed adjectives (17,394) divided by 3 so that the average number of adjectives in a cluster is 3.

Weak-Strong Patterns	Strong-Weak Patterns
* (,) but not *	not * (,) just *
* (,) if not *	not * (,) but just *
* (,) although not *	not * (,) still *
* (,) though not *	not * (,) but still *
* (,) (and/or) even *	not * (,) although still *
* (,) (and/or) almost *	not * (,) though still *
not only * but *	* (,) or very *
not just * but *	

Table 4.1: Regular expression patterns used to obtain weak-strong directions between adjective pairs in de Melo and Bansal (2013).

4.3.2 Inferring intensity ordering

We follow de Melo and Bansal (2013)'s approach to order the adjectives in each cluster. For every possible pair of adjectives in the cluster, we search for regular expressions like "$\langle * \rangle$ but not $\langle * \rangle$" in Google N-gram (Brants and Franz, 2006). Table 4.1 shows the used regular expression patterns in de Melo and Bansal (2013).

These patterns give us the direction of the ordering between the adjectives. For example, if "good but not great" appears frequently in Google N-gram, we infer that "great" is semantically stronger than "good".[10] Once we have the intensity differences of adjective pairs in a cluster, mixed integer linear programming (MILP) is used for optimal ordering of all the adjectives in the cluster given the pairwise intensity information of the adjective pairs, following de Melo and Bansal (2013).

[10] Shivade et al. (2015) used Tregex (Levy and Andrew, 2006) to extract patterns including more words but it is not necessary when we extract patterns from phrases consisting of less or equal to five words.

4.3.3 Adjusting word vectors based on intensity orders

Now that we have word clusters whose constituent words are ordered according to their semantic intensities, we adjust the word vectors in two ways, as follows.

Adjusting words with the same intensity order to be closer

When intensity orders are assigned to words in a cluster, different words can have the same rank. For example, given a word cluster {"interesting", "provocative", "exciting", "sexy", "exhilarating", "thrilling"}, both "exhilarating" and "thrilling" are assigned the highest order, and "exciting" and "sexy" are assigned the second highest order. Since words in a same cluster are considered to be very close in both the meaning and the intensity, it is desirable to let them to be similar in the vector space. Therefore, we formulate a max-margin function:

$$SO\left(V\right) = \sum_{(u,w)\in E} \tau\left(1 - \cos\left(v_u, v_w\right)\right), \tag{4.5}$$

where E is the word pairs of the same intensities from the intensity clusters.

Adjusting weaker/stronger word pairs based on antonyms

For two similar words with different intensities (e.g., "good" and "great"), the similarity between the weaker word vector and its antonym vector should be higher than the similarity between the stronger word vector and the antonym vector. Figure 4.1 shows an example of word vectors which are wrongly ordered.

To reduce wrong orderings, we formulate a max-margin function:

$$AO\left(V\right) = \sum_{(w,a)\in A} \sum_{s\in Str(w)} \tau\left\{\cos\left(v_s, v_a\right) - \cos\left(v_w, v_a\right)\right\}, \tag{4.6}$$

where A is the set of antonym pairs and $Str\left(w\right)$ is a set of words semantically stronger than w. By minimizing this function, out-of-order vectors are adjusted so that the stronger

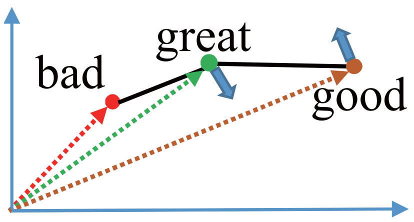


Figure 4.1: An example of incoherent word vector positions, where “bad” should be closer to “good” than “great” but the similarity between “bad” and “good” is lower than the similarity between “bad” and “great”.

word vector gets farther from the antonym vector and the weaker word vector gets closer to the antonym vector.

Both equations 4.5 and 4.6 can be either solely used or summed to others like equation 4.4 to serve as a term of the objective function.

4.4 Evaluation

We evaluate the representation of semantic intensities on the three following human-annotated datasets.

4.4.1 WordNet synset pairs

We obtained a dataset of 670 synonymous adjective pairs coming from synsets in WordNet from Christopher Potts. Each adjective pair was annotated for intensity order on Mechanical Turk. For each adjective pair <A, B> (e.g., “good” and “great”), ten different Turkers were asked to judge whether A is semantically stronger than B, B is semantically stronger than A, or A is equal to B. For consistency of annotation with the other datasets, we mapped “A is semantically stronger than B" to “no”, “B is semantically stronger than A" to “yes”, and “A is equal to B" to “uncertain”.

Max # Turkers agreeing	Coverage (%)
10	17.5
9	17.2
8	13.3
7	14.6
6	14.9
5	16.7
4	6

Table 4.2: Percentage of adjective pairs and the maximum number of Turkers who agree with each other on the annotation.

For 77.3% of adjective pairs, at least 6 out of the 10 Turkers agreed with each other on the same annotation. Table 4.2 gives a breakdown of how often Turkers agree with each other. The inter-annotator agreement (Fleiss' kappa) of this dataset is 0.359. Note that Fleiss' kappa is a very conservative measure given the partial order in the annotation, which is not taken into account in Fleiss' kappa.

4.4.2 Indirect question-answer pairs (IQAP)

IQAP (de Marneffe et al., 2010) is a corpus consisting of 127 indirect question-answer pairs in which both the question and the answer contain a gradable adjective (*Is Obama qualified? I think he's young.*). For each pair, 30 Turkers decided whether the answer implies a "yes", "no" or "uncertain" response to the question. A majority "yes" response implies that the adjective in the question entails the adjective in the answer.

The ordering between the adjectives in the question and in the answer can be used to infer a "yes" or "no" answer: if the adjective in the answer is semantically equivalent or stronger to the adjective in the question, we infer a "yes" answer (*Was the movie good? It was excellent.*); if not, we infer a "no" answer.

Adjustment methods	WordNet synset pairs			IQAP			de Melo & Bansal (2013)		
	GloVe	CBOW	Pgrm	GloVe	CBOW	Pgrm	GloVe	CBOW	Pgrm
baseline	0.5614	0.5092	0.5224	0.7044	0.7016	0.7591	0.9468	0.9347	0.9803
syn&ant	0.5106	0.5516	0.5572	**0.8143**	**0.8045**	0.8307	0.9632	0.9444	0.9791
same_ord (kmeans only)	**0.5762**	0.5163	0.5196	0.7044	0.7016	0.7473	0.9480	0.9359	0.9791
same_ord, diff_ord	0.5505	0.5331	0.5167	0.7119	0.6889	0.7718	0.9456	0.9371	0.9701
syn&ant,same_ord	0.5364	0.5639	0.5782	0.7922	0.7818	0.8284	0.9632	0.9492	0.9803
syn&ant,diff_ord	0.5300	0.5551	0.5765	**0.8143**	0.7922	0.8307	0.9735	0.9539	**0.9825**
syn&ant,same_ord,diff_ord	0.5467	0.5730	**0.5960**	**0.8143**	0.8033	**0.8395**	**0.9758**	0.9539	**0.9825**
syn&ant,same_ord,diff_ord (kmeans only)	0.5186	0.5516	0.5729	0.8033	**0.8045**	0.8194	0.9609	0.9468	0.9803
syn&ant,same_ord,diff_ord (+kmeans)	0.5512	**0.5828**	**0.5960**	0.8033	0.8033	**0.8395**	0.9735	**0.9609**	0.9814

Table 4.3: F1 scores for determining semantic intensity ordering on three datasets, across three baseline models (GloVe, CBOW, Paragram), using different compositions of adjustment techniques, including **syn**onyms, **ant**onyms, **same** intensity **ord**ers, and **diff**erent intensity **ord**ers.

4.4.3 Word intensity orders in clusters

We also use the test set from de Melo and Bansal (2013) consisting of 507 pairs of adjectives in 88 clusters annotated by two native English speakers for intensity ordering. From this set, we generated all the possible adjective pairs from the ordered list in a cluster. For example, for "known" < "famous" < "legendary" in the test set, we generated "known" < "famous", "known" < "legendary", and "famous" < "legendary".

4.4.4 Evaluation results

In our evaluation of the semantic orderings of adjective pairs, we decide which adjective in a pair <A, B> is semantically stronger following Kim and de Marneffe (2013)'s approach. First, we look for an antonym of A.[11] Then, we check whether the word vector

[11] If there are no antonyms of A in WordNet, we obtain antonyms from Roget's thesaurus (Kipfer, 2009).

Datasets	# pairs	# syn	# ant
WordNet synset pairs	670	79	0
IQAP	127	7	9
de Melo & Bansal	507	54	1

Table 4.4: The numbers of total adjective pairs, synonymous pairs, and antonymous pairs for each dataset.

for B is more similar to the vector for A than to the vector for A's antonym, or whether the vector for B is more similar to the vector for A's antonym. We infer a "yes" answer in the former case, and a "no" in the other case. If A has more than one antonym, we select the antonym that is most collinear with the vectors for A and B assuming that the most collinear antonym is most semantically related to A and B.

Table 4.3 shows the F1 scores of different combinations of the adjustments on the three datasets,[12] whereas Table 4.4 shows the number of total adjective pairs in each dataset, as well as the number of pairs in which both adjectives are synonyms (*Equivalence* relations from PPDB) and the number of pairs in which both adjectives are antonyms (*Exclusion* relations from PPDB and antonyms from WordNet).

Expanding on the results in Table 4.3, as the baselines, we used three different 300 dimensional off-the-shelf word vectors: GloVe,[13] CBOW,[14] and Paragram-SL999.[15] Following Mrkšić et al. (2016), for each of the word vector sets, we extracted word vectors corresponding to the 76,427 most frequent words from Open-Subtitles.[16]

[12]For simplicity of the evaluation in vector spaces, we calculate F1 scores without "uncertain" cases.

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

[14]Available from `https://code.google.com/p/word2vec/`

[15]Available from `https://drive.google.com/file/d/0B9w48e1rj-MOck1fRGxaZW1LU2M/`

[16]Available from `invokeit.wordpress.com/frequency-word-lists`

Compared adjustment methods	GloVe				CBOW			
	WN	IQAP	dM&B	merged	WN	IQAP	dM&B	merged
baseline v. syn&ant	-	+	+			+		
baseline v. syn&ant,same_ord,diff_ord	-	+	+			+	+	+
syn&ant v. syn&ant,same_ord,diff_ord			+	+				
baseline v. syn&ant,same_ord,diff_ord (+kmeans)		+	+	+		+	+	
syn&ant v. syn&ant,same_ord,diff_ord (+kmeans)	+			+			+	+

Table 4.5: McNemar's χ^2 test results (p-value < 0.05) for different methods of GloVe/CBOW adjustments across WordNet synset (WN), IQAP, and de Melo & Bansal (dM&B) datasets, as well as concatenating the three datasets (merged). For x v. y, '+' denotes that y's score is significantly higher than that of x, ''-' denotes the opposite, and no value denotes that the difference is not statistically significant.

Table 4.5 indicates whether the differences in performance of the adjustment methods in Table 4.3 are statistically significant (McNemar's χ^2 test with p-value < 0.05). In the table, "merged" columns are the results of the concatenation of all the datasets. For each comparison, '+' denotes that the performance of the latter is significantly higher than that of the former, and '-' denotes the opposite, whereas no value indicates that the difference in performance is not statistically significant. For Paragram vectors, only one case ("baseline" vs "syn&ant,same_ord") is significantly different.

In Table 4.3, "baseline" shows the performance of the baseline word vectors without any adjustments. Since Paragram-SL999 are optimized to perform best on evaluating SimLex-999 dataset, the baseline performance of Paragram-SL999 on SimLex-999 as well as two of the other datasets are noticeably better than word vectors from GloVe and CBOW.

In "syn&ant", corresponding to the optimization with equation 4.4, 15,509 words are adjusted with the synonyms and 6,162 words are adjusted with the antonyms. This adjustment significantly improves the performance of CBOW vectors and Paragram vectors on the IQAP and de Melo and Bansal (2013)'s datasets. Specifically, for the IQAP dataset, where many of the pairs are either synonyms or antonyms, “syn&ant” showed better performance than including adjustments with semantic intensity orders. However, this adjustment makes GloVe vectors yield significantly worse performance on the WordNet synset pair dataset. This shows that the adjustment with just synonyms and antonyms can worsen the representation of subtle semantics considering intensities. In this case, using just the adjustment with semantic intensity orders can be helpful. “same_ord (kmeans only)”, corresponding to equation 4.5, adjusts word vectors by just making vectors of words with the same intensity order to be more similar without using synonyms and antonyms. For GloVe vectors, “same_ord (kmeans only)” showed the highest score for the WordNet synset pair dataset. For adjustments with semantic intensity orders, 616 words are adjusted when WordNet dumbbells and *Equivalence* relations from PPDB word pairs are used as the clusters. When clusters from k-means++ are used, several hundreds of words are adjusted, where the adjusted words vary depending on the vector space for each iteration.

For the WordNet synset pair dataset and de Melo and Bansal (2013)’s dataset, where the subtle semantic intensity differences are more critical, using synonyms, antonyms, and semantic intensity orders altogether (“syn&ant,same_ord,diff_ord”) showed significantly higher scores than “syn&ant” in many settings. Here, “diff_ord” corresponds to equation 4.6.

Table 4.6 shows the adjective pairs whose intensity judgements were changed by including adjustments with semantic intensity orders. The pairs are from the WordNet synset pairs and GloVe vectors were used as the baseline. “baseline” is compared to “same_ord (kmeans only)” in the first column and “syn&ant” is compared to “syn&ant,same_ord,diff_ord(+kmeans)”. In both cases, we observe that some of the incorrectly judged pairs are corrected when adding the adjustment with semantic intensity orders. In these cases, there were no pairs

baseline v. same_ord (kmeans only)	syn&ant v. syn&ant, same_ord,diff_ord(+kmeans)
satisfactory < superb	mediocre < severe
unfavorable < poor	troublesome < rocky
crazy < ardent	upfront < blunt
outspoken < expansive	solid < redeeming
sad < tragic	warm < uneasy
deserving < sacred	valuable < sacred

Table 4.6: Adjective pairs whose incorrect decisions with the former models are corrected by the latter models. For those model comparisons, there were no pairs that were correctly judged with the former models but not with the latter models.

that were correctly judged by the adjustments without semantic intensity orders but incorrectly judged with semantic intensity orders.

Since the numbers of adjectives pairs in the datasets and the numbers of words that are adjusted with semantic intensity orders are small, not all the cases comparing the adjustments using just synonyms and antonyms to the adjustments including semantic intensity orders were significant for p-value < 0.05, as shown in Table 4.5. However, since many of them are slightly insignificant (like p-value=0.07) and the scores noticeably increased in many cases, using semantic intensity orders for the adjustments seem promising.

In addition, to show that the adjustments are not harmful for the representation of the general semantics of the words, we also evaluated on SimLex-999 (Hill et al., 2015), where 999 word pairs were annotated on Mechanical Turk to score the degree of semantic similarities. This dataset has been widely used to evaluate the quality of semantic representations of words.

Table 4.7 shows Spearman's ρ scores on the SimLex-999 dataset for the different adjustment methods. Since SimLex-999 dataset is not directly related to semantic intensities

	GloVe	CBOW	Pgrm
baseline	0.4453	0.4567	0.6920
syn&ant	0.5969	0.5768	0.7268
same_ord (kmeans only)	0.4420	0.4585	0.6926
same_ord, diff_ord	0.4522	0.4613	0.6872
syn&ant,same_ord	0.5969	0.5768	0.7261
syn&ant,diff_ord	0.5958	0.5767	**0.7274**
syn&ant,same_ord,diff_ord	0.5962	**0.5773**	0.7271
syn&ant,same_ord,diff_ord (kmeans only)	**0.5980**	0.5769	0.7269
syn&ant,same_ord,diff_ord (+kmeans)	0.5956	0.5771	0.7273

Table 4.7: Spearman's ρ on SimLex-999.

compared to the other evaluation datasets, there were no significant gains for the adjustments with semantic intensity orders. However, no significant drops indicate that the adjustments with semantic intensity orders are not harmful for the representation of general word semantics.

4.5 Future work

In future work, we plan to investigate clustering techniques beyond WordNet dumbbells and k-means++ as preprocessing in the semantic ordering. The clusters using WordNet dumbbells depend on a preexisting semantic lexicon that may not cover all the semantically related words. With k-means++, clusters may contain semantically opposite words and a word can belong to only one cluster. As both techniques have limitations, by using another clustering method, the performance could be further improved. In addition, we plan to use larger corpora than Google N-gram so that we can find more intensity orderings within clusters. We can also further improve the performance by using semantic intensity information from other linguistic resources. For example, given a list of base, comparative, and superlative forms of adjectives and adverbs, we can let those adjectives aligned more correctly in vector spaces. We can also use word definitions from dictionaries. For example, from *American Heritage Dictionary*, one of the definitions of “furious” is “extremely angry” and one of that of “excellent” is “exceptionally good”. Therefore, by analyzing word definitions, we can obtain word intensity orders.

4.6 Conclusion

In this chapter, we built on the realization of Chapter 3 that semantic intensity scales were present to some degree in word vectors, and adjusted pretrained word vectors with inferred semantic intensity orders as well as information from WordNet and PPDB to better reflect semantic intensity scales. Adjusting word vectors with semantic intensity orders, synonyms, and antonyms altogether showed the best performance for all the three datasets we evaluated on. Using the semantic intensity orders for adjusting word vectors can help represent semantic intensities of words in vector spaces. In addition, we showed the adjustments including semantic intensity orders are not harmful for the representation of semantics in general by evaluating on SimLex-999.

This chapter evaluated the enriched word vectors only on word-level NLP tasks. Although they are still meaningful, we cannot verify yet whether the word vector enrichment is actually helpful for downstream NLP tasks. We focus on whether the enrichment can be actually necessary for certain downstream tasks in the next chapter.

CHAPTER 5: ENRICHING WORD EMBEDDINGS FOR INTENT DETECTION

In the previous chapters, we focused on word-level semantics in vector spaces through evaluating on word-level NLP tasks related to the representation of semantic intensity scales. Specifically, we showed that enriching word embeddings with semantic intensity orders as well as synonyms and antonyms is helpful for those tasks. Although those tasks are meaningful by themselves, it is not clear whether the word-level enrichment is helpful for sentence-level downstream tasks, which are more realistic and more crucial tasks. In this chapter, we focus on showing that enriching word embeddings can be helpful to improve the performance of intent detection, which is an important spoken language understanding task that is directly related to real dialog systems.

A portion of this chapter has been published as Kim et al. (2016b).

5.1 Introduction

Word embedding models such as word2vec (skip-gram and continuous bag-of-words (CBOW)) (Mikolov et al., 2013a,b), and GloVe (Pennington et al., 2014) generate word vectors based on the distributional hypothesis (Harris, 1954), 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 (Fellbaum, 1998) and the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015), which contain semantic relations among words, have been used to enrich word embeddings. In Faruqui et al. (2015), each word vector is adjusted to be in the middle between its initial vector and the average of its synonymous words. In Mrkšić et al. (2016), 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 Ono et al. (2015); Pham et al. (2015); Liu et al. (2015); Nguyen et al. (2016), 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 (Kim et al., 2016a). Hypernym and hyponym relations can also be used to enrich word embeddings regarding semantic hierarchies (Liu et al., 2015; Fu et al., 2014).

However, these enrichment approaches showed improved performance only on word-level tasks such as word similarity (Faruqui et al., 2015; Mrkšić et al., 2016; Pham et al., 2015), antonym detection (Ono et al., 2015; Pham et al., 2015; Chen et al., 2015; Nguyen et al., 2016), semantic hierarchies (Liu et al., 2015; Fu et al., 2014), and semantic scale inference (Kim et al., 2016a). More critical language understanding tasks such as intent detection and slot filling in spoken language understanding (Tur and de Mori, 2011) require dealing with phrases and sentences.

Recently, gated recurrent neural network models such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) 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 (Ravuri and Stolcke, 2015; Liu and Lane, 2016), bidirectional LSTM (BLSTM) (Hakkani-Tür et al., 2016), or BLSTM with attention mechanism (Zhang and Wang, 2016). 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 chapter, we extend the word embedding enrichment in Mrkšić et al. (2016) by incorporating the enrichment with related words from Macmillan Dictionary, and build a BLSTM on top of the enriched word embeddings. By evaluating on a publicly available language understanding benchmark corpus, ATIS, and a dataset of real user transactions 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.

5.2 Enriching word embeddings

As a baseline semantic representations, we start with 200 dimensional GloVe word vectors, which showed competitive performance in various NLP tasks (Ghannay et al., 2016), as the baseline word vectors, and enrich them with the enrichment method based on Mrkšić et al. (2016). 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. The first three functions are the same as the functions with the same names in Chapter 4.

5.2.1 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:

$$AF(V) = \sum_{(u,w) \in A} \tau\left(\cos\left(v_u, v_w\right)\right), \tag{5.1}$$

where $\tau(x) = \max(0, x)$, 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.

5.2.2 Enriching with synonyms

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

$$SC(V) = \sum_{(u,w) \in S} \tau\left(\delta - \cos\left(v_u, v_w\right)\right), \tag{5.2}$$

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.

5.2.3 Regularizing by keeping the similarity to the initial neighboring words

If we adjust word vectors too much, we may lose information about their distributional semantics coming from the initial word vectors, leading to a degradation of 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:

$$KN\left(V, V^0\right) = \sum_{i=1}^{N} \sum_{j \in N(i)} \tau\left(\cos\left(v_i, v_j\right) - \cos\left(v_i^0, v_j^0\right)\right), \tag{5.3}$$

where V^0 is the initial vocabulary matrix, N is the vocabulary size, and $N(i)$ 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:

$$C\left(V, V^0\right) = AF(V) + SC(V) + KN\left(V, V^0\right) \tag{5.4}$$

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

5.2.4 Enriching with clusters of related words

The objective functions described so far are from Mrkšić et al. (2016) 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[17] is higher than or equal to 0 using:

$$RC(V) = \sum_{(u,w)\in R} \tau\left(\delta - \cos\left(v_u, v_w\right)\right), \tag{5.5}$$

where R is the set of related words and δ is 0. Different from Equation (5.2), 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 nouns are important content words in intent detection, we tried both cases of using related verbs and nouns and using all the related words.

5.3 Intent detection

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

$$y' = \arg\max_{y} p\left(y|w_1, ..., w_n\right), \tag{5.6}$$

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 BLSTM, where we train an end-to-end model that jointly learns the slot and intent classes from the training data. Figure 5.1 shows the architecture of the model.

In Figure 5.1, 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

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

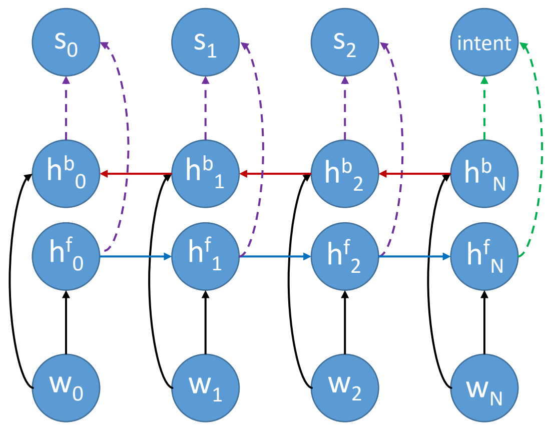


Figure 5.1: The BLSTM architecture used in this chapter. w_i, h_i^f, h_i^b, and s_i denote the word embedding layer, the forward hidden unit layer, the backward hidden unit layer, and the slot output layer at i-th time step. $intent$ is the intent output layer of the current sentence.

vectors after each update and we insert a dropout layer (Pham et al., 2014) with drop rate 0.5 between hidden layers and the output layers.

In Hakkani-Tür et al. (2016), 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 Hakkani-Tür et al. (2016), 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, in contrast to the models in Hakkani-Tür et al. (2016), 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 Hakkani-Tür et al. (2016), 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 reduced while those from intent detection errors are not changed, 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.

5.4 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.

5.4.1 Evaluation datasets

We evaluate intent detection on ATIS and Places from Microsoft Cortana. Table 5.1 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 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.

	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 5.1: 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 5.2: The numbers of adjusted words by different semantic lexicons.

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 5.1).

5.4.2 Semantic lexicons for enriching word embeddings

As described in Section 5.2, we enrich word embeddings with WordNet antonyms, PPDB antonyms, PPDB synonyms, and related words from Macmillan Dictionary. Table 5.2 shows how many words in the vocabulary were adjusted by specific semantic lexicons for each training set.

5.4.3 Experiment results

On top of the enriched word embeddings, we build a BLSTM model for the joint intent detection and slot filling. We optimize the model with ADAM[18] (Kingma and Ba, 2015). 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 5.3 and 5.4 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

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

	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 5.3: Intent detection accuracies on ATIS, varying the coverage of the training set to simulate low-resource settings.

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. For ATIS, we used off-the-shelf GloVe vectors[19] that are trained with 2014 Wikipedia dump and English Gigaword Fifth Edition, 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

[19]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 5.4: Intent detection accuracies on Places, varying the coverage of the training set similar to Table 5.3.

and Having a Baby," and so on. These belongings can be represented by setting 1 to the corresponding elements in a vector representing 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 Mrkšić et al. (2016) 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 5.3 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 mixed because the two types of word vector movements (enrichments and updates during the training) can interfere each other, which causes overall degradation of the word vector placements.

Table 5.4 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.

5.5 Future work

One issue of using enriched word embeddings as the initial word embeddings is that the enrichment can be lost during the training of the downstream task when allowing the modification of word embeddings. To address this issue, we can jointly enrich word embeddings with semantic lexicons along with the main target objectives. Table 5.5 shows the intent detection accuracies for ATIS when only 10 % of the training examples are used and enriching word embeddings were jointly done during the model training. In the table, the fourth column represents the accuracies when both model training and enriching word

Accuracy (%)	Fixing word embs	Updating word embs	Updating & enriching word embs	Updating & enriching word embs only for last 10 epochs
GloVe	83.99	**87.23**	**87.23**	86.00
Enrich w/ syns & ants	88.69	84.66	84.32	**89.14**
Enrch w/ all related words	88.91	86.56	87.68	**89.25**
Enrich w/ syns & ants & all related words	88.47	85.55	85.44	**89.14**

Table 5.5: Intent detection accuracies on ATIS when only 10% of the training examples are used and word embeddings are jointly trained during the model training.

embeddings are proceeded, and the last column is when word embedding modification is made only during the last ten epochs of the model training. We can see that the model updating and word embedding enrichment for all the epochs is not helpful compared to fixing word embeddings when only a few training examples are used. However, if we properly control the degree of word embedding updates, we can improve the performance further as shown in the last column. This experiment shows the possibility that joint model training and the enrichment can improve the performance of downstream tasks when given training sets are small. We need more extensive studies in this direction to figure out what would be the best approach for the performance improvement when training sets are small and given certain knowledge resources.

Another direction is trying more complex models such as stacked BLSTM (Graves et al., 2013) and LSTM with Memory Networks (Weston et al., 2015; Sukhbaatar et al., 2015). We can also try using Convolutional Neural Networks on top of word vector outputs or LSTM outputs, which have been shown effective for text classification tasks (Kim, 2014). 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. Also, as Mrkšić et al. (2017) showed that better word embeddings can help improve the performance of dialog state tracking, which is another downstream task related to spoken language understanding, we should try the enriched word embeddings on other downstream tasks.

Lastly, we can also try different lexicons such as hypernym/hyponym relations and other knowledge base entries for enriching word embeddings.

5.6 Conclusion

In this chapter, 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 BLSTM 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 (Mrkšić et al., 2016)'s work, we have 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.

In this chapter, we focused on word-level enrichment with linguistic resources for intent detection, which is a sentence-level downstream task. However, the word-level enrichment can only improve the word embedding layer of the model for a downstream task and it cannot exploit sentence-level linguistic knowledge or models for improving the target task. In the following chapters, we discuss sentence-level knowledge transfer for improving downstream tasks.

CHAPTER 6: CROSS-DOMAIN TRANSFER LEARNING WITH MULTI-SOURCE DOMAIN-ADVERSARIAL TRAINING FOR SLOT-FILLING

Previous chapters focused on word embeddings and the enrichment with word-level linguistic resources. Although our methods showed improvements for certain NLP tasks, since the knowledge transfer is limited to be done only in word-level, we could not utilize sentence-level knowledge or models for other related tasks to improve the target task performance. In this chapter, we introduce a sentence-level inductive transfer learning method, where datasets for source domains are available and the objective is to improving the performance of slot-filling for target domains.

6.1 Introduction

Due to recent boom of spoken dialog systems such as Amazon Alexa, Google Assistant, Apple Siri, and Microsoft Cortana, spoken language understanding (SLU), which figures out and processes user's goals, has been an important research topic.

Slot filling is a sequence tagging task, where an IOB (inside, outside, and begin)-formatted tag is predicted for each word of given sentences. It is one of the three major tasks in SLU, where the other tasks are intent detection and domain classification (Tur and de Mori, 2011). For example, a user could say *"show me movies with Brad Pitt"*. Then the goal of slot filling task would be to tag *"Brad Pitt"* as `actor`, and *"movie"* as `contents_type`.

Although the demand of dealing with new domains in SLU tasks is increasing, it is expensive to collect a large number of new training examples. Transfer learning methods can utilize datasets from other domains to help improve the performance of the target domain reducing the data insufficiency issue (Kim et al., 2015c).

There have been many transfer learning methods utilized for SLU tasks. Tur (2006) introduced multi-task learning for intent classification of spoken utterances. Li et al. (2011) applied multi-task learning for semi-Markov CRFs for slot filling. In their work, they exploit that some of the slots are shared among different domains. Kim et al. (2015c) used hidden unit conditional random field (HUCRF) models with label embeddings induced by Canonical Correlation Analysis (CCA) (Hotelling, 1936) so that datasets from different domains with different label spaces can be used for knowledge transfer. Jaech et al. (2016) and Kim et al. (2016c) utilized LSTM networks for transfer learning. Jaech et al. (2016) used a single LSTM layer to cover datasets from all the domains, and K feed-forward softmax output layers, where the k-th output layer is used for dealing with the dataset from the k-th domain. Kim et al. (2016c) used a single common LSTM, K private LSTMs and K feed-forward layers. The common LSTM is trained with datasets from all the domains so that domain-general representations can be learned, while the k-th private LSTM and the k-th feed-forward layer is trained only using datasets from the k-th domain to separately learn domain-specific representations. They achieved improved performance on multi-task learning with datasets from 17 domains. In SLU cases, using datasets across multiple domains for training the common LSTM is desirable since out-of-vocabulary problem can be reduced and the common LSTM outputs can be regularized to be more general rather than specific to inputs from certain domains (Jaech et al., 2016; Kim et al., 2016c).

Inspired by Kim et al. (2016c)'s multi-task slot-filling model, our model utilizes a common BLSTM for representing domain-generic information, which allows knowledge transfer from other domains, and private BLSTMs for representing domain-specific information. The common BLSTM is additionally encouraged to be domain-invariant with domain-adversarial training so that the domain-general representations to be more compatible among different domains. Specifically, focusing on the cases with multiple domains, we introduce cross-domain transfer learning with multi-source domain-adversarial training, which transfer knowledge from multiple source domains. Multi-source domain-adversarial training uses K binary domain classifiers when there are K source domains so that the input data from a specific source domain is indistinguishable with the input from the target domain and vice versa. Also, we apply a representation separation technique from unsupervised domain adaptation for image recognition (Bousmalis et al., 2016) to transfer learning for slot filling.

We evaluate our proposed model focusing on the case with datasets from two source domains for transferring to a target domain.

6.2 Model

6.2.1 Cross-domain training

Figure 6.1 shows the overall architecture of the proposed model. The baseline slot-filling model is similar to Kim et al. (2016c)'s model, and it corresponds to having only a word embedding layer, a common BLSTM, private BLSTMs and a softmax output in Figure 6.1.

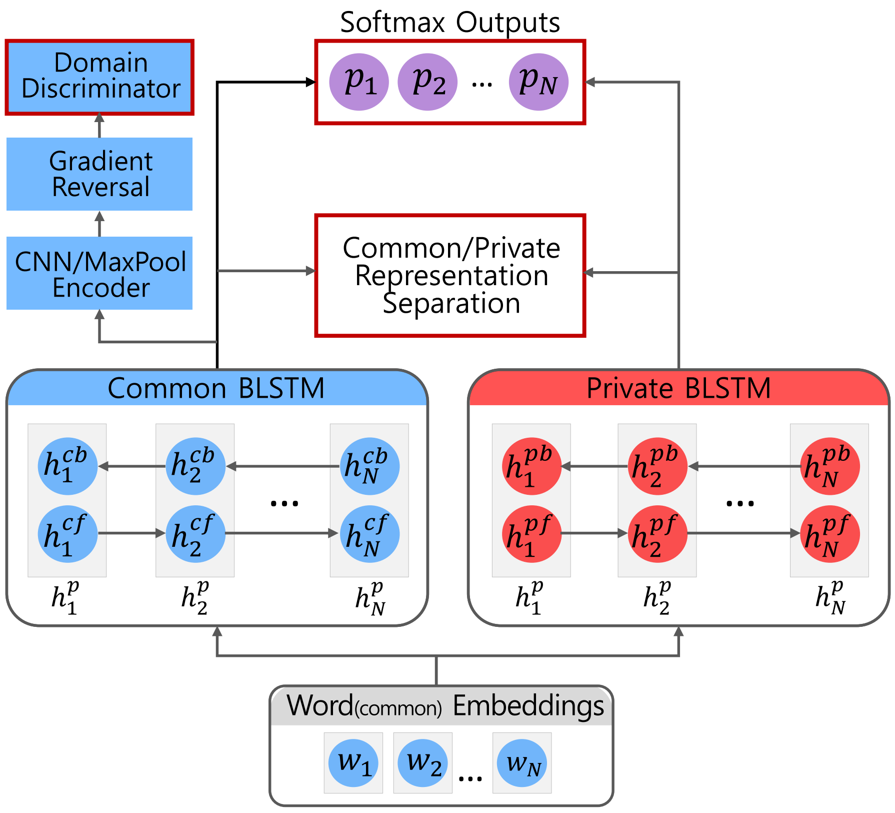


Figure 6.1: The architecture of cross-domain transfer learning model for slot-filling.

For the cross-domain transfer learning, the word embeddings and the common BLSTM are shared for all the given domains while private BLSTMs and softmax output layers have different parameters for different domains.

The outputs of the common BLSTM and the private BLSTM of the current domain are added to be used as the input to the softmax layer to predict the slots of given word sequences. The loss function of the slot-filling is formulate as:

$$\mathcal{L}_p = -\sum_{i=1}^{S}\sum_{j=1}^{N} p_{i,j} \log\left(\hat{p}_{i,j}\right), \tag{6.1}$$

where S is the number of sentences in the current minibatch, N is the number of words in the current sentence, $p_{i,j}$ is the label of the j-th slot of the i-th sentence in the minibatch, and $\hat{p}_{i,j}$ is the predicted slot. In addition to this main objective, two more objectives for improving the transfer learning are described in the following subsections.

6.2.2 Domain-adversarial training

We encourage the outputs of the common BLSTM to be domain-invariant by using domain-adversarial training inspired by domain-adversarial training (Ganin et al., 2016; Bousmalis et al., 2016). First, we encode a BLSTM output sequence as a single vector using a CNN/MaxPool encoder, which is implemented the same as a CNN for text classification (Kim, 2014). The encoder is with three convolution filters whose sizes are 3, 4, and 5. For each filter, we pass the BLSTM output sequence as the input sequence and obtain a single vector from the filter output by using max pooling, and then $tanh$ activation function is used for transforming the vector. Then, the vector outputs of the three filters are concatenated and forwarded to the domain discriminator through the gradient reversal layer. The discriminator is implemented as a fully-connected neural network with a single hidden layer, whose activation function is Leaky ReLU (Maas et al., 2013), where we multiply 0.2 to negative input values as the outputs.

Since the gradient reversal layer is below the domain classifier, the gradients minimizing domain classification errors are passed back with opposed sign to the sentence encoder, which adversarially encourages the sentence encoder to be domain-invariant. The loss function of the domain classifier is formulated as:

$$\mathcal{L}_a = -\sum_{i=1}^{S} d_i \log \hat{d}_i, \tag{6.2}$$

where S is the number of sentences, d_i is the domain of the i-th sentence, and $\hat{d}_i$ is the softmax output of the slot-filling. Note that though the domain classifier is optimized to minimize the domain classification error, the gradient from the domain classifier is negated so that the bottom layers are trained to be domain-invariant.

Multi-source Domain-adversarial Training

To address the cases when datasets from K source domains and one target domain are given, we tried the following discriminators for the multi-class classification for multi-source domain-adversarial training.

Using a single $(K + 1)$-ary domain discriminator When we have multiple sources, we can simply use $K + 1$ class labels in a single $(K + 1)$-ary domain classifier. However, in this case, the direction of the reversed gradients when the input data are from the k-th source domain is not for making the k-th source domain to be confused with the target domain, but making the data from k-th source domain not to be classified as the k-to source domain. Therefore, updating the common BLSTM with the gradient reversal from the domain classifier does not guarantee that its outputs are getting more domain invariant.

Using K binary domain discriminators Since our goal of transfer learning is to improve the performance on target domain datasets, we encourage the inputs from source domains to be indistinguishable with the target domain. For target domain inputs, we encourage them to be indistinguishable with the source domain datasets rather than encouraging the target domain inputs to be non-target domain inputs. For this approach, we use K binary domain classifiers, where the k-th domain classifier decides whether the input belongs to the k-th source domain or the target domain. Therefore, if the current input belongs to the k-th source domain, the gradient from the classifier is reversed to confuse the input to be in the target domain. If the current input is from the target domain, the gradients from each domain classifier is reversed so that the input is less distinguishable with the source datasets. In Section 6.3, we show that this approach actually improves the performance for the case with two-sources.

6.2.3 Separating common representations and private representations

Bousmalis et al. (2016) applied an objective function that reduces the correlations between outputs of the common CNN encoder and the outputs of the private CNN encoder so that the noisy influence of domain-specific information to the domain-general information can be minimized. This was shown to be effective for unsupervised domain adaptation for image recognition tasks.

Our approach applies the separation of domain-general representations and domain-specific representations for supervised transfer learning for sequence tagging. In unsupervised domain adaptation, just removing noisy domain-specific representations is the goal of the representation separation since only the common representations are used. However, in supervised transfer learning, domain-specific representations are also used for the final prediction, thereby preserving domain-specific information is important.

Assuming that the minibatch size of the model is N and the dimensionality of each output representation is M, then the current outputs of the BLSTMs at time step t can be represented as two $N \times M$ matrices, H_t^c and H_t^p. At time step t, each row of H_t^c and each row of H_t^p represent outputs of the common BLSTM and the private BLSTM for a sentence, respectively. Then, after normalizing the matrices so that each row is with zero mean and a unit $l2$-norm, we use multiplication of the two matrices for formulating the loss function:

$$\mathcal{L}_c = ||H_t^{c\top} H_t^p||_F, \tag{6.3}$$

which is Frobenius norm of the multiplied matrix. The matrix $H_t^{c\top} H_t^p$ is the sum of cross covariance matrices over the sentences in the current minibatch.[20] Entry (i, j) of the matrix represents the sum of covariance between the values at dimension i of the common BLSTM outputs and the values at dimension j of the private BLSTM outputs.

All the three loss functions are added to be optimized altogether as:

$$\mathcal{L} = w_s \left(\mathcal{L}_p + \lambda \mathcal{L}_a + \lambda \mathcal{L}_c \right), \tag{6.4}$$

where w_s is used to give different weights to the source domain and the target domain. Since the source domain has a larger train set and we are focusing on improving the performance of the target domain, w_s is set to 1 when training the target domain. For the source domain, instead, it is set as the size of the target train set divided by the size of the source

[20]Optimizing $H_t^c H_t^{p\top}$ corresponds to encouraging soft subspace orthogonality. We also tried to optimize it but they showed worse performance in our evaluations. Bousmalis et al. (2016) described about the soft subspace orthogonality but they also optimized cross covariance matrices.

Source (S) / Target (T)	Domain	Description
S	Calendar	Set events in a calendar app
S	Places	Find location and direction for places
T	Taxi	Find and book a cab like Uber
T	Health	Monitor health condition such as Fitbit.
T	Real-estate	Find the real-estimate info like Zillow.

Table 6.1: The description of each domain.

train set. λ is gradually increased from 0 to 1 for stable training with auxiliary objectives as follows (Ganin et al., 2016):

$$\lambda = \frac{2}{1 + \exp(-10p)} - 1, \tag{6.5}$$

where p is 0 for the first epoch and linearly increases to 1 until the final epoch.

6.3 Experiments

In this section, we describe a set of experiments performed to evaluate the performance of our approach, we also consider several baselines and variants besides our primary transfer learning model.

6.3.1 Test domains and tasks

To evaluate the effectiveness of our proposed model, we apply it to a suite of six personal assistant domains with slot filling task in spoken language understanding.

Table 6.1 shows the descriptions of the domains we used for the evaluation.

The number of training sentences in each source domain dataset is around 10,000 and that of the sentences in each target domain dataset is around 1,000. The number of sentences in each of the development sets and the test sets is also around 1,000.

The datasets in different domains are not directly related, the label spaces are all different, and the numbers of training examples in the target datasets are much smaller than those of the source datasets. This is not an easy but frequently occurring scenario, where transfer learning is necessary to help dealing with new domain datasets.

Our experimental settings are rather different than previously considered settings for Bousmalis et al. (2016) in many aspects:

- *Insufficient target data:* the amount of target data is limited.
- *Variant label spaces:* The label space is invariant across all domains. Here, the label space can be different in different domains, which is a more challenging setting.
- *Imbalance:* The number of source data is bigger than that of target data.
- *Multiple source domains:* We have K source domains.

6.3.2 Implementation details

In all experiments, all the models were optimized using ADAM (Kingma and Ba, 2015)[21] with minibatches of 64 sentences. Each minibatch represents sentences with different lengths by padding the remainder of short sentences with zero. For all the evaluations, we optimized the model for total 100 epochs, and we picked the model showing the best F-score on the development set to report the score on the test set.

In the word embedding layer, the word vectors are initialized with normalized off-the-shelf 200 dimensional GloVe vectors (Pennington et al., 2014).[22] The word vector outputs are regularized with dropout rate 0.5 (Pham et al., 2014). For the consistent dropout usages,

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

[22]Available from `http://nlp.stanford.edu/data/glove.6B.zip`

Targets	Target only	JAECH16	KIM16	+adv	+sep	+adv, +sep
taxi	75.66	76.11	78.24	**78.5**	78.2	77.9
health	76.03	76.62	**81.3**	79.1	79.6	79.1
real-estate	66.15	67.41	**68.91**	67.2	68.4	67.3
average	72.61	73.38	**76.15**	74.93	75.4	74.77

Table 6.2: F-scores on transferring from single source ("calendar").

we let the dropout masks to be identical for all the word vector outputs of each sentence Gal and Ghahramani (2016). The dimensionality of each hidden output of LSTMs is 100, and the hidden outputs of both forward LSTM and backward LSTM are concatenated, thereby the output of each BLSTM for each time step is 200 dimensional.

6.3.3 Experimental setup

In testing our approach, we consider two possible transfer learning scenarios: (1) single-source, (2) two-sources. We use "calendar" dataset as the source for single-source cases, and both "calendar" and "places" as the source datasets for the cases of two-sources.

We use a vanilla model with a single LSTM and a single softmax output layer as the baseline model, which is trained with only the training set of target domains. Two of previous slot filling models (Jaech et al., 2016; Kim et al., 2016c) for transfer learning are used for comparisons, and we evaluate the models with different combinations of proposed single or multi-source domain-adversarial training and the representation separation. The evaluated models are as follows.

- Target only : Just the training examples from the target domain is used for training.
- JAECH16 : Jaech et al. (2016)'s model with a common LSTM.

- KIM16 : Kim et al. (2016c)'s model with a common LSTM and multiple private LSTMs.

- +adv : domain-adversarial training with a single $(K + 1)$-ary classifier in Section 6.2.2

- +K adv : domain-adversarial training with a K binary domain classifiers in Section 6.2.2

- +sep : denotes representation separation in Section 6.2.3

6.3.4 Results: Single source

We first consider a scenario of a single source. Table 6.2 shows test F-scores of experiments on these cases. It demonstrates that just using domain-adversarial training and the representation separation did not improve the performance over (Kim et al., 2016c). Both methods do not directly optimize the end task but act as regularization methods. For unsupervised domain adaptation, since there is no target label information, it has been shown to be helpful to just encourage the source inputs and target inputs less distinguishable. However, for supervised transfer learning, since the target performance is already on a certain level with its own labeled training set, it is not trivial to further improve the performance with just domain-adversarial training or the representation separation.

One point is that we did not tune any hyperparameters to increase the scores. Therefore, using domain-adversarial training and representation separation might be helpful after finding proper hyperparameters with valid sets, but that is not our focus in this chapter.

Targets	Target only	JAECH16	KIM16	+adv	+sep	+adv, +sep	+K adv	+K adv, +sep
taxi	75.66	76.0	82.79	72.3	80.3	72.3	**83.2**	82.4
health	76.03	76.6	81.4	77.5	81.2	77.6	81.5	**81.8**
real-estate	66.15	70.8	78.8	75.5	77.9	76.6	79.4	**79.5**
average	72.61	74.51	81	75.1	79.8	75.5	**81.37**	81.23

Table 6.3: F-scores on the case of transferring from two sources ("calendar" and "places").

6.3.5 Results: Two sources

For single source domain cases in the previous section, we observed that it may not be helpful to simply apply domain-adversarial training or the representation separation for supervised transfer learning. However, as aforementioned in Section 5.1, it is important to use datasets from multiple source domains for more effective transfer learning on SLU tasks.

Table 6.3 shows test F-scores of experiments on two-sources cases. Compared to the scores using a single source in Table 6.2, given the same model, we could observe that the performance is improved in general. However, just using domain-adversarial training with a single ternary (two for sources and one for a target) domain classifier was still worse than using (Kim et al., 2016c)'s model. As described in Section 6.2.2, this method of domain-adversarial training may not encourage source inputs and target inputs to be indistinguishable.

However, using domain-adversarial training with two binary domain classifiers, we could observe the best performance on average. In this case, as the inputs from the k-th source domain is directly encouraged to be indistinguishable with the target inputs by using the k-th domain classifier, and the target inputs are also encouraged to be indistinguishable with inputs from source domains, domain-adversarial training can be effective for improving the target domain performance. Since utilizing multiple source domains can be crucial for transfer learning, good performance with multiple source domains is more meaningful than the performance with a single source domain.

6.4 Conclusion

In this chapter, we proposed a cross-domain transfer learning method separately representing domain-general information and domain-specific information applied to slot filling. We showed the effectiveness of encouraging domain-general representations for all the domains to be domain-invariant and domain-specific representations to be less correlated to the domain-general representations on transfer learning for slot filling tasks. We demonstrated that this approach can improve the performance of transfer learning for sequence models given multiple domains with different label spaces. Specifically, our method showed the best performance on transfer learning scenarios using two source domains. Since using multiple domains is important for transfer learning on SLU tasks, our model can be applicable for realistic slot filling scenarios, where target domain datasets are small but multiple source domain datasets are available.

So far, we focused on inductive transfer learning, where the input spaces of the source domains and the target domains are the same. In the next chapter, we discuss transductive transfer learning, where knowledge transfer is available even when the source input spaces and target input spaces are different.

CHAPTER 7: CROSS-LINGUAL TRANSFER LEARNING FOR POS TAGGING WITHOUT CROSS-LINGUAL RESOURCES

In the previous chapters, we only discussed inductive transfer learning, where the input spaces of the source domains and the target domains are the same. Those inductive transfer learning models have been shown to be effective to improve the performance of many NLP tasks within a single language. However, when the source languages and the target languages are different, we cannot directly utilize inductive transfer learning methods since the source input spaces and the target input spaces are not equivalent. In this chapter, we propose a transductive transfer learning method,[23] which allows knowledge transfer even when the sources and the targets have different input spaces, to improve the performance of the target tasks when the source languages and the target languages are different.

A portion of this chapter has been accepted for publication as Kim et al. (2017a).

7.1 Introduction

BLSTM-based models (Graves and Schmidhuber, 2005), along with word embeddings and character embeddings, have shown competitive performance on Part-of-Speech (POS) tagging given sufficient amount of training examples (Ling et al., 2015; Lample et al., 2016; Plank et al., 2016; Yang et al., 2017).

Given insufficient training examples, we can improve the POS tagging performance by cross-lingual POS tagging, which exploits affluent POS tagging corpora from other source

[23] Since the character embedding is shared, our model is also a partially inductive transfer learning method.

languages. It usually requires linguistic knowledge or resources about the relation between the source language and the target language such as parallel corpora (Täckström et al., 2013; Duong et al., 2013; Kim et al., 2015b; Zhang et al., 2016a), morphological analyses (Hana et al., 2004), dictionaries (Wisniewski et al., 2014), and gaze features (Barrett et al., 2016).

Given no linguistic resources between the source language and the target language, transfer learning methods can be utilized instead. Transfer learning for cross-lingual cases is a type of transductive transfer learning, where the input domains of the source and the target are different (Pan and Yang, 2010) since each language has its own vocabulary space. When the input space is the same, lower layers of hierarchical models can be shared for knowledge transfer (Collobert et al., 2011).

Yang et al. (2017) categorized three types of inductive transfer learning for sequence tagging as Figure 7.1.[24] In Figure 7.1, Yang et al. (2017) used shared character embeddings for different languages as a cross-lingual transfer method while using different word embeddings for different languages. Although the approach showed improved performance on named entity recognition, it is limited to character-level representation transfer and it is not applicable for knowledge transfer between languages without overlapped alphabets.

We propose a cross-lingual transfer learning model for POS tagging requiring no cross-lingual resources, where knowledge transfer is made in the BLSTM layers on top of word embeddings and character embeddings. Similar to the approach using a common BLSTM and different private BLSTMs for different domains in Chapter 6, we utilize a common BLSTM for representing language-generic information, which allows knowledge transfer from other languages, and private BLSTMs for representing language-specific information.

[24]Figure 7.1 assumes that the models utilize CRF on top of the BLSTM layers, but we simply used softmax output layers instead.

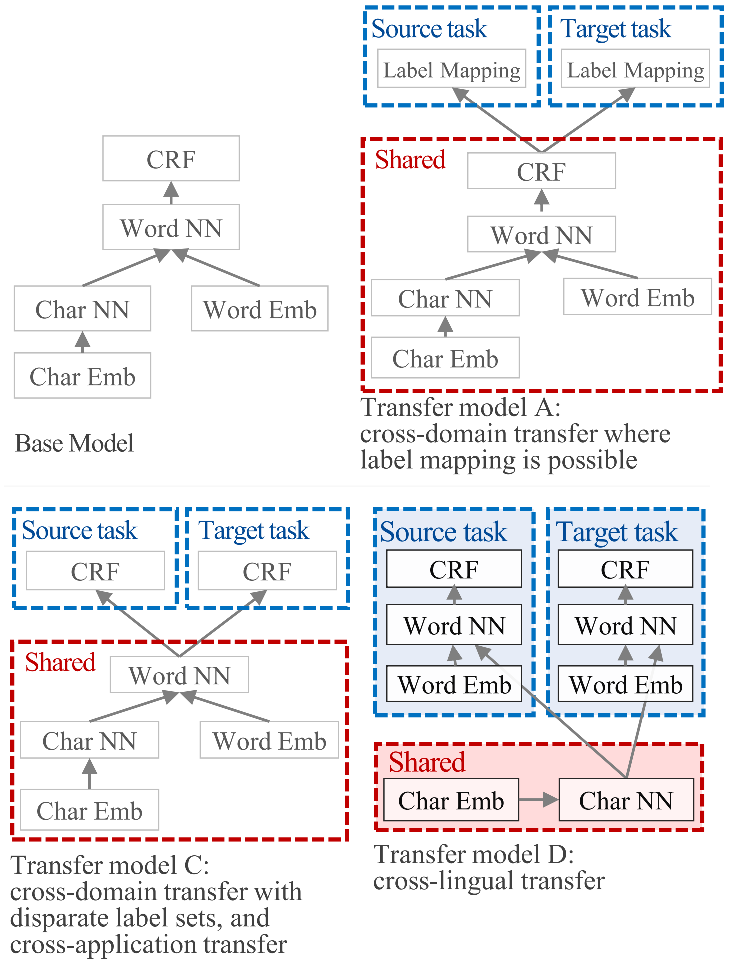


Figure 7.1: A baseline architecture and transfer learning models for sequence tagging (Yang et al., 2017).

The common BLSTM is additionally encouraged to be language-agnostic with language-adversarial training (Chen et al., 2017) so that the language-general representations to be more compatible among different languages.

Evaluating on POS datasets from 14 different target languages with English as the source language in the Universal Dependencies corpus 1.4 (Nivre et al., 2016), the proposed model showed significantly better performance when the source language and the target language are in the same language family, and competitive performance when the language families are different.

7.2 Model

7.2.1 Cross-lingual training

Figure 7.2 shows the overall architecture of the proposed model. The baseline POS tagging model is similar to Plank et al. (2016)'s model, and it corresponds to having only word+char embeddings, private BLSTM, and softmax output in Figure 7.2. Given an input word sequence, a BLSTM is used for the character sequence of each word, where the outputs of the ends of the character sequences from the forward LSTM and the backward LSTM are concatenated to the word vector of the current word to supplement the word representation. These serve as an input to a BLSTM, and an output layer are used for POS tag prediction.

For the cross-lingual transfer learning, the character embedding, the BLSTM with the character embedding (Yang et al., 2017),[25] and the common BLSTM are shared for all the given languages while word embeddings and private BLSTMs have different parameters for different languages.

[25]We also tried isolated character-level modules but the overall performance was worse.

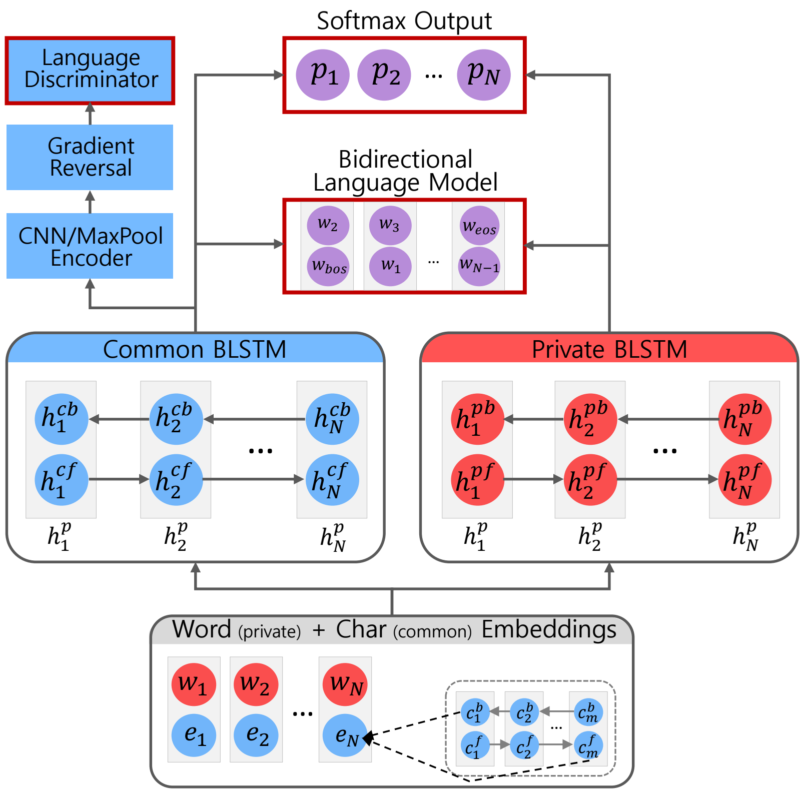

Language Discriminator
Gradient Reversal
CNN/MaxPool Encoder
Softmax Output
p_1 p_2 ... p_N
Bidirectional Language Model
w_2 w_3 w_{eos} w_{bos} w_1 ... w_{N-1}
Common BLSTM
h_1^{cb} h_2^{cb} h_N^{cb} h_1^{cf} h_2^{cf} h_N^{cf}
h_1^p h_2^p h_N^p
Private BLSTM
h_1^{pb} h_2^{pb} h_N^{pb} h_1^{pf} h_2^{pf} h_N^{pf}
h_1^p h_2^p h_N^p
Word (private) + Char (common) Embeddings
w_1 w_2 w_N e_1 e_2 e_N
c_1^b c_2^b c_m^b c_1^f c_2^f c_m^f

Figure 7.2: Model architecture: blue modules share parameters for all the languages and red modules have different parameters for different languages. w_i and e_i denote the i-th word vector and the i-th character vector composition, respectively. h_i^{cf}, h_i^{cb}, h_i^{pf}, and h_i^{pb} denote the i-th hidden outputs of the forward common LSTM, the backward common LSTM, the forward private LSTM, and the backward private LSTM, respectively. h_i^c and h_i^p denote the concatenated output of the common BLSTM and the private BLSTM, respectively. Violet circles represent target labels that are predicted with different parameters for different languages, where the inputs are output summation of the common BLSTM and the private BLSTM. The model is trained with three objectives denoted with red boxes.

The outputs of the common BLSTM and the private BLSTM of the current language are summed to be used as the input to the softmax layer to predict the POS tags of given word sequences. The loss function of the POS tagging can be formulate as:

$$\mathcal{L}_p = -\sum_{i=1}^{S}\sum_{j=1}^{N} p_{i,j} \log\left(\hat{p}_{i,j}\right), \tag{7.1}$$

where S is the number of sentences in the current minibatch, N is the number of words in the current sentence, $p_{i,j}$ is the label of the j-th tag of the i-th sentence in the minibatch, and $\hat{p}_{i,j}$ is the predicted tag. In addition to this main objective, two more objectives for improving the transfer learning are described in the following subsections.

7.2.2 Language-adversarial training

Similarly to domain-adversarial training in Section 6.2.2, we use language-adversarial training to encourage the common BLSTM outputs to be language-agnostic.[26] The loss function of the language classifier is formulated as:

$$\mathcal{L}_a = -\sum_{i=1}^{S} l_i \log \hat{l}_i, \tag{7.2}$$

where S is the number of sentences, l_i is the language of the i-th sentence, and $\hat{l}_i$ is the softmax output of the tagging. Note that though the language classifier is optimized to minimize the language classification error, the gradient from the language classifier is negated so that the bottom layers are trained to be language-agnostic.

7.2.3 Bidirectional language modeling

Rei (2017) showed the effectiveness of the bidirectional language modeling objective, where each time step of the forward LSTM outputs predicts the word of the next time step,

[26]We focus on the single source language case.

and each of the backward LSTM outputs predicts the previous word. Figure 7.3 shows an example how bidirectional language modeling objective is used along with the tagging objective. If the current sentence is “I am happy”, the forward LSTM predicts “am happy <eos>” and the backward LSTM predicts “<bos> I am”.

This objective encourages the BLSTM layers and the embedding layers to learn linguistically general-purpose representations, which are also useful for specific downstream tasks (Rei, 2017). It is more effective when a large number of unlabeled training examples are available. The objective is also used not to lose too much information from adversarial training, where reconstruction objectives were used instead in previous work (Bousmalis et al., 2016; Zhang et al., 2017; Kim et al., 2017b).

We adopted the bidirectional language modeling objective, where the sum of the common BLSTM and the private BLSTM is used as the input to the language modeling module. It can be formulated as:

$$\mathcal{L}_l = -\sum_{i=1}^{S}\sum_{j=1}^{N} \log\left(P\left(w_{j+1}|f_j\right)\right) + \log\left(P\left(w_{j-1}|b_j\right)\right), \tag{7.3}$$

where f_j and b_j represent the j-th outputs of the forward direction and the backward direction, respectively, given the output sum of the common BLSTM and the private BLSTM.

All the three loss functions are added to be optimized altogether as:

$$\mathcal{L} = w_s\left(\mathcal{L}_p + \lambda\mathcal{L}_a + \lambda\mathcal{L}_l\right), \tag{7.4}$$

where w_s and λ are decided the same as Equation 6.4.

7.3 Experiments

For the evaluation, we used the POS datasets from 14 different languages in Universal Dependencies corpus 1.4 (Nivre et al., 2016). We used English as the source language,

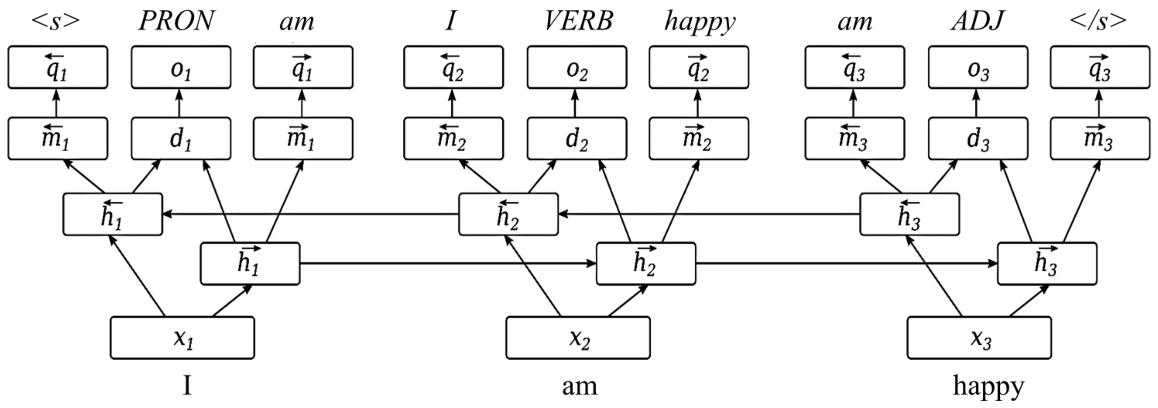


Figure 7.3: Bidirectional language modeling on top of the BLSTM outputs along with POS tagging (Rei, 2017). (The image is borrowed from Rei (2017) but the input and the output texts are modified.)

Language family	Language	# POS tags	# train sents	# dev sents	# test sents
Germanic	English	17	12,543	2,002	2,077
	Swedish	16	4,303	504	1,219
	Danish	17	4,868	322	322
	Dutch	16	13,000	349	386
	German	16	14,118	799	977
Slavic	Slovenian	16	6,471	735	790
	Polish	15	6,800	700	727
	Slovak	15	8,483	1,060	1,061
	Bulgarian	16	8,907	1,115	1,116
Romance	Romanian	17	7,141	1,191	1,191
	Portuguese	18	8,800	271	288
	Italian	18	12,837	489	489
	Spanish	17	14,187	1,552	274
Indo-Iranian	Persian	16	4,798	599	600
Uralic	Hungarian	16	1,433	179	188

Table 7.1: The source and the 14 target datasets for POS tagging from the Universal Dependency corpus.

which is with 12,543 training sentences.[27] We chose datasets with 1k to 14k training sentences. In determining our selection of the target languages, we wanted to sample languages across language families. Since we used English as the source language, we hypothesized that Germanic languages are more likely to benefit from the transfer learning. As a contrast, we also picked two other language families (with 4 languages a piece), as well as two additional languages (Hungarian, Persian.) The number of tag labels differs for each language from 15 to 18 though most of them are overlapped within the languages. Table 7.1 shows the description of evaluated datasets.

Table 7.2 shows the POS tagging accuracies of different transfer learning models when we limited the number of training sentences of the target languages to be the same as 1,280 for fair comparison among different languages. 1,280 is chosen since it is adequately less than the smallest number of training sentences in the target languages[28] to see performance difference comparing to the case using all the sentences. The remainder training examples of the target languages are still used for both language-adversarial training and bidirectional language modeling since the objectives do not require tag labels. Training with only the train sets in the target languages (p) showed 91.61% on average. When bidirectional language modeling objective is used (p, l), the accuracies were significantly increased to 92.82% on average. Therefore, we used the bidirectional language modeling for all the transfer learning evaluations.

With transfer learning, the three cases of using only the private BLSTMs (p), using only the common BLSTM (c), and using both (p, c) were evaluated. They showed better average accuracies than target only cases, but they showed mixed results. However, our

[27]The accuracies of English POS tagging are 94.01 and 94.33 for models without the bidirectional language modeling and with it, respectively.

[28]Hungarian has the fewest 1,433 training sentences.

		Target only		Source (English) $\rightarrow$ Target				
Language Family	Language	p	p,l	p,l	c,l	p,c,l	c,l+a	p,c,l+a
	Swedish	93.26	94.31	94.39	94.36	94.51	94.38	**94.63***
	Danish	92.13	93.41	93.76	93.34	94.05	93.74	**94.26***
Germanic	Dutch	83.24	84.73	84.92	85.20	84.85	84.99	**85.83***
	German	89.27	90.69	90.40	90.06	90.01	90.14	**90.71**
	Avg	89.47	90.78	90.87	90.74	90.86	90.82	**91.36**
	Slovenian	93.06	93.79	94.06	93.83	**94.20**	93.93	$\underline{94.06}$
	Polish	91.30	91.30	**92.11**	91.69	91.86	91.77	**92.11***
Slavic	Slovak	86.53	89.56	89.88	90.11	89.98	**90.40**	90.01*
	Bulgarian	93.45	95.27	95.50	95.33	95.52	95.25	**95.65***
	Avg	91.09	92.48	92.89	92.74	92.89	92.84	**92.95**
	Romanian	93.20	94.09	94.17	**94.22**	94.05	93.91	$\underline{94.20}$
	Portuguese	94.23	95.18	95.15	95.42	**95.55**	95.36	$\underline{95.51}$
Romance	Italian	93.80	**95.95**	95.61	95.79	95.84	95.70	$\underline{95.92}$
	Spanish	91.94	93.34	93.31	93.34	93.29	92.94	**93.44**
	Avg	93.29	94.64	94.56	94.69	94.68	94.48	**94.77**
Indo-Iranian	Persian	93.91	94.63	94.79	94.68	94.78	94.49	**94.83**
Uralic	Hungarian	93.20	93.27	94.66	94.40	**94.69**	94.29	94.45*
	Total Avg	91.61	92.82	93.05	92.98	93.08	92.95	**93.26**

Table 7.2: POS tagging accuracies (%) when setting the numbers of the tag-labeled training examples of the target languages to be the same as 1,280 (All the training examples are still used for the language modeling and the adversarial training.) p: using private BLSTMs, c: using common BLSTM, l: bidirectional language modeling objectives, a: language-adversarial training. (Each score with $*$ denotes that the difference between the score and the corresponding score of "Target only (p, l)" case is statistically significant, and each underlined score denotes that the difference between the score and the highest score of the other models is statistically insignificant with McNemar's χ^2 test with p-value < 0.05.)

proposed model $(p, c, l + a)$, which utilizes both the private BLSTMs and the common BLSTM with language-adversarial training, showed the highest average score, 93.26%. For all the Germanic languages, where the source language also belongs to, the accuracies are significantly higher than those of other transfer learning models. For the languages belonging to Slavic, Romance, or Indo-Iranian, our model shows competitive performance with the highest average accuracies among the compared models. Since languages in the same family are more likely to be similar and compatible, it is expected that the gain from the knowledge transfer to the languages in the same family to be higher than transferring to the languages in different families, which was shown in the results.

This shows that utilizing both language-general representations that are encouraged to be more language-agnostic and language-specific representations effectively helps improve the POS tagging performance with transfer learning.

Tables 7.3 and 7.4 show the results when using 320 and 32 tag-labeled training sentences, respectively. For both cases, transfer learning methods show better accuracies than target-only approaches on average. However, the performance gain is weakened compared to using 1,280 labeled training sentences and there are some mixed results. In several cases, just utilizing private BLSTMs without the common BLSTM showed better accuracies than utilizing the common BLSTM. When using 32 labeled sentences, which is an extremely low-resourced setting, not using the common BLSTM even showed better performance than using it on average. The main reason would be that we are not given a sufficient number of labeled training sentences to train both the common BLSTM and the private BLSTMs. In this case, just having private BLSTMs without the common BLSTM can show better performance.

		Target only		Source (English) → Target				
Language Family	Language	p	p,l	p,l	c,l	p,c,l	c,l+a	p,c,l+a
Germanic	Swedish	87.43	90.49	**91.02**	90.45	90.48	90.72	90.70
	Danish	86.42	90.00	90.74	90.69	90.02	90.16	**90.79**
	Dutch	76.76	82.24	**82.61**	82.46	82.10	82.58	82.15
	German	86.25	88.95	89.10	88.69	88.93	88.08	**89.68**
	Avg	84.22	87.92	**88.37**	88.07	87.88	87.88	88.33
Slavic	Slovenian	87.02	89.97	90.29	90.00	90.32	89.58	**90.59**
	Polish	82.10	84.13	85.21	85.41	85.30	85.46	**85.50**
	Slovak	76.22	81.03	82.95	**83.40**	82.68	82.70	83.17
	Bulgarian	87.32	**92.81**	92.68	92.07	92.30	92.20	92.39
	Avg	83.16	86.98	87.78	87.72	87.65	87.48	**87.91**
Romance	Romanian	88.67	**91.44**	**91.44**	90.87	91.22	90.85	91.37
	Portuguese	90.66	93.73	93.55	93.90	93.81	93.58	**94.20**
	Italian	89.78	93.99	93.82	93.27	93.46	93.51	**94.00**
	Spanish	85.91	**91.07**	90.59	90.59	**91.07**	90.17	90.88
	Avg	88.76	92.56	92.35	92.16	92.39	92.03	**92.61**
Indo-Iranian	Persian	90.64	**92.40**	91.98	91.97	92.12	92.18	91.83
Uralic	Hungarian	89.14	90.65	91.45	91.48	90.91	**91.52**	90.72
	Total Avg	86.02	89.49	89.82	89.66	89.62	89.52	**89.86**

Table 7.3: POS tagging accuracies (%) with 320 tag-labeled training examples for each target language. All the training examples are still used for the other objectives.

		Target only		Source (English) → Target				
Language Family	Language	p	p,l	p,l	c,l	p,c,l	c,l+a	p,c,l+a
Germanic	Swedish	60.85	81.08	**83.85**	82.19	82.43	83.00	82.52
	Danish	56.42	83.67	84.59	84.60	84.11	83.51	**84.72**
	Dutch	59.59	75.85	**77.34**	77.20	76.42	76.35	76.18
	German	61.58	81.88	82.09	80.92	**82.59**	82.34	81.79
	Avg	59.61	80.62	**81.97**	81.23	81.39	81.30	81.30
Slavic	Slovenian	58.15	80.90	80.91	**81.97**	80.82	80.51	80.80
	Polish	55.95	72.69	74.84	**76.78**	73.51	75.11	73.54
	Slovak	51.69	68.80	71.14	**75.15**	72.79	74.12	72.14
	Bulgarian	62.58	84.50	84.58	84.49	83.37	**85.21**	83.94
	Avg	57.09	76.72	77.87	**79.60**	77.62	78.74	77.60
Romance	Romanian	61.44	83.79	**84.10**	83.87	82.58	83.32	83.79
	Portuguese	66.81	88.74	89.20	88.61	**89.40**	88.94	89.19
	Italian	69.79	87.60	87.57	86.55	**87.92**	86.61	87.21
	Spanish	57.74	82.50	82.92	81.99	82.19	81.71	**83.33**
	Avg	63.94	85.66	**85.95**	85.26	85.52	85.14	85.88
Indo-Iranian	Persian	72.11	**85.76**	85.49	84.53	85.11	84.05	84.41
Uralic	Hungarian	60.50	78.42	**80.31**	78.18	79.67	79.48	79.93
	Total Avg	61.09	81.16	**82.07**	81.93	81.64	81.73	81.68

Table 7.4: POS tagging accuracies (%) with 32 tag-labeled training examples for each target language. All the training examples are still used for the other objectives.

We also evaluated the opposite cases, which use all the tag-labeled sentences in the train set of target languages, and they showed mixed results. For example, the accuracy of German with the target only model is 93.31% while that of the proposed model is 93.04%. This is as we expected since transfer learning is effective when the target train set is small.

7.3.1 Mutl-source cross-lingual transfer learning

In Section 6.3.5, we showed that the target domain performance can be improved by utilizing multiple source domains with multi-source domain-adversarial training. Similarly, we can improve the target language performance by knowledge transfer from multiple source languages with multi-source language-adversarial training.

We evaluated the utilization of multiple languages as the source languages for the cross-lingual transfer learning. Since we have four languages for each of Germanic, Slavic, and Romance language families, we evaluated the performance of those languages using the other languages in the same language families as the source languages since we expect that languages in the same language family are more likely to be compatible and helpful each other.

For the efficiency of the evaluations, we performed multi-task learning for multiple languages rather than differentiating the targets from sources. For the adversarial training with K sources and a single target, we utilized $K+1$ binary language discriminators, which is a variation of using K binary discriminators in Section 6.2.2. For the input from the $i-th$ language, we train the language discriminators by setting the target label of the $i-th$ discriminator to be 1 and those of all the other discriminators to be 0. Since there is no differentiation between the sources and the targets in the $K+1$ discriminators, utilizing the $K+1$ binary discriminators is more proper than using the K binary discriminators for multi-task

		Target only		Source (Languages in the same family) $\rightarrow$ Target						
Language Family	Language	p	p,l	p,l	c,l	c,l+a	c,l+aw	p,c,l	p,c,l+a	p,c,l+aw
Germanic	Swedish	93.26	94.31	94.57	94.96	94.89	94.62	94.72	**94.99**	94.73
	Danish	92.13	93.41	93.95	93.98	94.00	93.73	**94.19**	94.00	93.97
	Dutch	83.24	84.73	**86.22**	85.76	85.28	84.97	85.35	85.33	86.14
	German	89.27	**90.69**	90.48	89.94	89.77	90.23	90.48	90.58	90.45
	Avg	89.47	90.78	91.30	91.16	90.98	90.89	91.18	91.23	**91.32**
Slavic	Slovenian	93.06	93.79	**94.65**	94.28	94.38	94.41	94.44	94.52	94.57
	Polish	91.30	91.30	92.73	92.96	92.80	92.86	**93.21**	92.93	92.89
	Slovak	86.53	89.57	91.11	91.17	91.42	91.50	91.59	91.72	**91.94**
	Bulgarian	93.45	95.27	**95.72**	95.02	95.20	95.37	95.70	95.58	95.72
	Avg	91.09	92.48	93.56	93.36	93.45	93.53	93.74	93.69	**93.78**
Romance	Romanian	93.20	94.09	94.72	94.51	94.37	94.44	94.65	94.81	**94.82**
	Portuguese	94.23	95.18	95.79	95.61	95.86	95.89	**95.91**	95.86	95.79
	Italian	93.80	95.95	96.31	96.13	95.90	96.05	**96.41**	96.25	96.30
	Spanish	91.94	93.34	93.28	93.13	92.88	93.37	93.06	**93.50**	93.22
	Avg	93.29	94.64	95.02	94.84	94.75	94.94	95.01	**95.10**	95.03
	Total Avg	91.29	92.64	93.29	93.12	93.06	93.12	93.31	93.34	**93.38**

Table 7.5: POS tagging accuracies (%) with 1,280 tag-labeled training examples for all the languages. All the training examples are still used for the other objectives. Languages in the same family are trained together with multi-task learning. *aw* denotes utilizing adversarial training with Wasserstein distance.

learning cases that we evaluated. In this multi-source language-adversarial training, we also evaluated adversarial training with Wasserstein distance, which was shown effective for cross-lingual sentiment classification tasks (Chen et al., 2017).

Tables 7.5, 7.6, and 7.7 show POS tagging accuracies with 1,280, 320, and 32 tag-labeled training examples utilizing multiple languages as the sources, respectively.

Comparing Tables 7.2 and 7.5, Tables 7.3 and 7.6, and Tables 7.4 and 7.7, utilizing three other languages belonging to the same language family as the source languages shows noticeably better performance than using English as a single source language. Considering that utilizing 1,280*3=3,840, 320*3=960, or 32*3=96 tag labels from three other languages showed better results than using 12,543 English tag labels as the source, we can see that

		Target only		Source (Languages in the same family) → Target						
Language Family	Language	p	p,l	p,l	c,l	c,l+a	c,l+aw	p,c,l	p,c,l+a	p,c,l+aw
Germanic	Swedish	87.43	90.49	91.67	91.51	91.02	91.70	**91.71**	91.66	91.69
	Danish	86.42	90.01	91.20	91.20	90.75	90.86	91.43	**91.52**	91.30
	Dutch	76.76	82.24	**82.68**	81.47	81.40	81.91	82.37	82.12	81.96
	German	86.25	88.95	89.05	88.50	87.99	87.98	**89.48**	89.22	89.35
	Avg	84.22	87.92	88.65	88.17	87.79	88.13	**88.75**	88.63	88.57
Slavic	Slovenian	87.02	89.97	90.77	90.81	90.88	91.13	**91.24**	90.84	91.06
	Polish	82.10	84.13	**87.82**	87.27	87.32	86.71	87.14	87.56	87.56
	Slovak	76.22	81.03	84.31	85.05	85.16	85.27	85.29	**85.49**	85.22
	Bulgarian	87.32	92.81	92.91	92.16	92.31	92.28	92.88	**92.95**	92.85
	Avg	83.16	86.98	88.95	88.82	88.92	88.85	89.14	**89.21**	89.17
Romance	Romanian	88.67	91.44	**92.56**	91.95	91.83	91.91	92.21	92.11	92.11
	Portuguese	90.66	93.73	94.32	94.33	94.07	94.08	94.31	**94.71**	94.50
	Italian	89.78	93.99	94.08	93.61	93.33	93.40	94.29	**94.50**	94.32
	Spanish	85.91	91.07	90.97	90.60	90.01	90.60	90.92	90.44	**91.09**
	Avg	88.76	92.56	92.98	92.62	92.31	92.50	92.93	92.94	**93.00**
	Total Avg	85.38	89.15	90.20	89.87	89.67	89.82	**90.27**	90.26	90.25

Table 7.6: POS tagging accuracies (%) with 320 tag-labeled training examples for all the languages. All the training examples are still used for the other objectives. Languages in the same family are trained together with multi-task learning.

		Target only		Source (Languages in the same family) → Target						
Language Family	Language	p	p,l	p,l	c,l	c,l+a	c,l+aw	p,c,l	p,c,l+a	p,c,l+aw
Germanic	Swedish	60.85	81.08	84.30	85.07	84.88	84.82	85.06	85.12	**85.35**
	Danish	56.62	83.67	84.84	84.86	85.18	84.43	**86.22**	86.03	85.93
	Dutch	59.59	75.85	77.10	76.30	74.43	75.71	**78.16**	77.94	77.92
	German	61.58	81.88	81.67	**82.11**	82.03	81.62	81.93	81.97	81.47
	Avg	59.61	80.62	81.98	82.08	81.63	81.65	**82.84**	82.76	82.67
Slavic	Slovenian	58.15	80.90	83.09	82.05	82.00	81.85	**83.43**	82.88	82.61
	Polish	55.95	72.69	76.02	76.59	77.34	77.30	**78.36**	78.11	77.86
	Slovak	51.69	68.80	73.20	74.59	75.21	74.36	74.42	75.36	**75.76**
	Bulgarian	62.58	84.50	84.24	84.49	84.40	84.28	84.00	84.38	**84.91**
	Avg	57.09	76.72	79.14	79.43	79.74	79.45	80.05	80.18	**80.29**
Romance	Romanian	61.44	83.79	86.38	85.98	85.18	85.61	**86.42**	86.14	86.02
	Portuguese	66.81	88.74	91.06	90.09	90.67	89.80	**91.40**	91.18	91.22
	Italian	69.79	87.60	88.79	88.56	88.60	89.06	**89.79**	89.68	89.34
	Spanish	57.74	82.50	81.25	82.62	82.28	82.04	**82.65**	81.78	82.21
	Avg	63.94	85.66	86.87	86.81	86.68	86.63	**87.57**	87.19	87.20
	Total Avg	60.22	81.00	82.66	82.77	82.68	82.57	**83.49**	83.38	83.38

Table 7.7: POS tagging accuracies (%) with 32 tag-labeled training examples for all the languages. All the training examples are still used for the other objectives. Languages in the same family are trained together with multi-task learning.

knowledge transfer from multiple languages can be more helpful than that from single resource-rich source language.

One issue is that utilizing adversarial training with $K+1$ binary discriminators was not noticeably effective on average in the multi-task cases. Also, utilizing Wasserstein distance did not show significant improvements compared to using the original adversarial training. It would be necessary to find more effective adversarial training methods for the multi-task cases.

7.3.2 Implementation details

All the models were optimized using ADAM (Kingma and Ba, 2015)[29] with minibatch size 32 for total 100 epochs and we picked the parameters showing the best accuracy on the development set to report the score on the test set. The dimensionalites of all the BLSTM related layers follow Plank et al. (2016)'s model. Each word vector is 128 dimensional and each character vector is 100 dimensional. They are randomly initialized with Xavier initialization (Glorot and Bengio, 2010).[30] For stable training, we use gradient clipping, where the threshold is set to 5. The dimensionality of each hidden output of LSTMs is 100, and the hidden outputs of both forward LSTM and backward LSTM are concatenated, thereby the output of each BLSTM for each time step is 200. Therefore, the input to the common BLSTM and the private BLSTM is 128+200=328 dimensional. The inputs and the outputs of the BLSTMs are regularized with dropout rate 0.5 (Pham et al., 2014). For the consistent dropout usages, we let the dropout masks to be identical for all the time steps of each sentence (Gal and Ghahramani, 2016). For all the BLSTMs, forget biases are

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

[30]Sampled from $N\left(0, 2/(n_{in} + n_{out})\right)$, where n_{in} and n_{out} are the dimensionalities of the input and the output.

initialized with 1 (Jozefowicz et al., 2015) and the other biases are initialized with 0. Each convolution filter output for the sentence encoding is 64 dimensional, and the three filter outputs are concatenated to represent each sentence with a 192 dimensional vector.

7.4 Conclusion

In this chapter, we introduced a cross-lingual transfer learning model for POS tagging which uses separate BLSTMs for language-general and language-specific representations. Evaluating on 14 different languages, including the source language improved tagging accuracies in almost all the cases. Specifically, our model showed noticeably better performance when the source language and the target languages belong to the same language family, and competitively performed with the highest average accuracies for target languages in different families.

Acknowledgements

All the experiments in this chapter were conducted with machines at Ohio Supercomputer Center (1987).

CHAPTER 8: CONCLUSION

8.1 Summary of contributions

In this dissertation, we first addressed the representations of semantic intensity scales with word vectors obtained from a RNNLM. Then, we utilized external linguistic resources including synonyms, antonyms, and semantic intensity orders to enrich word embeddings so that the semantic intensities can be represented better in vector spaces. We also showed that word embedding enrichment can be helpful for not only word-level tasks but also sentence-level downstream tasks such as intent detection.

In addition to word-level knowledge transfer methods, we introduced sentence-level transfer learning models. We showed that cross-domain transfer learning for slot-filling, which is shown effective when multiple source domain datasets are available, and cross-lingual transfer learning for POS tagging, which allows knowledge transfer across languages.

8.2 Future directions

We briefly discuss two of potential future directions for improving the sentence-level transfer learning models.[31]

[31]Future directions for word-level transfer learning were discussed in Section 4.5 and 5.5.

8.2.1 Improving adversarial training methods

There are approaches trying to improve Wasserstein adversarial training. Gulrajani et al. (2017) utilized gradient norm penalization for enforcing a Lipschitz constraint instead of the weight clipping of the discriminator in (Arjovsky et al., 2017). Since the weight clipping can cause capacity underuse and potential exploding/vanishing gradient problems, removing that constraint can improve the GAN performance.

Another work by Bellemare et al. (2017) showed that minimizing Wasserstein distance may fail to converge to minimize the true distance because of biased sample gradients when using stochastic gradient descent for the optimization. They proposed utilize the energy distance, which has unbiased sample gradients.

We can adopt those new adversarial training techniques to further improve the performance of cross-domain or cross-lingual transfer learning tasks.

8.2.2 Multi-source cross-lingual transfer learning

In Section 7.3.1, we showed that utilizing multiple languages as the sources can perform significantly better than using single language as the source. As future work, we can try even more languages belonging to other language families as the source languages. Other than language families, there are also other factors such as typology and alphabet systems, which categorize languages. We can try to find out which factors should be used for the source language selection to further improve the target language performance.

8.3 Final remarks

As recent deep learning models are becoming more complex and new types of NLP tasks are emerging without sufficient amount of training instances, transfer learning methods are becoming more important since they enable improving the target task performance with complex models without enough target training examples. We introduced different types of transfer learning models that can be used for improving the performance of various word-level and sentence-level NLP downstream tasks. Although we showed the effectiveness of our knowledge transfer models on specific tasks, as ours are general without utilizing too much task specific heuristics, we expect that our approaches can easily help other types of tasks as well.

APPENDIX A: WORD EMBEDDINGS WITH MULTIPLICATIVE FEATURE INTERACTIONS FOR TENSOR-BASED COMPOSITIONS

In this appendix, we discuss an extension of the CBOW model, which utilizes different types of phrase composition methods during the CBOW training. We show that making phrase composition methods in the CBOW training to be similar to the phrase composition in the models for downstream tasks can be helpful to improve the performance of the downstream tasks.

A portion of this chapter has been published as Kim et al. (2015a).

A.1 Introduction

Mikolov et al. (2013a,b) introduced `word2vec` that includes the continuous bag-of-words (CBOW) model and the skip-gram model.[32] These models have been widely used for generating word vectors to be used for word related tasks because of their efficient but still effective architectures. The CBOW model takes the mean vector of projections of the context words and use it to predict the target word as the following objective function:[33]

$$\frac{1}{T}\sum_{t=1}^{T} \ln p\left(w_t \middle| \frac{1}{2c}\sum_{-c\leq j\leq c, j\neq 0} p_{t+j}\right), \tag{A.1}$$

where T is the total number of words in a corpus, w_t is the tth word, p_t is the tth word vector, and c is the half window size.

[32] `https://code.google.com/p/word2vec`

[33] Although sum is used in Mikolov et al. (2013a), the current version of `word2vec` implementation uses mean.

Milajevs et al. (2014) showed that the word vectors generated from the CBOW model are competitive with those from co-occurrence based models for both simple arithmetic compositions and tensor-based compositions for categorical compositional distributional models (Coecke et al., 2010).[34]

Categorical compositional distributional models represent compositional semantics with algebra of Pregroup by representing each grammatical reduction as a linear map in vector spaces (Coecke et al., 2010; Kartsaklis et al., 2012). For example, *cats like milk* consists of a subject noun, a transitive verb requiring a subject and an object, and an object noun, respectively. In Pregroup grammar, the types of the three words in this example are n, $\left(n^r s n^l\right)$, and n, respectively, where n is a noun, n^r can be combined with a n in the left, n^l can be combined with a n in the right, and s is a declarative statement. Then, they can be reduced to represent the entire phrase with single entity as follows:

$$n\left(n^r s n^l\right) n \rightarrow 1 s n^l n \rightarrow 1 s 1 \rightarrow s \tag{A.2}$$

In the reduction, n^r is composed with the left n resulting in an identity element, 1. Then, n^l is composed with the right n resulting in another 1. Because 1 is an identity element, $1s1$ is reduced to s.

Since there is no specification of actual implementation of the composition in categorical compositional distributional models, different composition methods have been introduced; they are reviewed in Section A.2. However, there are few studies about the vector representation of single words regarding those compositions.

One issue of using the word vectors from the CBOW model as the constituent vectors for tensor-based composition is that their assumptions of the composition are different.

[34]Although Milajevs et al. (2014) described that the skip-gram model was used to generate the word vectors, the CBOW model was actually used in their work.

Method	Phrase	Composition formula	Reference
Addition	$w_1 w_2 ... w_n$	$\overrightarrow{w_1} + \overrightarrow{w_2} + ... + \overrightarrow{w_n}$	Mitchell and Lapata (2008)
Multiplication		$\overrightarrow{w_1} \odot \overrightarrow{w_2} \odot ... \odot \overrightarrow{w_n}$	
Relational	Sbj Verb Obj	$\overline{verb} \odot (\overrightarrow{Sbj} \otimes \overrightarrow{Obj})$	Grefenstette and Sadrzadeh (2011a)
Kronecker		$\widetilde{verb} \odot (\overrightarrow{Sbj} \otimes \overrightarrow{Obj})$	Grefenstette and Sadrzadeh (2011b)
Copy sbj.	Sbj Verb Obj	$\overrightarrow{Sbj} \odot (\overline{Verb} \times \overrightarrow{Obj})$	Kartsaklis et al. (2012)
Copy obj.		$\overrightarrow{Obj} \odot (\overline{Verb}^{\top} \times \overrightarrow{Sbj})$	
Frob. add.	Sbj Verb Obj	$(\overrightarrow{Sbj} \odot (\overline{Verb} \times \overrightarrow{Obj})) + (\overrightarrow{Obj} \odot (\overline{Verb}^{\top} \times \overrightarrow{Sbj}))$	Kartsaklis and Sadrzadeh (2014)
Frob. mult.		$(\overrightarrow{Sbj} \odot (\overline{Verb} \times \overrightarrow{Obj})) \odot (\overrightarrow{Obj} \odot (\overline{Verb}^{\top} \times \overrightarrow{Sbj}))$	
Frob. outer		$(\overrightarrow{Sbj} \odot (\overline{Verb} \times \overrightarrow{Obj})) \otimes (\overrightarrow{Obj} \odot (\overline{Verb}^{\top} \times \overrightarrow{Sbj}))$	

Table A.1: Tensor-based composition methods (Milajevs et al., 2014).

Word embeddings of the CBOW model are trained with an additive context composition, which is the mean of the context projection. However, most tensor-based compositions use point-wise multiplication or tensor product as composition operators. This means that there is a mismatch between the composition method used for the training of the underlying word vectors and the actual composition methods we evaluate.

To alleviate the mismatch, we introduce extensions of the CBOW model with multiplicative interactions between word projections to obtain word embeddings more suitable for the tensor-based compositions. For four datasets, evaluating different types of compositions, we show that those extensions of the CBOW model improve the performance of the actual composition tasks with multiplication or tensor product operations.

A.2 Tensor-based compositions

Prior to discussing the modification to the CBOW algorithm, we review different composition methods used in the literature (Table A.1).

Addition and *Multiplication* are compositions by point-wise addition and multiplication, respectively (Mitchell and Lapata, 2008). They can be done simply without any other information, but they cannot reflect word orders and grammatical structures.

Mitchell and Lapata (2008, 2009) showed that composition by multiplication can be more effective than composition by addition because additive models compose by considering the content altogether whereas multiplicative models focus on the content relevant to the composition by scaling each element of one with the strength of the corresponding element of the other. Using multiplication as the composition method could be unstable in the previous work because multiplication with zero or negative values changes the value abruptly (Mitchell and Lapata, 2009). In our models, however, these instability issues could be alleviated since the training model adapt the constituent word vectors to be proper for the composition by multiplication. Mitchell and Lapata (2010) also showed that the tensor product is effective to represent composition because it allows the interactions between different features in different vectors whereas point-wise multiplication can interact with only the same feature in different vectors. Therefore, we also examine an extension of the CBOW model using tensor product for modeling local context.

There are neural network models using multiplicative interactions in the architectures. Sum-Product Networks use layer-wise multiplicative interactions (Poon and Domingos, 2011; Cheng et al., 2014) and multiplicative recurrent neural networks use multiplication of hidden state outputs from previous time step with the current word projections (Sutskever et al., 2011; Irsoy and Cardie, 2015). These approaches capture multiplicative interactions with hidden layer outputs. Our approach instead utilizes multiplicative interactions in the training of the CBOW model, making the embedded vector spaces more in tune with the compositions of end tasks.

The third to the last composition methods of Table A.1 shows tensor-based composition methods for representing phrases consist of subjects, transitive verbs, and objects in categorical compositional distributional models. $\overline{verb} = \sum_i \overrightarrow{Sbj_i} \otimes \overrightarrow{Obj_i}$ represents a verb with the subjects and the objects of the verb across the corpus. The subject and the object of each transitive verb required for calculating $\overline{verb}$ are identified from the dependency tree of PukWaC 1.0 dataset, which consists of web documents in .uk domain crawled with the medium-frequency words from the British National Corpus (BNC) (Burnard, 2007) as the seeds (Baroni et al., 2009; Johansson, 2007). [35] $\widetilde{verb} = \overrightarrow{verb} \otimes \overrightarrow{verb}$ represents a verb as the tensor product of the corresponding verb vector. Those methods consider the relations between transitive verbs and their subjects and objects. Therefore, we can represent their compositions more effectively. Recursive neural tensor networks also use tensor product information in the recursive composition (Socher et al., 2013b), but they require training labels and only support binary compositions.

Relational and *Kronecker* represent each phrase by the multiplication of the verb matrix to the tensor product of the subject and the object (Grefenstette and Sadrzadeh, 2011a,b). Although they can represent interactions between subjects and objects as well as the verbs, it is difficult to compose them with other phrases in a uniform way since the result dimensionality is the square of the original vectors. In addition, dealing with large dimensional tensors is not very scalable.

The fifth to the last composition methods use Frobenius operators for the compositions (Kartsaklis et al., 2012), which can resolve the dimensionality issues by maintaining the original dimensionality through matrix-vector multiplication. In *Copy subject*, the verb

[35] Available from http://wacky.sslmit.unibo.it/doku.php?id=corpora

matrix $\overline{verb}$ is multiplied with the object vector and then composed with the subject vector by point-wise multiplication. *Copy object* is opposite in terms of the positions of the subject and the object. These two methods are different ways of diagonal placement of a plane into a cube (Kartsaklis et al., 2012). The last three methods, *Frobenius addition*, *multiplication*, and *outer product*, represent different combinations of *Copy subject* and *Copy object* (Kartsaklis and Sadrzadeh, 2014).

A.3 Extending the CBOW model with multiplicative interactions between word projections

As briefly discussed in the introduction, the CBOW model is an additive model in terms of the composition since the mean of the context word projections is used to predict the target word. As many composition methods in Table A.1 use multiplication or tensor product as the composition operators, if these operators are used to compose the contexts in the CBOW model, then the training process can optimize the model to consider their word embeddings to be composed with those multiplicative operations. Therefore, we can train word embeddings that are more suitable for the composition methods that we are evaluating.

In the CBOW model, the point-wise mean of the word projections is used to predict the target word as shown in Equation A.1. In addition to the baseline, we experimented with adding different multiplicative terms as shown in Table A.2. The added terms are selected to reflect the operations of composition methods in Table A.1 and their combinations. In the expressions, p_i is the projection of the ith input context word and c is the size of the context window, which is the number of neighboring words used as the input for each direction.

	Type	Expression
1	mean (baseline, Milajevs et al. (2014))	$\sum_{-c\leq i\leq c, i\neq 0} p_i/2c$
2	pointwise multiplication	$\prod_{-c\leq i\leq c, c\neq 0} p_i$
3	mean + pointwise multiplication	mean $+ \prod_{-c\leq i\leq c, c\neq 0} p_i$
4	concat{mean, pointwise multiplication}	concat{mean, $\prod_{-c\leq i\leq c, c\neq 0} p_i$}
5	mean + projection of p_{i-1} and p_{i+1}	mean $+W_p$concat$\{p_{i-1}, p_{i+1}\}$
6	projection of tensor product of p_{i-1} and p_{i+1}	$W_{tp}(p_{i-1} \otimes p_{i+1})$
7	mean + projection of tensor product of p_{i-1} and p_{i+1}	mean $+W_{tp}(p_{i-1} \otimes p_{i+1})$

Table A.2: Different outputs of the projection layer. p_i is the projection of the ith input context word, c is the size of the context window, and W_p and W_tp are projection matrices.

The second model, *mult*, uses only the multiplication of projections, which best fits to the composition by point-wise multiplication. The third and the fourth models evaluate the performance when both the additive and multiplicative interactions are used together since their combination has been shown to be effective (Mitchell and Lapata, 2008). The third model adds the additive terms and multiplicative terms whereas the fourth model concatenates these terms so that they influence the output separately.

In the fifth to the last models, we try to further use the information from p_{i-1} and p_{i+1}, which are the projections of the nearest neighbor words of the ith target word in the training corpus. The fifth model concatenates p_{i-1} and p_{i+1} and project to the original dimension with a projection matrix W_p. This result is added to the baseline model so that information from the nearest words considering the order can be used to estimate the target. W_p is also updated during the training.

In the sixth model, since the tensor-based compositions are used as Table A.1 and they can represent multiplicative interactions between different features, we use the tensor product of the projections of (p_{i-1} and p_{i+1}). The tensor product output is also projected to the original dimensionality by multiplying a projection matrix W_{tp}, which is also updated during the training. Although this model can use more powerful interactions of neighbor words, it can only use the information from the nearest neighbor words and it cannot use two word sentences in the training corpus for the training. To deal with these issues, in the last model, we combine the mean with the projection of the tensor product.

A.4 Experiments

To evaluate the five different CBOW-based models proposed in Section A.3, we use the following datasets: similarity of transitive verbs with multiple senses from Grefenstette and Sadrzadeh (2011a), three-word sentence similarity from Kartsaklis and Sadrzadeh (2014), paraphrase detection from Dolan et al. (2005), and dialog act tagging for the Switchboard corpus (Godfrey et al., 1992) from Stolcke et al. (2000). These are all the datasets evaluated in Milajevs et al. (2014)'s work as well. Each phrase in the first two datasets is fixed as a subject, a transitive verb, and an object whereas the length of each phrase in the last two datasets is arbitrary.

There are several differences between our word vectors and the ones used in Milajevs et al. (2014). First, we use BNC as the training set while Milajevs et al. (2014) use pretrained word vectors from `word2vec` that are trained using GoogleNews dataset. To reduce the size of projection matrices, all the words are lower-cased and words occurring 20 times or less are converted to the words' POS tags. Second, instead of negative sampling, our models use hierarchical softmax as the objective function, where each word is represented as a leaf node of Huffman tree since hierarchical softmax is better for training with infrequent words (Mikolov et al., 2013b). Third, we use gradient clipping for more stable training since gradient can be fluctuating when the projections are multiplied. All the other parameters for the training are the same as those used for Milajevs et al. (2014)'s experiments.

Using the mean as the network combination function can be considered a reimplementation of Milajevs et al. (2014)'s system subject to the changes mentioned above. We trained the CBOW-based models and obtained 300 dimensional word vectors, which are with the same dimensionality used in Mikolov et al. (2013a,b); Milajevs et al. (2014).

A.4.1 Fixed phrases (three-word)

Table A.3 shows the experiment results for the three-word phrases. The first column represents the two evaluation tasks, the second column is the composition methods described in Table A.1, and the third column shows the results of neural word embeddings (NWE) from previous work (Milajevs et al., 2014).[36] Bold entries in the table indicate the highest scores among our models.

In the datasets, human annotators rated each phrase pair for semantic similarity (from 1 “no similarity" to 7 “high similarity"). As each unique phrase pair is judged by multiple people, following Milajevs et al. (2014), we took the mean of the ratings to set the rating of each unique pair. Scores in the table entries are Spearman's ρs. A high value of Spearman's ρ in the table means that the similarity of the composed phrases in the vector space is highly correlated with the semantic similarity of the phrases judged by humans. Therefore, if a model shows high scores, it reflects that the model is good at representing the semantics for those short phrases.

[36]The word vectors are available from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

Task	Method	(Milajevs et al., 2014)	mean	mult	mean + mult	concat {mean,mult}	mean + nbr_prj	nbr_outer_prj	mean + nbr_outer_prj
Similarity of transitive verbs	Verb only	0.107	0.130	0.014	0.136	0.204	0.187	0.072	**0.250**
	Addition	0.149	0.066	0.012	0.046	-0.030	0.100	0.111	**0.145**
	Multiplication	0.095	0.160	**0.249**	0.058	0.219	0.113	0.050	0.204
	Kronecker	0.117	0.160	0.160	0.121	0.229	0.168	0.047	**0.245**
	Relational	0.362	0.330	0.276	0.319	0.280	0.344	0.316	**0.365**
	Copy sbj.	0.131	0.249	0.064	0.262	0.209	0.262	0.168	**0.290**
	Copy obj.	0.456	0.302	0.361	0.329	**0.382**	0.300	0.371	0.322
	Frob. add.	0.359	0.337	0.293	0.345	0.288	0.349	0.250	**0.355**
	Frob. mult.	0.239	0.270	0.252	0.255	0.189	0.293	0.196	**0.309**
	Frob. outer.	0.375	0.330	0.275	0.339	0.351	0.329	0.293	**0.387**
Sentence similarity	Verb only	0.561	0.528	0.360	0.520	0.531	0.527	0.260	**0.536**
	Addition	0.689	0.728	0.572	0.738	**0.770**	0.722	0.401	0.706
	Multiplication	0.341	0.062	**0.625**	0.178	0.440	0.110	0.269	0.220
	Kronecker	0.561	0.206	**0.623**	0.277	0.501	0.203	0.003	0.457
	Relational	0.618	0.505	**0.665**	0.540	0.527	0.525	0.157	0.574
	Copy sbj.	0.405	0.390	0.453	0.353	0.436	0.396	0.139	**0.454**
	Copy obj.	0.655	0.481	**0.607**	0.487	0.500	0.488	0.190	0.510
	Frob. add.	0.585	0.489	**0.610**	0.407	0.528	0.439	0.210	0.501
	Frob. mult.	0.387	0.211	**0.608**	0.323	0.419	0.335	0.065	0.349
	Frob. outer.	0.622	0.504	**0.664**	0.510	0.544	0.524	0.165	0.569

Table A.3: Spearman's ρ on the similarity of transitive verbs with multiple senses (top) and three-word sentence similarity (bottom). The mean column can be considered an implementation of the Milajevs et al. (2014)'s model on the BNC corpus.

Similarity of transitive verbs

The top of Table A.3 shows the results for the similarity of 199 three-word phrase (subject, transitive verb, and object) pairs introduced in Grefenstette and Sadrzadeh (2011a).[37] In each phrase pair, the transitive verbs are the same but the subjects and the objects are different for each other. We try to identify the senses of a transitive verb with the different contexts. For example, "meet" is a verb with multiple senses. If the given subject is "system" and the object is "specification", "meet" would be semantically closer to "satisfy" than "visit". Then, given "system meets specification" and "system satisfies specification" as a pair, the judge would give a high rating for the similarity of the verbs.

[37] Available from http://www.cs.ox.ac.uk/activities/compdistmeaning/GS2011data.txt

Our results were not consistently better than Milajevs et al. (2014)'s results. However, considering that the model used for the previous work and our baseline (*mean*) are similar CBOW models, the performance difference would mainly due to the different training sets (GoogleNews and BNC). Among our models, adding tensor product result to the mean (*mean+nb_outer_prj*) showed the best performance in most types of compositions. Interestingly, the power seems to come from the combination of mean and the tensor product, as both individually perform worse than the combination. For this dataset, as the verbs are the same for both phrases in each pair, the subjects and the objects play important roles for the verb disambiguation. When a transitive verb is an ith word in a sentence denoted as w_i, in many cases, the subject and the object are w_{i-1} and w_{i+1}, respectively. Since tensor product of the $i-1$th word projection and the $i+1$th word projection can represent multiplicative interactions between different features of the two words, considering the tensor product of those projections in the model could be helpful to identify the transitive verbs.

Similarity of three-word phrases

The bottom of Table A.3 shows the evaluation results on the similarity of 109 three-word phrase pairs with human judged ratings from (Kartsaklis and Sadrzadeh, 2014).[38] For example, the similarities of two sentences like “programme offer support" and “service provide help" are evaluated.

[38]Available from http://www.cs.ox.ac.uk/activities/compdistmeaning/emnlp2013_turk.txt

Task	Method	(Milajevs et al., 2014)	mean	mult	mean + mult	concat {mean,mult}	mean + nbr_prj	nbr_outer_prj	mean + nbr_outer_prj
Paraphrase detection	Addition	0.73	0.686	0.665	**0.690**	0.688	0.689	0.684	0.688
	Multiplication	0.42	0.393	**0.652**	0.388	0.587	0.387	0.412	0.371
Dialog act tagging	Addition	0.63	**0.638**	0.636	0.633	0.636	0.636	0.565	0.626
	Multiplication	0.58	0.522	**0.606**	0.593	0.515	0.581	0.573	0.598

Table A.4: Accuracies on the paraphrase detection (top) and the dialog act tagging (bottom). The mean column can be considered an implementation of the Milajevs et al. (2014)'s model on our training set.

In this evaluation, considering the interleaved words with tensor product (*mean+nb_outer_prj*) still showed better performance than the baseline (*mean*) for the most composition methods except addition. However, the multiplication only model, *mult*, showed the best performance in most cases except when verb only or addition were used as the composition methods.

A.4.2 Arbitrary length phrases

The three-word phrases in the previous section are useful for the evaluation of the tensor-based compositions since we do not need to care about the structural variations of the phrases. However, we would be more interested in phrases where the lengths are not fixed. As each phrase can have different length, we cannot use the tensor-based compositions used for the fixed-length phrases. Therefore, we evaluated the composition of each phrase by only using point-wise addition and multiplication.

Table A.4 shows the accuracies of classification tasks given arbitrary length phrases as the inputs. The results evaluate whether composition of arbitrary length phrases can be well represented with the word vectors from the proposed models.

Paraphrase detection

The top of Table A.4 shows the binary classification accuracies on the Microsoft Research Paraphrase Corpus (Dolan et al., 2005), which consists of arbitrary length phrase pairs. In this dataset, each phrase pair comes with a binary label: 1 if the phrases were judged to be paraphrases, 0 otherwise. The minimum, mean, and maximum lengths of the phrases in the training set are 6, 19.8, and 35, respectively.

With this dataset, we can evaluate if our models work well for representing general phrases. Following the setting of Milajevs et al. (2014)'s work, we trained a linear binary classifier on 2000 phrase pairs and tested on 1726 phrase pairs. The classifier is trained to find the threshold of cosine similarity deciding if two phrases are paraphrases or not.

Comparing to the baseline CBOW model, there were no significant gain in the proposed models for the composition by addition. However, using multiplication of the projections (*mult*) showed significantly better performance when composed by multiplication, and started to show statistical insignificance to additive composition methods when tested by McNemar's test with p-value 0.05.

Dialog act tagging

The bottom of Table A.4 shows the classification accuracies of dialog act tagging (Stolcke et al., 2000) on the Switchboard corpus (Godfrey et al., 1992). Switchboard is a collection of about 2400 telephone dialogs among 543 speakers in the United States. Each utterance is assigned one of 42 dialog-act tags, which summarize syntactic, semantic and pragmatic information about the turns (e.g., yes/no question, yes answer, agree).[39] The minimum, mean, and maximum lengths of the phrases in the training set are 0, 34.1, and

[39]The tags are described in `http://web.stanford.edu/~jurafsky/ws97/manual.august1.html`.

549, respectively. Zero length phrases exist because of the preprocessing, and they are ignored.

The task in this section is identifying the dialog act tags from given utterances. Following Milajevs and Purver (2014); Milajevs et al. (2014), we used the first 1115 utterances as the training set and the following 19 utterances as the test set. We also concatenated utterances separated by an interruption by the other person (Webb et al., 2005), and we removed disfluency markers and punctuation signs. Once we have the vectors composed by either addition or multiplication for all of the utterances in the training set, the vector dimensionality is reduced to 50 by Singular Value Decomposition (SVD) and a k-nearest-neighbor classifier (k=5) is used to identify the dialog act tags.[40] The baseline (*mean*) model showed the best performance for the composition by addition and the *mult* model was the best for the composition by multiplication, but the differences were insignificant in this case.

The results on both evaluation for arbitrary length phrases support that matching the composition of contexts for the training of constituent word vectors with the actual composition methods shows better or competitive performance.

A.5 Conclusion

We showed the experiment results on seven types of word vectors trained using different composition methods. Overall, we can see that multiplicative interactions in the CBOW models can help representing compositions that are multiplicative in nature.

Using only the multiplication of projections showed significant improvement for all the evaluated datasets when the phrases are composed with multiplications. Because the

[40]We used scikit-learn (Pedregosa et al., 2011) to run SVD and k-NN classifiers.

composition used for the training of word vectors is matching to the actual evaluated compositions, we can think that the word vectors are trained to represent their multiplications properly. One evidence is that the mean of word vectors of the $mult$ model is around 0.12 while the means of the other models are around 0. Since there are fewer negative elements in the word vectors of the $mult$ model, the composition by multiplication produces relatively more positive values. This possibly gives more stable results when used in multiplication-based compositions since fluctuations of the composition by multiplication with negative values is reduced. In the task of transitive verb disambiguation, since the interactions between non-adjacent subjects and objects are important, having their tensor product as a term in the model (*mean+nbr_outer_prj*) was noticeably helpful. In the task of three-word phrase similarity, using the tensor product as a term still showed better performance than using the models of *mean* and *mean+mult* in most cases except when the phrases are composed with addition. Interestingly, however, the model with only multiplication showed the best performance for most of the compositions by multiplication and tensor product.

In summary, for better representation of phrase compositions, we showed that it can be helpful to train the word embedding models by composing the input contexts of the model to be similar to the actual composition methods to be used because the word vectors are adjusted to more properly represent the composition by the composition method used. Specifically, using point-wise multiplication in the training model consistently showed better performance when the actual composition is also multiplication. The *mean+nbr_outer_prj* model, which is with the combination of mean and tensor product also showed better or similar performance for tensor-based composed phrases compared to the *mean* model and the *mean+mult* model.

BIBLIOGRAPHY

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. In *International Conference on Machine Learning (ICML)*. 22, 23, 101

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035. 42

Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora. *Language Resources & Evaluations*, 43(3):209–226. 107

Barrett, M., Keller, F., and Søgaard, A. (2016). Cross-lingual transfer of correlations between parts of speech and gaze features. In *Proceedings of COLING*, pages 1330–1339. 84

Bellemare, M. G., Danihelka, I., Dabney, W., and Mohamed, S. (2017). The Cramér distance as a solution to biased wasserstein gradients. In *arXiv:1705.10743*. 101

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2006). Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 19 (NIPS)*, pages 137–144. 21

Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., Kulesza, A., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175. 21

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research (JMLR)*, 3:1137–1155. xvi, 11, 13, 24

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 2787–2795. 38

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. (2016). Domain separation networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 343–351. 21, 71, 73, 75, 76, 78, 89

Brants, T. and Franz, A. (2006). The Google Web 1T 5-gram Version 1.1. 43

Burnard, L., editor (2007). *Reference Guide for the British National Corpus*. Research Technologies Service at Oxford University Computing Services. 107

Chen, X., Sun, Y., Athiwaratkun, B., Cardie, C., and Weinberger, K. (2017). Adversarial deep averaging networks for cross-lingual sentiment classification. In *arXiv:1606.01614*. 22, 86, 96

Chen, Z., Lin, W., Chen, Q., Chen, X., Wei, S., Jiang, H., and Zhu, X. (2015). Revisiting word embedding for contrasting meaning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 106–115. 37, 55

Cheng, W.-C., Kok, S., Pham, H. V., Chieu, H. L., and Chai, K. M. A. (2014). Language modeling with sum-product networks. In *Interspeech*. 106

Cho, K., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. 20, 55

Coecke, B., Sadrzadeh, M., and Clark, S. (2010). Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384. 104

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167. 24

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learing Research (JMLR)*, 12:2493–2537. 2, 84

de Marneffe, M.-C., Manning, C. D., and Potts, C. (2010). Was it good? It was provocative. Learning the meaning of scalar adjectives. In *Proceedings of the 48th annual Meeting of the Association for Computational Linguistics (ACL)*, pages 167–176. 4, 25, 30, 31, 32, 39, 46

de Melo, G. and Bansal, M. (2013). Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics (TACL)*, 1:279–290. xii, 40, 42, 43, 47, 49, 50

Dolan, B., Brockett, C., and Quirk, C. (2005). Microsoft research paraphrase corpus. Retrieved May, 29:2013. 110, 115

Duong, L., Cook, P., Bird, S., and Pecina, P. (2013). Simpler unsupervised POS tagging with bilingual projections. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 634–639. 84

Eckart, C. and Young, G. (1936). The approximation of one matrix by another lower rank. *Psychometrika*, 1. 10

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1606–1615. 38, 39, 55

Fellbaum, C. (1998). *WordNet: An electronic lexical database*. MIT Press. 4, 30, 39, 55

Firth, J. R. (1957). A synopsis of linguistic theory. *Studies in linguistic analysis*, pages 1–32. 9

Fu, R., Guo, J., Qin, B., Che, W., Wang, H., and Liu, T. (2014). Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1199–1209. 55

Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 1019–1027. 79, 98

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 17:1–35. 21, 73, 77

Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2013). PPDB: The paraphrase database. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 758–764. 4, 18, 55

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *arXiv:1705.03122*. 1

Ghannay, S., Favre, B., Estève, Y., and Camelin, N. (2016). Word embeddings evaluation and combination. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, pages 300–305. 56

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256. 98

Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992). Switchboard: Telephone speech corpus for research and development. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 517–520. 110, 115

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 2672–2680. 21, 22

Graves, A., Jaitly, N., and rahman Mohamed, A. (2013). Hybrid speech recognition with deep bidirectional lstm. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 273–278. 67

Graves, A. and Schmidhuber, J. (2005). Frame-wise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610. 2, 83

Grefenstette, E. and Sadrzadeh, M. (2011a). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1394–1404. 105, 107, 110, 112

Grefenstette, E. and Sadrzadeh, M. (2011b). Experimenting with transitive verbs in a discocat. In *Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics, EMNLP*, pages 62–66. 105, 107

Gross, D. and Miller, K. J. (1990). Adjectives in WordNet. *International Journal of Lexicography*, 3(4):265–277. 42

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of Wasserstein GANs. In *arXiv:1704.00028*. 101

Hakkani-Tür, D., Tur, G., Celikyilmaz, A., Chen, Y.-N., Gao, J., Deng, L., and Wang, Y.-Y. (2016). Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Interspeech*. 56, 60

Hana, J., Feldman, A., and Brew, C. (2004). A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 222–229. 84

Harris, Z. (1954). Distributional structure. *Word*, 10:146–162. 38, 54

He, Y. and Young, S. (2003). A data-driven spoken language understanding system. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 583–588. 3

Hill, F., Reichart, R., and Korhonen, A. (2015). SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695. 18, 51

Hirschberg, J. B. (1985). *A Theory of Scalar Implicature*. PhD thesis, University of Pennsylvania. 39

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780. 2, 55

Horn, L. (1972). *On the Semantic Properties of Logical Operators in English*. Bloomington, Indiana: Indianan University Linguistics Club. 39

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377. 70

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 873–882. 24

Irsoy, O. and Cardie, C. (2015). Modeling compositionality with multiplicative recurrent neural networks. In *International Conference on Learning Representations (ICLR)*. 106

Jaech, A., Heck, L., and Ostendorf, M. (2016). Domain adaptation of recurrent neural networks for natural language understanding. In *Interspeech*. 70, 79

Johansson, R. (2007). Dependency syntax in the CoNLL shared task 2008. 107

Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2342–2350. 99

Kartsaklis, D. and Sadrzadeh, M. (2014). A study of entanglement in a categorical framework of natural language. In *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*. 105, 108, 110, 113

Kartsaklis, D., Sadrzadeh, M., and Pulman, S. (2012). A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *Proceedings of COLING*, pages 549–558. 104, 105, 107, 108

Kim, J.-K. and de Marneffe, M.-C. (2013). Deriving adjectival scales from continuous space word representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1625–1630. 24, 38, 47

Kim, J.-K., de Marneffe, M.-C., and Fosler-Lussier, E. (2015a). Neural word embeddings with multiplicative feature interactions for tensor-based compositions. In *NAACL 2015 workshop on Vector Space Modeling for NLP (VSM-NLP)*, pages 143–150. 16, 103

Kim, J.-K., de Marneffe, M.-C., and Fosler-Lussier, E. (2016a). Adjusting word embeddings with semantic intensity orders. In *ACL 2016 workshop on Representation Learning for NLP (RepL4NLP)*, pages 62–69. 37, 55

Kim, J.-K., Kim, Y.-B., Sarikaya, R., and Fosler-Lussier, E. (2017a). Cross-lingual transfer learning for POS tagging without cross-lingual resources. In *Empirical Methods in Natural Language Processing (EMNLP)*. 83

Kim, J.-K., Tur, G., Celikyilmaz, A., Cao, B., and Wang, Y.-Y. (2016b). Intent detection using semantically enriched word embeddings. In *IEEE Workshop on Spoken Language Technology (SLT)*, pages 414–419. 54

Kim, T., Cha, M., Kim, H., Lee, J. K., and Kim, J. (2017b). Learning to discover cross-domain relations with generative adversarial networks. In *Internationl Conference on Machine Learning (ICML)*. 89

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. 20, 67, 73

Kim, Y.-B., Snyder, B., and Sarikaya, R. (2015b). Part-of-speech taggers for low-resource languages using CCA features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1292–1302. 84

Kim, Y.-B., Stratos, K., and Sarikaya, R. (2016c). Frustratingly easy neural domain adaptation. In *Proceedings of COLING*, pages 387–396. 70, 71, 79, 80, 81

Kim, Y.-B., Stratos, K., Sarikaya, R., and Jeong, M. (2015c). New transfer learning techniques for disparate label sets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 473–482. 70

Kingma, D. P. and Ba, J. L. (2015). ADAM: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. 63, 78, 98

Kipfer, B. A. (2009). *Rogets 21st Century Thesaurus*. Dell. 47

Kokkinos, F. and Potamianos, A. (2017). Structural attention neural networks for improved sentiment analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 586–591. 1

Kusner, M. J. and Hernández-Lobato, J. M. (2016). GANS for sequences of discrete elements with the gumbel-softmax distribution. In *NIPS Workshop on Adversarial Training*. 21

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 260–270. 83

Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284. 27

Levy, O. and Goldberg, Y. (2014). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 302–308. xvi, 16, 17

Levy, R. and Andrew, G. (2006). Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 2231–2234. 43

Li, J., Monroe, W., Shi, T., Ritter, A., and Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 21

Li, X., Wang, Y.-Y., and Tur, G. (2011). Multitask learning for spoken language understanding with shared slots. In *Interspeech*. 70

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015). Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1530. 83

Liu, B. and Lane, I. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech*. 56

Liu, Q., Jiang, H., Wei, S., Ling, Z.-H., and Hu, Y. (2015). Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1501–1511. 38, 39, 55

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*. 74

Mikolov, T., Corrado, G., Chen, K., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) workshop*. xvi, 14, 15, 38, 54, 103, 111

Mikolov, T., Karafiát, M., Burget, L., Cernocky, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, pages 1045–1048. xvi, 13, 24, 26

Mikolov, T., Povey, D., Burget, L., and Cernocky, J. (2011). Strategies for training large scale neural network language models. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 196–201. 25

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 3111–3119. 14, 54, 103, 110, 111

Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–751. xvi, 4, 13, 14, 24, 25, 27, 37

Milajevs, D., Kartsaklis, D., Sadrzadeh, M., and Purver, M. (2014). Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719. xiv, xv, 103, 104, 105, 109, 110, 111, 112, 113, 114, 115, 116

Milajevs, D. and Purver, M. (2014). Investigating the contribution of distributional semantic information for dialogue act classification. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 40–47. 116

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244. 105, 106, 109

Mitchell, J. and Lapata, M. (2009). Language models based on semantic compositon. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 430–439. 106

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive science*, 34:1388–1429. 106

Miyato, T., Dai, A. M., and Goodfellow, I. (2017). Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations (ICLR)*. 22

Mohtarami, M., Amiri, H., Lan, M., and Tan, C. L. (2011). Predicting the uncertainty of sentiment adjectives in indirect answers. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2485–2488. 32

Mohtarami, M., Amiri, H., Lan, M., Tran, T. P., and Tan, C. L. (2012). Sense sentiment similarity: an analysis. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI)*, pages 1706–1712. 33

Mrkšić, N., Ó Séaghdha, D., Thomson, B., Gašić, M., Rojas-Barahona, L., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. (2016). Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 142–148. 38, 39, 40, 48, 55, 56, 58, 65, 68

Mrkšić, N., Ó Séaghdha, D., Wen, T.-H., Thomson, B., and Young, S. (2017). The neural belief tracker: Data-driven dialogue state tracking. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. 18, 67

Nguyen, K. A., im Walde, S. S., and Vu, N. T. (2016). Integrating distributional lexical contrast into word embeddings for antonym–synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459. 55

Nivre, J., Agić, Ž., Ahrenberg, L., Aranzabe, M. J., Asahara, M., Atutxa, A., Ballesteros, M., Bauer, J., Bengoetxea, K., Berzak, Y., Bhat, R. A., Bick, E., Börstell, C., Bosco, C., Bouma, G., Bowman, S., CebiroÄ§lu EryiÄ§it, G., Celano, G. G. A., Chalub, F., Çöltekin, Ç., Connor, M., Davidson, E., de Marneffe, M.-C., Diaz de Ilarraza, A., Dobrovoljc, K., Dozat, T., Droganova, K., Dwivedi, P., Eli, M., Erjavec, T., Farkas, R., Foster, J., Freitas, C., Gajdošová, K., Galbraith, D., Garcia, M., Gärdenfors, M., Garza, S., Ginter, F., Goenaga, I., Gojenola, K., GökÄśrmak, M., Goldberg, Y., Gómez Guinovart, X., Gonzáles Saavedra, B., Grioni, M., Grūzı̄tis, N., Guillaume, B., Hajič, J., Hà Máżź, L., Haug, D., Hladká, B., Ion, R., Irimia, E., Johannsen, A., Jørgensen, F., KaşÄśkara, H., Kanayama, H., Kanerva, J., Katz, B., Kenney, J., Kotsyba, N., Krek, S., Laippala, V., Lam, L., Lê HáżŞng, P., Lenci, A., Ljubešić, N., Lyashevskaya, O., Lynn, T., Makazhanov, A., Manning, C., Mărănduc, C., Mareček, D., Martínez Alonso, H., Martins, A., Mašek, J., Matsumoto, Y., McDonald, R., Missilä, A., Mititelu, V., Miyao, Y., Montemagni, S., Mori, K. S., Mori, S., Moskalevskyi, B., Muischnek, K., Mustafina, N., Müürisep, K., NguyáżĚn tháżŃ, L., NguyáżĚn tháżŃ Minh, H., Nikolaev, V., Nurmi, H., Osenova, P., Östling, R., Øvrelid, L., Paiva, V., Pascual, E., Passarotti, M., Perez, C.-A., Petrov, S., Piitulainen, J., Plank, B., Popel, M., PretkalniÅĘa, L., Prokopidis, P., Puolakainen, T., Pyysalo, S., Rademaker, A., Ramasamy, L., Real, L., Rituma, L., Rosa, R., Saleh, S., Saulīte, B., Schuster, S., Seeker, W., Seraji, M., Shakurova, L., Shen, M., Silveira, N., Simi, M., Simionescu, R., Simkó, K., Šimková, M., Simov, K., Smith, A., Spadine, C., Suhr, A., Sulubacak, U., Szántó, Z., Tanaka, T., Tsarfaty, R., Tyers, F., Uematsu, S., Uria, L., van Noord, G., Varga, V., Vincze, V., Wallin, L., Wang, J. X., Washington, J. N., Wirén, M., Žabokrtský, Z., Zeldes, A., Zeman, D., and Zhu, H. (2016). Universal dependencies 1.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. 3, 6, 86, 89

Ohio Supercomputer Center (1987). Ohio supercomputer center. `http://osc.edu/ark:/19495/f5s1ph73`. 99

Ono, M., Miwa, M., and Sasaki, Y. (2015). Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 984–989. 37, 38, 39, 55

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(10):1345–1359. xii, 1, 2, 7, 84

Pavlick, E., Rastogi, P., Ganitkevich, J., Van Durme, B., and Callison-Burch, C. (2015). PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 425–430. 39, 55

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830. 116

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. 17, 38, 54, 78

Pham, N. T., Lazaridou, A., and Baroni, M. (2015). A multitask objective to inject lexical contrast into distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 21–26. 37, 38, 39, 55

Pham, V., Bluche, T., Kermorvant, C., and Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 285–290. 60, 78, 98

Plank, B., Søgaard, A., and Goldberg, Y. (2016). Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 412–418. 1, 83, 86, 98

Poon, H. and Domingos, P. (2011). Sum-product networks: A new deep architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*. 106

Ravuri, S. and Stolcke, A. (2015). A comparative study of neural network models for lexical intent classification. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 368–374. 56

Rei, M. (2017). Semi-supervised multitask learning for sequence labeling. In *Annual Meeting of the Association for Computational Linguistics (ACL)*. xvii, 88, 89, 90

Schwenk, H. (2007). Continuous space language models. *Computer Speech & Language*, 21(3):492–518. 11, 24

Shivade, C., de Marneffe, M.-C., Fosler-Lussier, E., and Lai, A. M. (2015). Corpus-based discovery of semantic intensity scales. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 483–493. 42, 43

Socher, R., Chen, D., Manning, C. D., and Ng, A. Y. (2013a). Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 926–934. 38

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., and Potts, C. (2013b). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642. 107

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Ess-Dykema, C. V., Martin, R., and Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373. 110, 115

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In *Advances in Neural Information Processing Systems 28 (NIPS)*, pages 2440–2448. 67

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1017–1024. 106

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 3104–3112. 20

Täckström, O., Das, D., Petrov, S., McDonald, R., and Nivre, J. (2013). Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics (TACL)*, 1:1–12. 84

Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. (2015). Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1509. 38

Tur, G. (2006). Multitask learning for spoken language understanding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 585–588. 70

Tur, G. and de Mori, R. (2011). *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. New York, NY: John Wiley and Sons. 55, 69

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics (ACL)*, pages 384–394. 2, 24

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *arXiv:1706.03762*. 1

Wang, H. (2010). `https://liqiangguo.files.wordpress.com/2011/06/lsi2.pdf`. xvi, 10

Wang, Z., Hamza, W., and Florian, R. (2017). Bilateral multi-perspective matching for natural language sentences. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 1

Webb, N., Hepple, M., and Wilks, Y. (2005). Dialogue act classification based on intra-utterance features. In *Proceedings of the AAAI Workshop on Spoken Language Understanding*. 116

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In *International Conference on Learning Representations (ICLR)*. 67

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2016). Towards universal paraphrastic sentence embeddings. In *International Conference on Learning Representations (ICLR)*. 19

Wieting, J., Bansal, M., Gimpel, K., Livescu, K., and Roth, D. (2015). From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL (TACL)*, 3:345–358. 18, 40

Wisniewski, G., Pécheux, N., Gahbiche-Braham, S., and Yvon, F. (2014). Cross-lingual part-of-speech tagging through ambiguous learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1779–1785. 84

Yang, Z., Salakhutdinov, R., and Cohen, W. W. (2017). Transfer learning for sequence tagging with hierarchical recurrent networks. In *International Conference on Learning Representations (ICLR)*. xii, xvii, 2, 7, 83, 84, 85, 86

Yoo, J. and Choi, S. (2008). `http://mlg.postech.ac.kr/research/nmf`. xvi, 9

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First Conference on Artificial Intelligence (AAAI)*, pages 2852–2858. 21

Zhang, X. and Wang, H. (2016). A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2993–2999. 56

Zhang, Y., Barzilay, R., and Jaakkola, T. (2017). Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics (TACL)*. 22, 89

Zhang, Y., Gaddy, D., Barzilay, R., and Jaakkola, T. (2016a). Ten pairs to tag – multilingual POS tagging via coarse mapping between embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1307–1317. 84

Zhang, Y., Gan, Z., and Carin, L. (2016b). Generating text via adversarial training. In *NIPS Workshop on Adversarial Training*. 21

Zhong, Z. and Ng, H. T. (2010). It makes sense: a wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83. 32