AN ABSTRACT OF THE DISSERTATION OF

Mingbo Ma for the degree of Doctor of Philosophy in Computer Science presented on May 25, 2018.

Title: Structured Neural Models for Natural Language Processing

Abstract approved: ______________________________

Liang Huang

Most tasks in natural language processing (NLP) involves structured information from both input (e.g., a sentence or a paragraph) and output (e.g., a tag sequence, a parse tree or a translated sentence). While neural models achieve great successes in other domains such as computer vision, applying those frameworks to NLP remains challenging for the following reasons. On the source side, we are dealing with input sentences with very complex structures. A simple local swap between two adjacent words could lead to opposite meanings. In addition, input sentences are often noisy as they are collected from real-world scenarios, e.g. online reviews or tweets. Our models need to be able to deal with syntactic variety and polysemy. On the target side, we are often expected to generate structured outputs like translated sentences or parse trees, by searching over an exponentially large search space. In this case, when exact search is intractable, we resort to inexact search methods such as beam search. In this thesis, we start by introducing several classification algorithms with structured information from the source side but unstructured outputs (sentence level classifications, e.g., sentiment analysis). Then we explore models which generate structured output from the unstructured input signal (e.g., image captioning). Finally, we investigate more complex frameworks that deal with structured information on both input and output sides (e.g., machine translation).

Structured Neural Models for Natural Language Processing

by
Mingbo Ma

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented May 25, 2018
Commencement June 2018

Doctor of Philosophy disseratation of Mingbo Ma presented on May 25, 2018.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my disseratation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my disseratation to any reader upon request.

Mingbo Ma, Author

ACKNOWLEDGEMENTS

This page is mostly here to demonstrate one of the pretext pages. However, there *are* a lot of people who deserve some credit for making the earlier incarnations of the osuthesis class possible. Unfortunately, I don't even know who they all are. I've gathered what names I could that were sprinkled around various source files to the original osuthesis class. One of the template files that I found acknowledges Dr. Gregg Rothermel of the Oregon State University Software Engineering Group, but states that "most of the credit for the real work in creating [one of the original versions], however, belongs to Lixin Li and Chengyun Chu." That file also states that it was modified by Alexey G. Malishevsky. I got these templates from Eric Altendorf, who in turn got them from Rogan Creswick, whence he got them, I don't know. Both Eric and Rogan apparently did a lot of work on what were a large number of templates and macros. Both of those intrepid explorers weeded through code a great deal to leave something better and easier for the next users. Some of the macros from those original templates are included here and have helped build this class— especially the table of contents formatting, which is the main section culled from those original templates. Some credit should also also go to Norman Gall of University of Calgary, who wrote a .cls file for Rayees Shamsuddin, who in turn sent it to me. I didn't actually use any of Norman's code (at least, I don't think I did), but the fact that he went through the trouble to create a LaTeX class for others to use is deserving of credit.

DEDICATION

To XX and XX.
Without them, would never have been possible.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

Chapter 1 –Using Structure Information for Sentence Level Classification

1.1 Motivation

In sentence modeling and classification, convolutional neural network approaches have recently achieved state-of-the-art results, but all such efforts process word vectors sequentially and neglect long-distance dependencies. To combine deep learning with linguistic structures, we propose a dependency-based convolution approach, making use of tree-based n-grams rather than surface ones, thus utilizing non-local interactions between words. Our model improves sequential baselines on all four sentiment and question classification tasks, and achieves the highest published accuracy on TREC.

1.1.1 Introduction

Convolutional neural networks (CNNs), originally invented in computer vision LeCun *et al.* (1995), has recently attracted much attention in natural language processing (NLP) on problems such as sequence labeling Collobert *et al.* (2011), semantic parsing Yih *et al.* (2014), and search query retrieval Shen *et al.* (2014). In particular, recent work on CNN-based sentence modeling Kalchbrenner *et al.* (2014); Kim (2014) has achieved excellent, often state-of-the-art, results on various classification tasks such as sentiment, subjectivity, and question-type classification. However, despite their celebrated success, there remains a major limitation from the linguistics perspective: CNNs, being invented on pixel matrices in image processing, only consider sequential n-grams that are consecutive on the surface string and neglect long-distance dependencies, while the latter play an important role in many linguistic phenomena such as negation, subordination, and *wh*-extraction, all of which might dully affect the sentiment, subjectivity, or other categorization of the sentence.

Indeed, in the sentiment analysis literature, researchers have incorporated long-distance information from syntactic parse trees, but the results are somewhat inconsis-

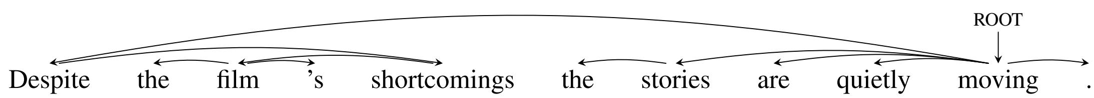


Figure 1.1: Dependency tree of an example sentence from the Movie Reviews dataset.

tent: some reported small improvements Gamon (2004); Matsumoto *et al.* (2005), while some otherwise Dave *et al.* (2003); Kudo and Matsumoto (2004). As a result, syntactic features have yet to become popular in the sentiment analysis community. We suspect one of the reasons for this is data sparsity (according to our experiments, tree n-grams are significantly sparser than surface n-grams), but this problem has largely been alleviated by the recent advances in word embedding. Can we combine the advantages of both worlds?

So we propose a very simple dependency-based convolutional neural networks (DCNNs). Our model is similar to Kim (2014), but while his sequential CNNs put a word in its sequential context, ours considers a word and its parent, grand-parent, great-grand-parent, and siblings on the dependency tree. This way we incorporate long-distance information that are otherwise unavailable on the surface string.

Experiments on three classification tasks demonstrate the superior performance of our DCNNs over the baseline sequential CNNs. In particular, our accuracy on the TREC dataset outperforms all previously published results in the literature, including those with heavy hand-engineered features.

1.2 Dependency-based Convolution

The original CNN, first proposed by LeCun *et al.* (1995), applies convolution kernels on a series of continuous areas of given images, and was adapted to NLP by Collobert *et al.* (2011). Following Kim (2014), one dimensional convolution operates the convolution kernel in sequential order in Equation 1.1, where $\mathbf{x}_i \in \mathbb{R}^d$ represents the d dimensional word representation for the i-th word in the sentence, and $\oplus$ is the concatenation operator. Therefore $\widetilde{\mathbf{x}}_{i,j}$ refers to concatenated word vector from the i-th word to the

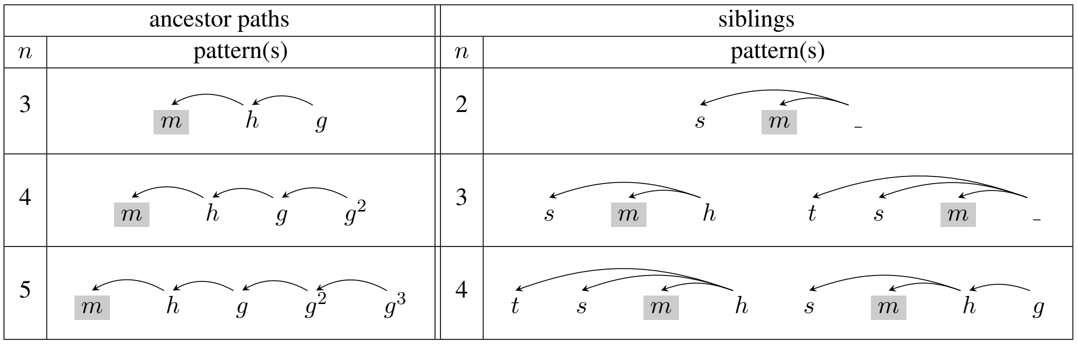


Figure 1.2: Convolution patterns on trees. Word concatenation always starts with m, while h, g, and g^2 denote parent, grand parent, and great-grand parent, etc., and "_" denotes words excluded in convolution.

$(i + j)$-th word:

$$\widetilde{\mathbf{x}}_{i,j} = \mathbf{x}_i \oplus \mathbf{x}_{i+1} \oplus \cdots \oplus \mathbf{x}_{i+j} \tag{1.1}$$

Sequential word concatenation $\widetilde{\mathbf{x}}_{i,j}$ works as n-gram models which feeds local information into convolution operations. However, this setting can not capture long-distance relationships unless we enlarge the window indefinitely which would inevitably cause the data sparsity problem.

In order to capture the long-distance dependencies we propose the dependency-based convolution model (DCNN). Figure 1.1 illustrates an example from the Movie Reviews (MR) dataset Pang and Lee (2005). The sentiment of this sentence is obviously positive, but this is quite difficult for sequential CNNs because many n-gram windows would include the highly negative word "shortcomings", and the distance between "Despite" and "shortcomings" is quite long. DCNN, however, could capture the tree-based bigram "Despite – shortcomings", thus flipping the sentiment, and the tree-based trigram "ROOT – moving – stories", which is highly positive.

1.2.1 Convolution on Ancestor Paths

We define our concatenation based on the dependency tree for a given modifier $\mathbf{x}_i$:

$$\mathbf{x}_{i,k} = \mathbf{x}_i \oplus \mathbf{x}_{p(i)} \oplus \cdots \oplus \mathbf{x}_{p^{k-1}(i)} \tag{1.2}$$

where function $p^k(i)$ returns the i-th word's k-th ancestor index, which is recursively defined as:

$$p^k(i) = \begin{cases} p(p^{k-1}(i)) & \text{if} \quad k > 0 \\ i & \text{if} \quad k = 0 \end{cases} \tag{1.3}$$

Figure 1.2 (left) illustrates ancestor paths patterns with various orders. We always start the convolution with x_i and concatenate with its ancestors. If the root node is reached, we add "ROOT" as dummy ancestors (vertical padding).

For a given tree-based concatenated word sequence $\mathbf{x}_{i,k}$, the convolution operation applies a filter $\mathbf{w} \in \mathbb{R}^{k \times d}$ to $\mathbf{x}_{i,k}$ with a bias term b described in equation 1.4:

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i,k} + b) \tag{1.4}$$

where f is a non-linear activation function such as rectified linear unit (ReLu) or sigmoid function. The filter $\mathbf{w}$ is applied to each word in the sentence, generating the feature map $\mathbf{c} \in \mathbb{R}^l$:

$$\mathbf{c} = [c_1, c_2, \cdots, c_l] \tag{1.5}$$

where l is the length of the sentence.

1.2.2 Max-Over-Tree Pooling and Dropout

The filters convolve with different word concatenation in Eq. 1.4 can be regarded as pattern detection: only the most similar pattern between the words and the filter could return the maximum activation. In sequential CNNs, max-over-time pooling Collobert *et al.* (2011); Kim (2014) operates over the feature map to get the maximum activation $\hat{c} = \max \mathbf{c}$ representing the entire feature map. Our DCNNs also pool the maximum activation from feature map to detect the strongest activation over the whole tree (i.e., over the whole sentence). Since the tree no longer defines a sequential "time" direction, we refer to our pooling as "max-over-tree" pooling. Another issue of sentence length variation are solved naturally by max-pooling.

In order to capture enough variations, we randomly initialize the set of filters to detect different structure patterns. Each filter's height is the number of words considered and the width is always equal to the dimensionality d of word representation. Each filter will be represented by only one feature after max-over-tree pooling. After a series of convolution with different filter with different heights, multiple features carry different structural information become the final representation of the input sentence. Then, this sentence representation is passed to a fully connected soft-max layer and outputs a distribution over different labels.

Neural networks often suffer from over-training. Following Kim (2014), we employ random dropout on penultimate layer Hinton *et al.* (2014). in order to prevent co-adaptation of hidden units. In our experiments, we set our drop out rate as 0.5 and learning rate as 0.95 by default. Following Kim (2014), training is done through stochastic gradient descent over shuffled mini-batches with the Adadelta update rule Zeiler (2012).

1.2.3 Convolution on Siblings

Ancestor paths alone is not enough to capture many linguistic phenomena such as conjunction. Inspired by higher-order dependency parsing McDonald and Pereira (2006); Koo and Collins (2010), we also incorporate siblings for a given word in various ways. See Figure 1.2 (right) for details.

1.2.4 Combined Model

Powerful as it is, structural information still does not fully cover sequential information. Also, parsing errors (which are common especially for informal text such as online reviews) directly affect DCNN performance while sequential n-grams are always correctly observed. To best exploit both information, we want to combine both models. The easiest way of combination is to concatenate these representations together, then feed into fully connected soft-max neural networks. In these cases, combine with different feature from different type of sources could stabilize the performance. The final sentence representation is thus:

$$\hat{\mathbf{c}} = [\underbrace{\hat{c}_a^{(1)}, ..., \hat{c}_a^{(N_a)}}_{\text{ancestors}}; \underbrace{\hat{c}_s^{(1)}, ..., \hat{c}_s^{(N_s)}}_{\text{siblings}}; \underbrace{\hat{c}^{(1)}, ..., \hat{c}^{(N)}}_{\text{sequential}}]$$

Category	Model	MR	SST-1	TREC	TREC-2
This work	DCNNs: ancestor	80.4†	47.7†	95.4†	88.4†
	DCNNs: ancestor+sibling	81.7†	48.3†	**95.6**†	89.0†
	DCNNs: ancestor+sibling+sequential	**81.9**	49.5	95.4†	88.8†
CNNs	CNNs-non-static Kim (2014) – baseline	81.5	48.0	93.6	86.4*
	CNNs-multichannel Kim (2014)	81.1	47.4	92.2	86.0*
	Deep CNNs Kalchbrenner *et al.* (2014)	-	48.5	93.0	-
Recursive NNs	Recursive Autoencoder Socher *et al.* (2011)	77.7	43.2	-	-
	Recursive Neural Tensor Socher *et al.* (2013)	-	45.7	-	-
	Deep Recursive NNs Irsoy and Cardie (2014)	-	**49.8**	-	-
Recurrent NNs	LSTM on tree Zhu *et al.* (2015)	**81.9**	48.0	-	-
Other deep learning	Paragraph-Vec Le and Mikolov (2014)	-	48.7	-	-
Hand-coded rules	SVM$_S$ Silva *et al.* (2011)	-		95.0	**90.8**

Table 1.1: Results on Movie Review (MR), Stanford Sentiment Treebank (SST-1), and TREC datasets. TREC-2 is TREC with fine grained labels. †Results generated by GPU (all others generated by CPU). *Results generated from Kim (2014)'s implementation.

where N_a, N_s, and N are the number of ancestor, sibling, and sequential filters. In practice, we use 100 filters for each template in Figure 1.2 . The fully combined representation is 1,100-dimensional by contrast to 300-dimensional for sequential CNN.

1.3 Experiments

Table 1.1 summarizes results in the context of other high-performing efforts in the literature. We use three benchmark datasets in two categories: sentiment analysis on both Movie Review (MR) Pang and Lee (2005) and Stanford Sentiment Treebank (SST-1) Socher *et al.* (2013) datasets, and question classification on TREC Li and Roth (2002).

For all datasets, we first obtain the dependency parse tree from Stanford parser Manning *et al.* (2014).[1] Different window size for different choice of convolution are shown in Figure 1.2. For the dataset without a development set (MR), we randomly choose 10% of the training data to indicate early stopping. In order to have a fare comparison with baseline CNN, we also use 3 to 5 as our window size. Most of our results are generated by GPU due to its efficiency, however CPU could potentially get

[1]The phrase-structure trees in SST-1 are actually automatically parsed, and thus can not be used as gold-standard trees.

better results.[2] Our implementation, on top of Kim (2014)'s code,[3] will be released.[4]

1.3.1 Sentiment Analysis

Both sentiment analysis datasets (MR and SST-1) are based on movie reviews. The differences between them are mainly in the different numbers of categories and whether the standard split is given. There are 10,662 sentences in the MR dataset. Each instance is labeled positive or negative, and in most cases contains one sentence. Since no standard data split is given, following the literature we use 10 fold cross validation to include every sentence in training and testing at least once. Concatenating with sibling and sequential information obviously improves DCNNs, and the final model outperforms the baseline sequential CNNs by 0.4, and ties with Zhu *et al.* (2015).

Different from MR, the Stanford Sentiment Treebank (SST-1) annotates finer-grained labels, very positive, positive, neutral, negative and very negative, on an extension of the MR dataset. There are 11,855 sentences with standard split. Our model achieves an accuracy of 49.5 which is second only to Irsoy and Cardie (2014).

1.3.2 Question Classification

In the TREC dataset, the entire dataset of 5,952 sentences are classified into the following 6 categories: abbreviation, entity, description, location and numeric. In this experiment, DCNNs easily outperform any other methods even with ancestor convolution only. DCNNs with sibling achieve the best performance in the published literature. DCNNs combined with sibling and sequential information might suffer from overfitting on the training data based on our observation. One thing to note here is that our best result even exceeds SVM_S Silva *et al.* (2011) with 60 hand-coded rules.

The TREC dataset also provides subcategories such as numeric:temperature, numeric:distance, and entity:vehicle. To make our task more realistic and challenging, we also test the proposed model with respect to the 50 subcategories. There are obvious improvements over sequential CNNs from the last column of Table 1.1. Like ours, Silva *et al.* (2011) is a tree-based system but it uses constituency trees compared to ours

[2] GPU only supports `float32` while CPU supports `float64`.

[3] `https://github.comw/yoonkim/CNN_sentence`

[4] `https://github.com/cosmmb/DCNN`

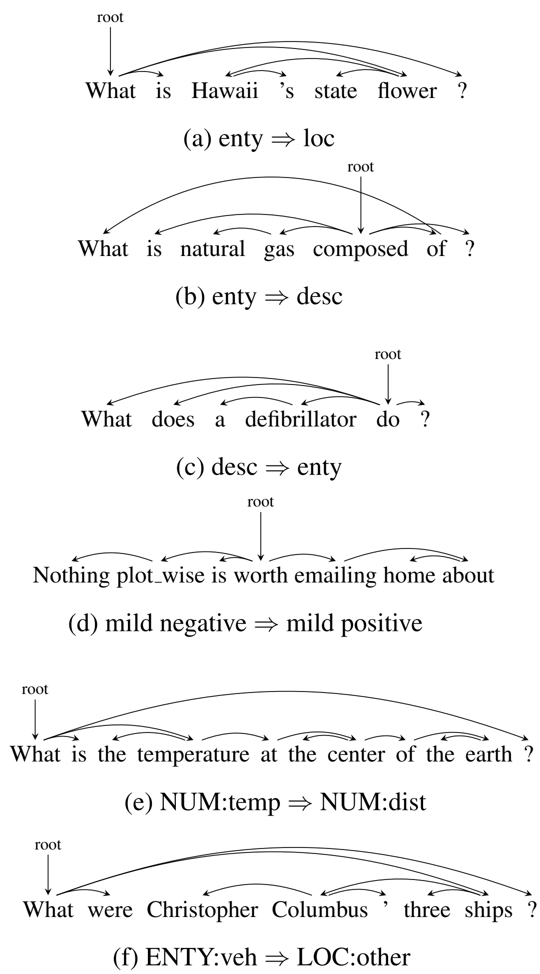
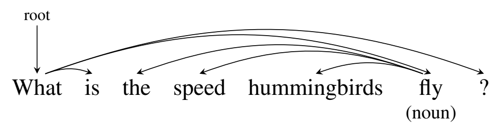
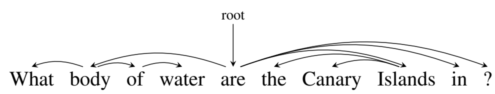


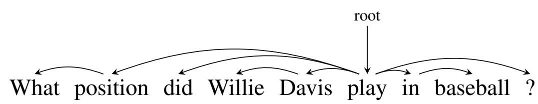
Figure 1.3: Examples from TREC (a–c), SST-1 (d) and TREC with fine-grained label (e–f) that are misclassified by the baseline CNN but correctly labeled by our DCNN. For example, (a) should be *entity* but is labeled *location* by CNN.



(a) num ⇒ enty



(b) loc ⇒ enty



(c) hum ⇒ enty

Figure 1.4: Examples from TREC datasets that are misclassified by DCNN but correctly labeled by baseline CNN. For example, (a) should be *numerical* but is labeled *entity* by DCNN.

dependency trees. They report a higher fine-grained accuracy of 90.8 but their parser is trained *only* on the QuestionBank Judge *et al.* (2006) while we used the standard Stanford parser trained on both the Penn Treebank and QuestionBank. Moreover, as mentioned above, their approach is rule-based while ours is automatically learned.

1.3.3 Discussions and Examples

Compared with sentiment analysis, the advantage of our proposed model is obviously more substantial on the TREC dataset. Based on our error analysis, we conclude that this is mainly due to the difference of the parse tree quality between the two tasks. In sentiment analysis, the dataset is collected from the *Rotten Tomatoes* website which includes many irregular usage of language. Some of the sentences even come from languages other than English. The errors in parse trees inevitably affect the classification accuracy. However, the parser works substantially better on the TREC dataset since all questions are in formal written English, and the training set for Stanford parser[5] already includes the QuestionBank Judge *et al.* (2006) which includes 2,000

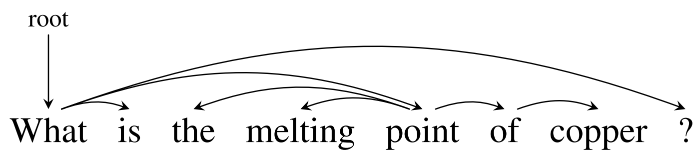
[5] `http://nlp.stanford.edu/software/parser-faq.shtml`

TREC sentences.

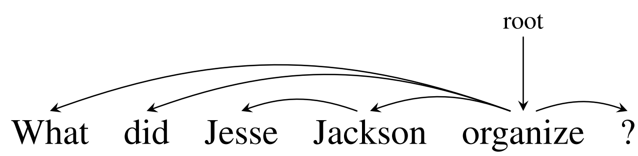
Figure 1.3 visualizes examples where CNN errs while DCNN does not. For example, CNN labels (a) as *location* due to “Hawaii” and “state”, while the long-distance backbone “What – flower” is clearly asking for an *entity*. Similarly, in (d), DCNN captures the obviously negative tree-based trigram “Nothing – worth – emailing”. Note that our model also works with non-projective dependency trees such as the one in (b). The last two examples in Figure 1.3 visualize cases where DCNN outperforms the baseline CNNs in fine-grained TREC. In example (e), the word “temperature” is at second from the top and is root of a 8 word span “the ... earth”. When we use a window of size 5 for tree convolution, every words in that span get convolved with “temperature” and this should be the reason why DCNN get correct.

Figure 1.4 showcases examples where baseline CNNs get better results than DCNNs. Example (a) is misclassified as *entity* by DCNN due to parsing/tagging error (the Stanford parser performs its own part-of-speech tagging). The word “fly” at the end of the sentence should be a verb instead of noun, and “hummingbirds fly” should be a relative clause modifying “speed”.

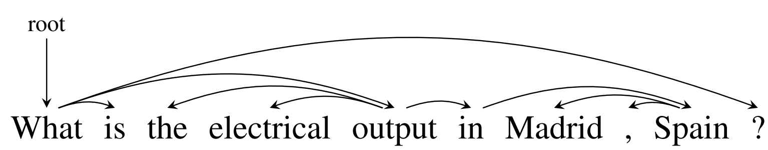
There are some sentences that are misclassified by both the baseline CNN and DCNN. Figure 1.5 shows three such examples. Example (a) is not classified as *numerical* by both methods due to the ambiguous meaning of the word “point” which is difficult to capture by word embedding. This word can mean location, opinion, etc. Apparently, the numerical aspect is not captured by word embedding. Example (c) might be an annotation error.



(a) num ⇒ enty and desc



(b) hum ⇒ enty and enty



(c) enty ⇒ num and num

Figure 1.5: Examples from TREC datasets that are misclassified by both DCNN and baseline CNN. For example, (a) should be *numerical* but is labeled *entity* by DCNN and *description* by CNN.

Chapter 2 –Jointly Training for Sequential Labeling and Sentence Level Classification

Sentence-level classification and sequential labeling are two fundamental tasks in language understanding. While these two tasks are usually modeled separately, in reality, they are often correlated, for example in intent classification and slot filling, or in topic classification and named-entity recognition. In order to utilize the potential benefits from their correlations, we propose a jointly trained model for learning the two tasks simultaneously via Long Short-Term Memory (LSTM) networks. This model predicts the sentence-level category and the word-level label sequence from the stepwise output hidden representations of LSTM. We also introduce a novel mechanism of "sparse attention" to weigh words differently based on their semantic relevance to sentence-level classification. The proposed method outperforms baseline models on ATIS and TREC datasets.

2.1 Introduction

We consider the dichotomy between two important tasks in spoken language understanding: the global task of sentence-level classification, such as intention or sentiment, and the local task of sequence labeling or semantic constituent extraction, such as slot filling or named-entity recognitions (NER). Conventionally, these two tasks are modeled separately, with algorithms such as SVM Boser *et al.* (1992) for the former, and Conditional Random Fields (CRF) Lafferty *et al.* (2001) or structured perceptron Collins (2002) for the latter.

In reality, however, these two tasks are often correlated. Consider the problems of sentence topic classification and NER in Figure 2.1. Different sentence-level classifications provide different priors for each word's label; for example if we know the sentence is about IT news then the word "Apple" is almost certainly about the company. Likewise, different word-level label sequence also influence the sentence-level category distribution; for example if we know the word "Apple" is about fruits then the sentence topic

Topic: *IT news*
Apple's new iPad will be out soon ... (Company)

Topic: *Agriculture reports*
Apple harvest in Washington state .... (Fruit)

Topic: *Tourist information*
"Big *Apple*" is a nickname of NYC ...(Location)

Figure 2.1: The same word "Apple" has different meanings (see tags in blue) in different different topics.

Category: Geography
Texas is located in the South Central region...

Category: Date or Time
The train leaves at nine o'clock

Category: Positive review
A smart, sweet and playful romantic comedy.

Figure 2.2: Examples of various magnitudes of attentions for sentence level classification. Darker words are more important.

is more likely to be agricultural.

Indeed, previous work has explored joint modeling between the two tasks. For example, Jeong and Lee Jeong and Lee (2008) propose a discrete CRF model for joint training of sentence-level classification and sequence labeling. A follow-up work Xu and Sarikaya (2013) leverages the feature representation power of convolution neural networks (CNNs) to make the CRF model generalize better for unseen data. However, the above CRF-based methods still suffer from the following limitations: firstly, although CRF is trained globally, it still lacks the ability for capturing long-term memory at each time step which is crucial for sentence-level classification and sequential labeling problem; secondly, the CNNs-based CRF Xu and Sarikaya (2013) only use CNNs for nonlinear local feature extraction. But globally, it is still a linear model which has limited generalization power to unseen data.

In order to overcome the two above issues, we propose a novel LSTM-based model to jointly train sentence-level classification and sequence labeling, which incorporates long-term memory and nonlinearity both locally and globally. We make the following contributions:

- Our Long Short-Term Memory (LSTM) Hochreiter and Schmidhuber (1997) network analyzes the sentence using features extracted by a convolutional layer. Basically, each word-level label is greedily determined by the hidden representation from LSTM in each step, and the global sentence category is determined by an aggregate (e.g., pooling) of hidden representations from all steps (Sec. 2.3.1).

- We also propose a novel **sparse attention model** which promotes important words in a given sentence and demotes semantically vacuous words (Fig. 2.2; Sec. 2.3.2).
- Finally, we develop a **latent variable** version of our jointly trained model which can be trained for the single task of sentence classification by treating word-level labels as latent information (Sec. 2.3.3).

2.2 LSTM for Labeling and Classification

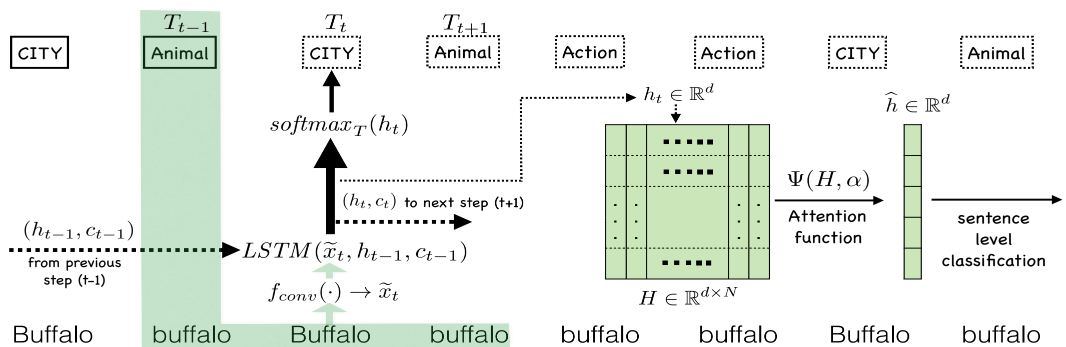


Figure 2.3: Jointly trained sequence labeling and sentence classification. The green mask means that convolution operate between one previous tag and two surrounding words (when window size is 3). Ψ is the attention function in Eq. 2.5. Here the sentence length is N=8.

We use LSTM to model the representation of the sentence at each word step, which is powerful in modeling sentence semantics Sutskever *et al.* (2014a); Cheng *et al.* (2016a). Assume the length of the sentence is N. LSTM represents the meaning of the sentence at the t-th word by a pair of vectors $(h_t, c_t) \in (\mathbb{R}^d, \mathbb{R}^d)$, where h_t is the output hidden representation of the word, c_t is the memory of the network, and d is the number of dimensions of the representation space:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W_{\text{LSTM}} \cdot [x_t, h_{t-1}] \tag{2.1}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{2.2}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.3}$$

Here x_t represents the t-th word, which is usually the word embedding vector, and vectors i_t, f_t, and o_t are gated activations that control flow of hidden information. The separation of the output representation h_t from its internal memory c_t, in principle, makes the knowledge about the sentence prefix be remembered by the network for longer time to interfere with the current output at word x_t. These carefully designed activation gates alleviate the problem of vanishing gradient problem in vanilla recurrent neural network models.

In the task of sequence labeling, the label for each word is determined by hidden representation h_t. As described in Eq. 2.3, at time step t, we will get an output h_t which represents all the current information. The probability distribution of the t-th word's label is calculated by $\mathit{softmax}_T(W_T \cdot h_t)$, where weight $W_T \in \mathbb{R}^{d\times|T|}$ maps h_t to the space of the labels, and $|T|$ is the number of possible labels. For the sequence labeling problem, the loss ℓ_{seq} is calculates as the sum of the Negative Log-Likelihood (NLL) over this label distribution $\mathit{softmax}_T(h_t)$ at each time step.

In the task of sentence classification, the entire sentence representation is obtained by aggregating all history outputs h_t which are stored in $H \in \mathbb{R}^{d\times N}$, where N is the length of sequence. Similar to CNNs, max pooling Collobert *et al.* (2011); Kim (2014); Ma *et al.* (2015) operates over the history outputs H to get the average activation $\hat{h} = \mathit{pooling}(H)$ summarizing the entire outputs. Then, this sentence representation is passed to a fully connected soft-max layer which outputs a distribution over sentence categories.

In above two different tasks, the hidden representation h_t functions as a key component in different, separate ways. In many cases, when sequence-level labels and sentence categories are both available we should use both information within the same framework by joint training the two tasks.

Table 2.1: Examples of ATIS sentences and annotated slots and categories.

Sentence 1	I	want	to	go	from	Denver	to	Boston	today	Category
Slots	O	O	O	O	O	B-FromCity	O	B-ToCity	B-Date	Flight
Sentence 2	to	come	back	to	Los	Angeles	on	Friday	evening	Category
Slots	O	O	O	O	B-ReturnCity	I-ReturnCity	O	B-RETURN.DAY	B-RETURN.PERIODOFDAY	Return

2.3 Joint Sequence Classification & Labeling

2.3.1 Joint Training Model

We aim at developing a model which could learn the label sequence and sentence-level category simultaneously. To this end, we modify the standard LSTM structure to generate the word labels on the fly based on output h_t, and predict the sentence category with the sequence of h_t, $t = 1, \ldots, N$.

In LSTM, at time step t, only information of the current word x_t is being fed into the network. This mechanism overlooks the problem that the meanings of the same word in different contexts might vary (*cf.* Fig. 2.1). In particular, words that follow x_t are not represented at step t in LSTM.

This observation motivates us to include more contextual information around the current word and previous tags as part of the input to LSTM. We employ convolutional neural network (CNN) to automatically mine the meaningful knowledge from both the context of word x_t and previous tags T_{t-1}, and use this knowledge as the input for LSTM. We formally define the new input for LSTM as:

$$\widetilde{x}_t = f_{\text{conv}}(W_{\text{conv}} \cdot x_{t,k} + b_{\text{conv}}) \tag{2.4}$$

where f_{conv} is a non-linear activation function like rectified linear unit (ReLU) or sigmoid, and $x_{t,k}$ is a vector representing the context of word x_t and previous tags $T_{t-k}...T_{t-1}$, e.g., the concatenation of the surrounding words and previous tags, $x_{t,k} = [x_{t-k}, \ldots, x_{t+k}, T_{t-k}, \ldots, T_{t-1}]$ in Eq. 2.4, where the convolution window size is $2k+1$ and T_{t-1} represents the embedding for tag T_{t-1}. W_{conv} is the collection of filters applied on the context. During the convolution, each row of W_{conv} is a filter that will be fired if it matches some useful pattern in the input context. The convolutional layer functions as a feature extraction tool to learn meaningful representation from both words and tags automatically. Note that the above contextual representation is different from bi-

LSTM which only learns surrounding contextual of a given word. However, our model can learn both contextual and label information by convolution.

In our model, the joint training between sentence-level classification and sequential labeling is done in two directions: in forward pass, word label representation sequence is used for sentence level classification; during backward training, the sentence level prediction errors also fine-tune the label sequence.

Fig. 2.3 illustrates our proposed model with one classical NLP exemplary sentence which only contains the word "buffalo" as the running example. At each time step, we first use the convolutional layer as a feature extractor to get the nonlinear feature combinations from the embeddings of words and tags. In the case of window size equaling to 1, the convolution operates over the $t-1$, t and $t+1$ words and the $t-1$ tag. The contextual representation $x_{t,1} = [x_{t-1}, x_t, x_{t+1}, T_{t-1}]$ is then fed into the convolution layer to find feature representation $\widetilde{x}_t$ following Eq. 2.4. $\widetilde{x}_t$ is used as the input for the following LSTM to generate h_t and c_t, based on history information h_{t-1} and previous cell information c_{t-1}.

The example in Fig 2.3 is a grammatically correct sentence in English Rapaport (2012). The word "buffalo" has three different meanings: Buffalo, NY (city), bison (animal), or bully (action). It is hard for standard LSTM to differentiate the different meanings of "buffalo" in different time step since the x_t is the same all the time. However, in our case, instead of simply using word representation x_t itself, we also consider the contextual information with their tags through convolution from $x_{t,k}$ (Eq. 2.4).

2.3.2 Sparse Attention

In the previous section, the sentence-level representation $\hat{h}$ is obtained by a simple average pooling on H. This process assumes every words contribute to the sentence equally, which is not the case in many scenarios. Fig. 2.2 shows a few examples of different words with different magnitudes of attention in different sentence categories. In order to incorporate these differences into consideration, we further propose a novel sparse attention constraint for sentence level classification. The sparse constraint assigns bigger weights for important words and lower the weights or even totally ignores the less meaningful words such like "the" or "a". The attention-based sentence-level representation is formulated as follows:

$$\widehat{h} = \Psi(H, \alpha) = \sum_{h_t \in H} \psi(h_t \cdot \alpha) h_t \tag{2.5}$$

where $\alpha \in \mathbb{R}^d$ is an attention measurement which decides the importances for different inputs based on their semantics h_i. This importance is calculated through a nonlinear function ψ which can be sigmoid or ReLU and we use sigmoid in our case.

Sparse Autoencoders Ng (2011); Makhzani and Frey (2014) show that getting sparse representations in the hidden layers can further improve the performance. In our model, we apply similar sparse constraints by first calculating the average attention over the training samples in the same batch:

$$\hat{\rho}_t = \frac{1}{m} \sum_{i=1}^{m} \psi(h_t^i \cdot \alpha) \tag{2.6}$$

where m is the size of training batch, and h_t^i is the output of LSTM at step t for example i. In order to keep the above attention within a fix budget, similar to Sparse Autoencoders Ng (2011), we have an extra penalty term as follows:

$$KL(\rho || \hat{\rho}_t) = \rho \log \frac{\rho}{\hat{\rho}_t} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_t},$$

where ρ is the desired sparsity of the attention. This penalty term uses KL divergence to measure the difference between two distributions. Then our new objective is defined as follows:

$$\ell_{\text{sparse}}(\cdot) = \ell_{\text{seq}}(\cdot) + \ell_{\text{sent}}(\cdot) + \beta \sum_{t=1}^{N} KL(\rho || \hat{\rho}_t), \tag{2.7}$$

where N is the number of hidden units, ℓ_{seq} is the sequence labeling loss, and ℓ_{sent} is the sentence classification loss. β controls the weight of the sparsity penalty term. Note that the term $\hat{\rho}_t$ is implicitly controlled by optimizing α and output h_t.

2.3.3 Label Sequence as Latent Variable

In practice, it is expensive to annotate the data with both sequential label and sequence category. In many cases, the sequence labels are missing since it requires significantly more efforts to annotate the labels word-by-word. However, even without this sequence labeling information, it is still helpful if we could utilize the possible hidden labels for each words.

In our proposed model, we could consider the sentence-level classification task as the major learning objective and treat the unknown sequence labels as latent information. The only adaptation we need to make is to replace the T_t (tag embedding) with h_t (output at time step t) in $x_{t,k}$. In this case, we exploit the latent meaning representation to further improve the feature extraction of the convolutional layer.

2.4 Experiments

We start by evaluating the performance on a conventional joint learning task (Sec. 2.4.1). Then we show the performance when we treat the label sequence as hidden information in Sec. 2.4.2. We also analyze some concrete examples to show the performance of the sparse attention constraint in our model (Sec. 2.4.3).

In the experiments, we set the convolution window sizes as 3, 5, and 7. There are 100 different filters for each window size. Word embeddings are randomly initialized in the ATIS experiments. In the TREC experiments, we use 300 dimension pre-trained word embeddings. We use AdaDelta for optimization Zeiler (2012) with learning rate 0.001 and minibatch 16. The weights in our framework are uniformly randomly initialized between $[-0.05, 0.05]$. We use ReLu at the convolutional layer and also regularize the feature with dropout 0.5 Hinton *et al.* (2014).

We evaluate on ATISJeong and Lee (2008) and TREC Li and Roth (2002) datasets. We follow the TriCRF paperJeong and Lee (2008) in our evaluation on ATIS. There are $5,138$ dialogs with 21 types of intents and 110 types of slots annotated [1]. This dataset is first used for joint learning model with both slot and intent labels in the joint learning experiments (Sec. 2.4.1). We later use it for evaluating the performance when the slot labels are not available (Sec. 2.4.2). The TREC dataset[2] is a factoid

---

[1] Note we do not compare our performance with Guo *et al.* (2014) since their ATIS dataset is not published and is different from our ATIS dataset.

[2] http://cogcomp.cs.illinois.edu/Data/QA/QC/

Model	Sent. Acc.
CNN non-static Kim (2014)	93.6
CNN multichannel Kim (2014)	92.2
Deep CNN Kalchbrenner *et al.* (2014)	93.0
Independent LSTM (baseline)	92.2
latent LSTM + CNN	92.6
latent LSTM + CNN + attention	93.4
latent LSTM + CNN + sparse att.	**94.0**

Figure 2.4: Sentence-level accuracy of our latent-variable model on TREC, compared with various neural network-based models.

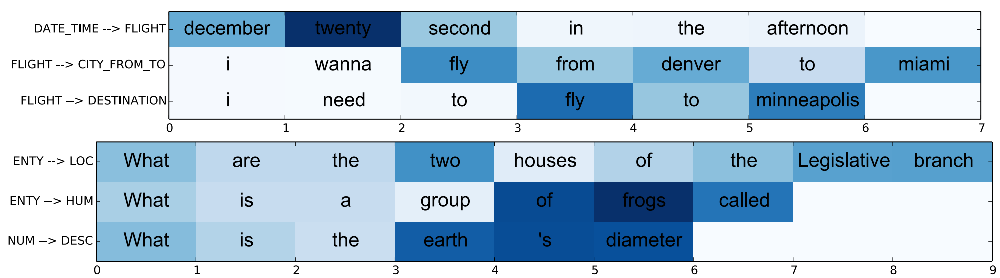


Figure 2.5: Examples that we outperform the model without sparse attention (LSTM+CNN). Higher weights are darker.

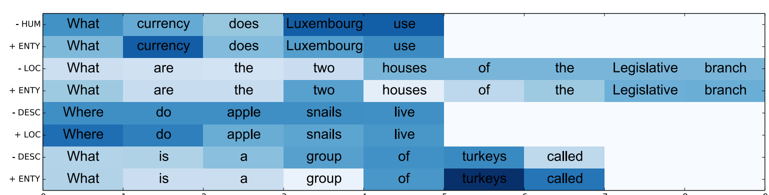


Figure 2.6: Comparison between softmax-based attention (upper) and sparse attention (lower) for for some examples. The sign '-' means mis-classified label, and '+' for the correct label. Darker blue represents higher weights.

question classification dataset, with 5,952 sentences being classified into 6 categories. Since only the sentence-level categories are annotated, we treat the unknown tags as latent information in our experiments (Sec. 2.4.2).

2.4.1 Joint Training Experiments

We first perform the joint training experiments on the ATIS dataset. Tab. 2.1 shows two examples from the ATIS dataset. As mentioned in Sec. 2.3.2, only a few keywords in the two sentences in Tab. 2.1 are relevant to determining the category, i.e., locations (cities) and date for sentence I, and locations, date, and time for sentence II. Our model should be able to recognize these important keywords and predict the categories mostly based on them. Once the model knows the tags of the words in the sentence, it is straightforward to determine the categories of the sentence.

We show the performance of our model comparing with other individually trained or jointly trained models in Tab. 2.2. We observe that due to its strong generalization power, neural networks based models outperform discrete models with an impressive margin. Our jointly trained neural model achieves the best performance, with an F1 boost of $\sim$2 points in slot filling. After adding the sparse attention constraint, our model achieves the lowest error rate for the sentence classification on ATIS.

2.4.2 Label Sequence as Latent Variable

We further show the performance of our jointly trained model when the sequence label information is missing. Tab. 2.2 (bottom) shows the results on ATIS dataset. There is a small increase of error rate when sequence labels are unobserved, but our model still outperforms existing models. Similarly, Fig. 2.4 compares our latent-variable model with conventional (non-latent) neural models on TREC dataset (in sentence category accuracy). Our model outperforms others after adding sparse attention.

2.4.3 Sparsity Visualization

In Fig. 2.5, we compare the sparse attention model to the model without sparse constraints. We list a few examples that the sparse attention is better than the one without sparse attention constraint. The labels on the right side are mis-classified by the model without sparse attention constraint. The label on the left side of the arrow is the ground truth.

In Fig. 2.6, we also compare the difference between two attention mechanisms: softmax-based attetion and sparse attention. From the first sentence, we can tell that softmax-based attention puts more emphasis on "Luxembourg" while sparse attention prefers "currency" which leads to the correct prediction of entity instead of human. In some cases, like the third example in Fig. 2.6, softmax-based sometimes gets confused by distributing the probabilities flatly. Compared with the attention model between dual sentences, the phenomenon of flat distribution is more obvious in single sentence attention. Similar results can be found in the first figure in Cheng *et al.* (2016b) as the word "run" is aligned to many unrelated words. It is possible that in single sentence attention, the softmax-based attention is easier to get confused since there is no obvious alignment or corresponding relationship between words for a given sentence or the words and their corresponding sentence category.

Table 2.2: **Main results:** Our jointly trained models compared with various independent (marked †) and existing joint models. The "Slot" column shows the F1 score of sequence labeling, and "intent" shows the error rates for sentence classification.

	Model	Slot	Intent
Independent Model	CRF Jeong and Lee (2008)	90.67†	7.91†
	CNN Xu and Sarikaya (2013)	92.43†	6.65†
CRF Joint Model	TriCRF Jeong and Lee (2008)	94.42	6.93
	CNN TriCRF Xu and Sarikaya (2013)	95.42	5.91
Independent Model	Vanilla LSTM (baseline)	93.74†	7.21†
Jointly Trained Model (Secs. 2.3.1 & 2.3.2)	+ joint	95.54	6.32
	+ joint + CNN	**97.35**	5.96
	+ joint + CNN + attention	96.73	5.71
	+ joint + CNN + sparse att.	96.98	**5.12**
Independent Model	Vanilla LSTM (baseline)	-	7.21†
Seq. Label as Latent Var. (Sec. 2.3.3)	+ joint + CNN	-	6.43
	+ joint + CNN + attention	-	5.61
	+ joint + CNN + sparse att.	-	**5.42**

Chapter 3 –Group Sparse Convolutional Neural Networks for Learning Label Hierarchy

Question classification is an important task with wide applications. However, traditional techniques treat questions as general sentences, ignoring the corresponding answer data. In order to consider answer information into question modeling, we first introduce novel group sparse autoencoders which refine question representation by utilizing group information in the answer set. We then propose novel group sparse CNNs which naturally learn question representation with respect to their answers by implanting group sparse autoencoders into traditional CNNs. The proposed model significantly outperform strong baselines on four datasets.

3.1 Problems in Question Answering

Question classification has applications in many domains ranging from question answering to dialog systems, and has been increasingly popular in recent years. Several recent efforts Kim (2014); Kalchbrenner *et al.* (2014); Ma *et al.* (2015) treat questions as general sentences and employ Convolutional Neural Networks (CNNs) to achieve remarkably strong performance in the TREC question classification task.

We argue, however, that those general sentence modeling frameworks neglect two unique properties of question classification. First, different from the flat and coarse categories in most sentence classification tasks (i.e. sentimental classification), question classes often have a hierarchical structure such as those from the New York State DMV FAQ[1] (see Fig. 3.1). Another unique aspect of question classification is the well prepared answers for each question or question category. These answer sets generally cover a larger vocabulary (than the questions themselves) and provide richer information for each class. We believe there is a great potential to enhance question representation with extra information from corresponding answer sets.

[1] Crawled from `http://nysdmv.custhelp.com/app/home`. This data and our code will be at `http://github.com/cosmmb`.

1: Driver License/Permit/Non-Driver ID
a: *Apply for original* (49 questions)
b: *Renew or replace* (24 questions)
...
2: Vehicle Registrations and Insurance
a: *Buy, sell, or transfer a vehicle* (22 questions)
b: *Reg. and title requirements* (42 questions)
...
3: Driving Record / Tickets / Points
...

Figure 3.1: Examples from NYDMV FAQs. There are 8 top-level categories, 47 sub-categories, and 537 questions (among them 388 are *unique*; many questions fall into multiple categories).

To exploit the hierarchical and overlapping structures in question categories and extra information from answer sets, we consider dictionary learning Candè and Wakin (2008); Rubinstein *et al.* (2010) which is a common approach for representing samples from many correlated groups with external information. This learning procedure first builds a dictionary with a series of grouped bases. These bases can be initialized randomly or from external data (from the answer set in our case) and optimized during training through Sparse Group Lasso (SGL) Simon *et al.* (2013).

To apply dictionary learning to CNN, we first develop a neural version of SGL, *Group Sparse Autoencoders* (GSAs), which to the best of our knowledge, is the first full neural model with group sparse constraints. The encoding matrix of GSA (like the dictionary in SGL) is grouped into different categories. The bases in different groups can be either initialized randomly or by the sentences in corresponding answer categories. Each question sentence will be reconstructed by a few bases within a few groups. GSA can use either linear or nonlinear encoding or decoding while SGL is restricted to be linear. Eventually, to model questions with sparsity, we further propose novel *Group Sparse Convolutional Neural Networks* (GSCNNs) by implanting the GSA onto CNNs, essentially enforcing group sparsity between the convolutional and classification layers. This framework is a jointly trained neural model to learn question representation with group sparse constraints from both question and answer sets.

3.2 Group Sparse Autoencoders

3.2.1 Sparse Autoencoders

Autoencoder Bengio *et al.* (2007) is an unsupervised neural network which learns the hidden representations from data. When the number of hidden units is large (e.g., bigger than input dimension), we can still discover the underlying structure by imposing sparsity constraints, using sparse autoencoders (SAE) Ng (2011):

$$J_{\text{sparse}}(\rho) = J + \alpha \sum_{j=1}^{s} KL(\rho \| \hat{\rho}_j) \tag{3.1}$$

where J is the autoencoder reconstruction loss, ρ is the desired sparsity level which is small, and thus $J_{\text{sparse}}(\rho)$ is the sparsity-constrained version of loss J. Here α is the weight of the sparsity penalty term defined below:

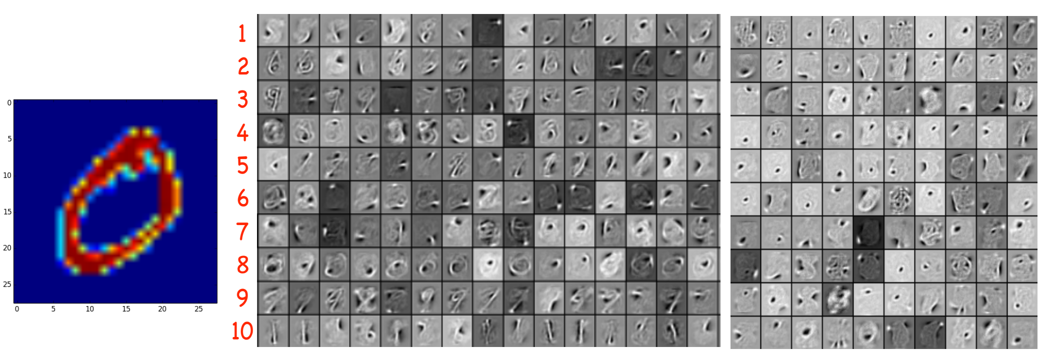
$$KL(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \tag{3.2}$$

where

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} h_j^i$$

represents the average activation of hidden unit j over m examples (SAE assumes the input features are correlated).

As described above, SAE has a similar objective to traditional sparse coding which tries to find sparse representations for input samples. Besides applying simple sparse constraints to the network, group sparse constraints is also desired when the class categories are structured and overlapped. Inspired by group sparse lasso Yuan and Lin (2006) and sparse group lasso Simon *et al.* (2013), we propose a novel architecture below.



(a) (b) (c)

Figure 3.2: The input figure with hand written digit 0 is shown in (a). Figure (b) is the visualization of trained projection matrix $\mathbf{W}$ on MNIST dataset. Different rows represent different groups of $\mathbf{W}$ in Eq. 3.5. For each group, we only show the first 15 (out of 50) bases. The red numbers on the left side are the indices of 10 different groups. Figure (c) is the projection matrix from basic autoencoders.The differences are easier to see in pdf.

3.2.2 Group Sparse Autoencoders

Group Sparse Autoencoder (GSA), unlike SAE, categorizes the weight matrix into different groups. For a given input, GSA reconstructs the input signal with the activations from only a few groups. Similar to the average activation $\hat{\rho}_j$ for sparse autoencoders, GSA defines each grouped average activation for the hidden layer as follows:

$$\hat{\eta}_p = \frac{1}{mg} \sum_{i=1}^{m} \sum_{l=1}^{g} \|h_{p,l}^{i}\|_2 \tag{3.3}$$

where g represents the size of each group, and $\hat{\eta}_j$ first sums up all the activations within p^{th} group, then computes the average p^{th} group respond across different samples' hidden activations.

Similar to Eq. 3.2, we also use KL divergence to measure the difference between estimated intra-group activation and global group sparsity:

$$KL(\eta \| \hat{\eta}_p) = \eta \log \frac{\eta}{\hat{\eta}_p} + (1-\eta) \log \frac{1-\eta}{1-\hat{\eta}_p} \tag{3.4}$$

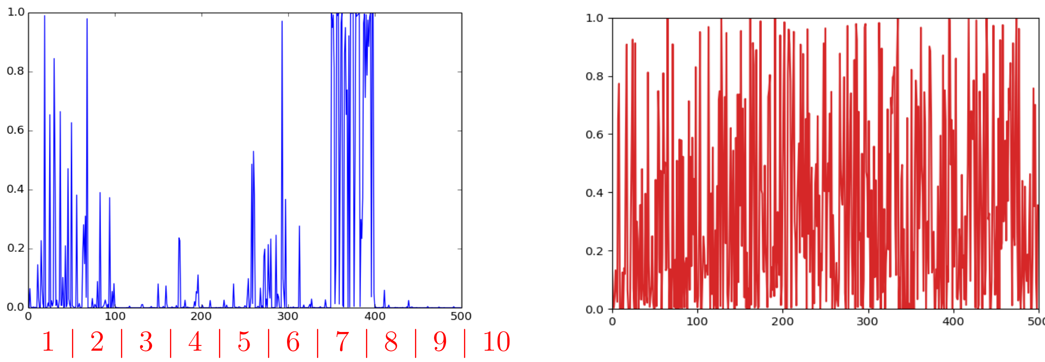
where G is the number of groups. Then the objective function of GSA is:

$$\begin{aligned} J_{\text{groupsparse}}(\rho, \eta) = J &+ \alpha \sum_{j=1}^{s} KL(\rho \| \hat{\rho}_j) \\ &+ \beta \sum_{p=1}^{G} KL(\eta \| \hat{\eta}_p) \end{aligned} \tag{3.5}$$

where ρ and η are constant scalars which are our target sparsity and group-sparsity levels, resp. When α is set to zero, GSA only considers the structure between difference groups. When β is set to zero, GSA is reduced to SAE.

3.2.3 Visualizing Group Sparse Autoencoders

In order to have a better understanding of GSA, we use the MNIST dataset to visualize GSA's internal parameters. Fig. 3.2 and Fig. 3.3 illustrate the projection matrix and the corresponding hidden activations. We use 10,000 training samples. We set the size



(a) (b)

Figure 3.3: (a): the hidden activations $\mathbf{h}$ for the input image in Fig. 3.2(a). The red numbers corresponds to the index in Fig. 3.2(b). (b): the hidden activations $\mathbf{h}$ for the same input image from basic autoencoders.

of the hidden layer to 500 with 10 groups. Fig. 3.2(a) visualizes the input image for hand written digit 0.

In Fig. 3.2(b), we find similar patterns within each group. For example, group 8 has different forms of digit 0, and group 9 includes different forms of digit 7. However, it is difficult to see any meaningful patterns from the projection matrix of basic autoencoders in Fig. 3.2(c).

Fig. 3.3(a) shows the hidden activations with respect to the input image of digit 0. The patterns of the 10^{th} row in Fig. 3.2(b) are very similar to digit 1 which is very different from digit 0 in shape. Therefore, there is no activation in group 10 in Fig. 3.3(a). The majority of hidden layer activations are in groups 1, 2, 6 and 8, with group 8 being the most significant. When compared to the projection matrix visualization in Fig. 3.2(b), these results are reasonable since the 8^{th} row has the most similar patterns of digit 0. However, we could not find any meaningful pattern from the hidden activations of basic autoencoder as shown in Fig. 3.3(b).

GSA could be directly applied to small image data (e.g. MINIST dataset) for pre-training. However, in tasks which prefer dense semantic representations (e.g. sentence classification), we still need CNNs to learn the sentence representation automatically. In order to combine advantages from GSA and CNNs, we propose Group Sparse Convolutional Neural Networks below.

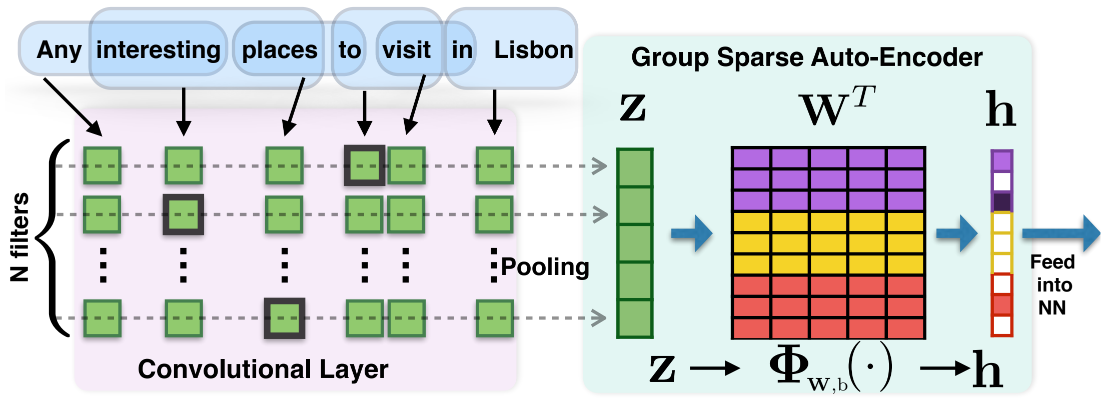


Figure 3.4: Group Sparse CNN. We add an extra dictionary learning layer between sentence representation $\mathbf{z}$ and the final classification layer. $\mathbf{W}$ is the projection matrix (functions as a dictionary) that converts $\mathbf{z}$ to the group sparse representation $\mathbf{h}$ (Eq. 3.5). Different colors in the projection matrix represent different groups. We show $\mathbf{W}^\intercal$ instead of $\mathbf{W}$ for presentation purposes. Darker colors in $\mathbf{h}$ mean larger values and white means zero.

3.3 Group Sparse CNNs

CNNs were first proposed by LeCun *et al.* (1995) in computer vision and adapted to NLP by Collobert *et al.* (2011). Recently, many CNN-based techniques have achieved great successes in sentence modeling and classification Kim (2014); Kalchbrenner *et al.* (2014).

Following sequential CNNs, one dimensional convolutions operate the convolution kernel in sequential order $\mathbf{x}_{i,j} = \mathbf{x}_i \oplus \mathbf{x}_{i+1} \oplus \cdots \oplus \mathbf{x}_{i+j}$, where $\mathbf{x}_i \in \mathbb{R}^e$ represents the e dimensional word representation for the i-th word in the sentence, and $\oplus$ is the concatenation operator. Therefore $\mathbf{x}_{i,j}$ refers to concatenated word vector from the i-th word to the $(i+j)$-th word in sentence.

A convolution operates a filter $\mathbf{w} \in \mathbb{R}^{n\times e}$ to a window of n words $\mathbf{x}_{i,i+n}$ with bias term b' by $a_i = \sigma(\mathbf{w}\cdot\mathbf{x}_{i,i+n} + b')$ with non-linear activation function σ to produce a new feature. The filter $\mathbf{w}$ is applied to each word in the sentence, generating the feature map $\mathbf{a} = [a_1, a_2, \cdots, a_L]$ where L is the sentence length. We then use $\hat{a} = \max\{\mathbf{a}\}$ to represent the entire feature map after max-pooling.

In order to capture different aspects of patterns, CNNs usually randomly initialize a set of filters with different sizes and values. Each filter will generate a feature as described above. To take all the features generated by N different filters into count, we use $\mathbf{z} = [\hat{a_1}, \cdots, \hat{a_N}]$ as the final representation. In conventional CNNs, this $\mathbf{z}$ will be directly fed into classifiers after the sentence representation is obtained, e.g. fully connected neural networks Kim (2014). There is no easy way for CNNs to explore the possible hidden representations with underlaying structures.

In order to exploit these structures, we propose Group Sparse Convolutional Neural Networks (GSCNNs) by placing one extra layer between the convolutional and the classification layers. This extra layer mimics the functionality of GSA from Section 3.2. Shown in Fig. 3.4, after the conventional convolutional layer, we get the feature map $\mathbf{z}$ for each sentence. In stead of directly feeding it into a fully connected neural network for classification, we enforce the group sparse constraint on $\mathbf{z}$ in a way similar to the group sparse constraints on hidden layer in GSA from Sec. 3.2. Then, we use the sparse hidden representation $\mathbf{h}$ in Eq. 3.5 as the new sentence representation, which is then fed into a fully connected neural network for classification. The parameters $\mathbf{W}$ in Eq. 3.5 will also be fine tunned during the last step.

Different ways of initializing the projection matrix in Eq. 3.5 can be summarized below:

- **Random Initialization**: When there is no answer corpus available, we first randomly initialize N vectors to represent the group information from the answer set. Then we cluster these N vectors into G categories with g centroids for each category. These centroids from different categories will be the initialized bases for projection matrix $\mathbf{W}$ which will be learned during training.

- **Initialization from Questions**: Instead of using random initialized vectors, we can also use question sentences for initializing the projection matrix when the answer set is not available. We need to pre-train the sentences with CNNs to get the sentence representation. We then select G largest categories in terms of number of question sentences. Then we get g centroids from each category by k-means. We concatenate these $G \times g$ vectors to form the projection matrix.

- **Initialization from Answers**: This is the most ideal case. We follow the same procedure as above, with the only difference being using the answer sentences in

Datasets	C_t	C_s	N_{data}	N_{test}	N_{ans}	Multi-label
TREC	6	50	5952	500	-	No
INSURANCE	-	319	1580	303	2176	Yes
DMV	8	47	388	50	2859	Yes
YAHOO Ans	27	678	8871	3027	10365	No

Table 3.1: Summary of datasets. C_t and C_s are the numbers of top-level and sub-categories, resp. N_{data}, N_{test}, N_{ans} are the sizes of data set, test set, and answer set, resp. Multilabel means each question can belong to multiple categories.

place of question sentences to pre-train the CNNs.

3.4 Experiments

Since there is little effort to use answer sets in question classification, we did not find any suitable datasets which are publicly available. We collected two datasets ourselves and also used two other well-known ones. These datasets are summarized in Table 3.1. INSURANCE is a private dataset we collected from a car insurance company's website. Each question is classified into 319 classes with corresponding answer data. All questions which belong to the same category share the same answers. The DMV dataset is collected from New York State the DMV's FAQ website. The YAHOO Ans dataset is only a subset of the original publicly available YAHOO Answers dataset Fleming *et al.* (2012); Shah and Pomerantz (2010). Though not very suitable for our framework, we still included the frequently used TREC dataset (factoid question type classification) for comparison.

We only compare our model's performance with CNNs for two following reasons: we consider our "group sparsity" as a modification to the general CNNs for grouped feature selection. This idea is orthogonal to any other CNN-based models and can be easily applied to them; in addition, as discussed in Sec. 3, we did not find any other model in comparison with solving question classification tasks with answer sets.

There is crucial difference between the INSURANCE and DMV datasets on one hand and the YAHOO set on the other. In INSURANCE and DMV, all questions in the same (sub)category share the same answers, whereas YAHOO provides individual answers to each question.

	TREC	INSUR.	DMV	YAHOO dataset sub	top	unseen
CNN†	93.6	51.2	60	20.8	53.9	47
+sparsity‡	93.2	51.4	62	20.2	54.2	46
$\mathbf{W}_R$	93.8	53.5	62	21.8	54.5	48
$\mathbf{W}_Q$	**94.2**	53.8	64	22.1	54.1	48
$\mathbf{W}_A$	-	**55.4**	**66**	**22.2**	**55.8**	**53**

Table 3.2: Experimental results. Baselines: †sequential CNNs ($\alpha = \beta = 0$ in Eq. 3.5), ‡CNNs with global sparsity ($\beta = 0$). $\mathbf{W}_R$: randomly initialized projection matrix. $\mathbf{W}_Q$: question-initialized projection matrix. $\mathbf{W}_A$: answer set-initialized projection matrix. There are three different classification settings for YAHOO: subcategory, top-level category, and top-level accuracies on unseen sub-labels.

For multi-label classification (INSURANCE and DMV), we replace the softmax layer in CNNs with a sigmoid layer which predicts each category independently while softmax is not.

All experimental results are summarized in Table 3.2. The improvements are substantial for INSURANCE and DMV, but not as significant for YAHOO and TREC. One reason for this is the questions in YAHOO/TREC are shorter, which makes the group information harder to encode. Another reason is that each question in YAHOO/TREC has a single label, and thus can not fully benefit from group sparse properties.

Besides the conventional classification tasks, we also test our proposed model on an unseen-label case. In these experiments, there are a few sub-category labels that are not included in the training data. However, we still hope that our model could still return the correct parent category for these unseen subcategories at test time. In the testing set of YAHOO dataset, we randomly add 100 questions whose subcategory labels are unseen in training set. The classification results of YAHOO-unseen in Table 3.2 are obtained by mapping the predicted subcategories back to top-level categories. The improvements are substantial due to the group information encoding.

Chapter 4 –Generating Structured Output from Non-Structured Input

This paper describes Oregon State University's submissions to the shared WMT'17 task "multimodal translation task I". In this task, all the sentence pairs are image captions in different languages. The key difference between this task and conventional machine translation is that we have corresponding images as additional information for each sentence pair. In this paper, we introduce a simple but effective system which takes an image shared between different languages, feeding it into the both encoding and decoding side. We report our system's performance for English-French and English-German with Flickr30K (in-domain) and MSCOCO (out-of-domain) datasets. Our system achieves the best performance in TER for English-German for MSCOCO dataset.

4.1 Problems in Multimodal

Natural language generation (NLG) is one of the most important tasks in natural language processing (NLP). It can be applied to a lot of interesting applications such like machine translation, image captioning, question answering. In recent years, Recurrent Neural Networks (RNNs) based approaches have shown promising performance in generating more fluent and meaningful sentences compared with conventional models such as rule-based model Mirkovic *et al.* (2011), corpus-based n-gram models Wen *et al.* (2015) and trainable generators Stent *et al.* (2004).

More recently, attention-based encoder-decoder models Bahdanau *et al.* (2014) have been proposed to provide the decoder more accurate alignments to generate more relevant words. The remarkable ability of attention mechanisms quickly update the state-of-the-art performance on variety of NLG tasks, such as machine translation Luong *et al.* (2015), image captioning Xu *et al.* (2015); Yang *et al.* (2016), and text summarization Rush *et al.* (2015); Nallapati *et al.* (2016).

However, for multimodal translation Elliott *et al.* (2015), where we translate a caption from one language into another given a corresponding image, we need to design a new model since the decoder needs to consider both language and images at the same time.

This paper describes our participation in the WMT 2017 multimodal task 1. Our model feeds the image information to both the encoder and decoder, to ground their hidden representation within the same context of image during training. In this way, during testing time, the decoder would generate more relevant words given the context of both source sentence and image.

4.2 Model Description

For the neural-based machine translation model, the encoder needs to map sequence of word embeddings from the source side into another representation of the entire sequence using recurrent networks. Then, in the second stage, decoder generates one word at a time with considering global (sentence representation) and local information (weighted context) from source side. For simplicity, our proposed model is based on the attention-based encoder-decoder framework in Luong *et al.* (2015), refereed as "Global attention".

On the other hand, for the early work of neural-basic caption generation models Vinyals *et al.* (2015), the convolutional neural networks (CNN) generate the image features which feed into the decoder directly for generating the description.

The first stage of the above two tasks both map the temporal and spatial information into a fixed dimensional vector which makes it feasible to utilize both information at the same time.

Fig. 4.1 shows the basic idea of our proposed model (OSU1). The red character **I** represents the image feature that is generated from CNN. In our case, we directly use the image features that are provided by WMT, and these features are generated by residual networks He *et al.* (2016).

The encoder (blue boxes) in Fig. 4.1 takes the image feature as initialization for generating each hidden representation. This process is very similar to neural-basic caption generation Vinyals *et al.* (2015) which grounds each word's hidden representation to the context given by the image. On the decoder side (green boxes in Fig. 4.1), we not only let each decoded word align to source words by global attention but also feed the image feature as initialization to the decoder.

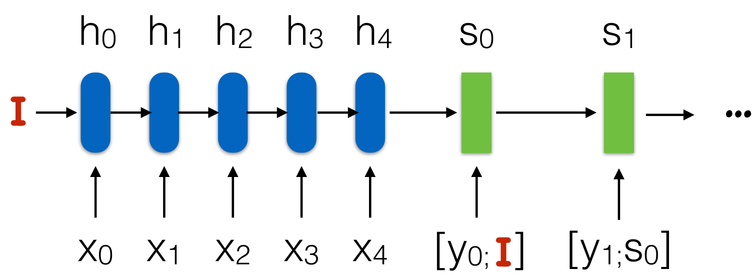


Figure 4.1: The image information is feed to both encoder and decoder for initialization. I (in red) represents the image feature that are generated by CNN.

4.3 Experiments

4.3.1 Datasets

In our experiments, we use two datasets Flickr30K Elliott *et al.* (2016) and MSCOCO Lin *et al.* (2014) which are provided by the WMT organization. For both datasets, there are triples that contains English as source sentence, its German and French human translations and corresponding image. The system is only trained on Flickr30K datasets but are also tested on MSCOCO besides Flickr30K. MSCOCO datasets are considered out-of-domain (OOD) testing while Flickr30K dataset are considered in-domain testing. The datasets' statics is shown in Table 4.1

Datasets	Train	Dev	Test	OOD ?
Flickr30K	29,000	1,014	1,000	No
MSCOCO	-	-	461	Yes

Table 4.1: Summary of datasets statistics.

4.3.2 Training details

For preprocessing, we convert all of the sentences to lower case, normalize the punctuation, and do the tokenization. For simplicity, our vocabulary keeps all the words that show in training set. For image representation, we use ResNet He *et al.* (2016) generated image features which are provided by the WMT organization. In our experiments, we only use average pooled features.

Our implementation is adapted from on Pytorch-based OpenNMT Klein *et al.* (2017). We use two layered bi-LSTM Sutskever *et al.* (2014b) on the source side as encoder. Our batch size is 64, with SGD optimization and a learning rate at 1. For English to German, the dropout rate is 0.6, and for English to French, the dropout rate is 0.4. These two parameters are selected by observing the performance on development set. Our word embeddings are randomly initialized with 500 dimensions. The source side vocabulary is 10,214 and the target side vocabulary is 18,726 for German and 11,222 for French.

4.3.3 Beam search with length reward

During test time, beam search is widely used to improve the output text quality by giving the decoder more options to generate the next possible word. However, different from traditional beam search in phrase-based MT where all hypotheses know the number of steps to finish the generation, while in neural-based generation, there is no information about what is the most ideal number of steps to finish the decoding. The above issue also leads to another problem that the beam search in neural-based MT prefers shorter sequences due to probability-based scores for evaluating different candidates. In this paper, we use Optimal Beam Search Huang *et al.* (2017) (OBS) during decoding time. OBS uses bounded length reward mechanism which allows a modified version of our beam search algorithm to remain optimal.

Figure 4.2 and Figure 4.3 show the BLEU score and length ratio with different rewards for different beam size. We choose beam size equals to 5 and reward equals to 0.1 during decoding.

4.3.4 Results

WMT organization provides three different evaluating metrics: BLEU Papineni *et al.* (2002), METEOR Lavie and Denkowski (2009) and TER Snover *et al.* (2006).

Table 4.2 to Table 4.5 summarize the performance with their corresponding rank among all other systems. We only show a few top performing systems in the tables to make a comparison. OSU1 is our proposed model and OSU2 is our baseline system without any image information. For MSCOCO dataset, the translation from English

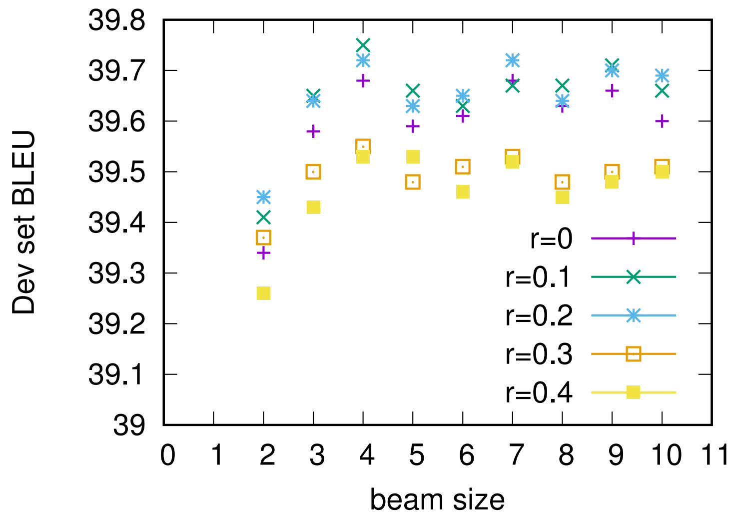


Figure 4.2: BLEU vs. beam size

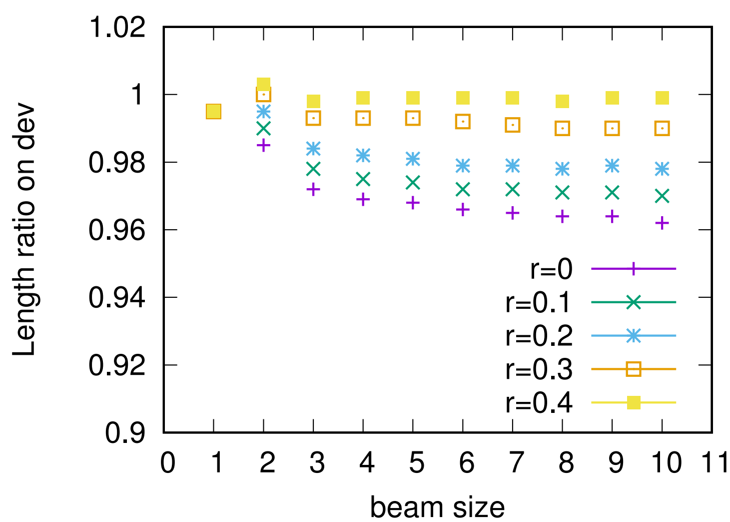


Figure 4.3: length ratio vs. beam size

to German (Table 4.3), which is the hardest tasks compared with others since it is from English to German on OOD dataset, we achieve best TER score across all other systems.

System	Rank	TER	METEOR	BLEU
UvA-TiCC	1	**47.5**	53.5	**33.3**
NICT	2	48.1	**53.9**	31.9
LIUMCVC	3 & 4	48.2	53.8	33.2
CUNI	5	50.7	51	31.1
OSU2†	6	50.7	50.6	31
OSU1†	8	51.6	48.9	29.7

Table 4.2: Experiments on Flickr30K dataset for translation from English to German. 16 systems in total. † represents our system.

System	Rank	TER	METEOR	BLEU
OSU1†	1	**52.3**	46.5	27.4
UvA-TiCC	2	52.4	48.1	28
LIUMCVC	3	52.5	**48.9**	**28.7**
OSU2†	8	55.9	45.7	26.1

Table 4.3: Experiments on MSCOCO dataset for translation from English to German. 15 systems in total. † represents our system.

System	Rank	TER	METEOR	BLEU
LIUMCVC	1	**28.4**	**72.1**	**55.9**
NICT	2	28.4	72	55.3
DCU	3	30	70.1	54.1
OSU2†	5	32.7	68.3	51.9
OSU1†	6	33.6	67.2	51

Table 4.4: Experiments on Flickr30K dataset for translation from English to French. 11 systems in total. † represents our system.

As describe in section 2.3.1, OSU1 is the model with image information for both encoder and decoder, and OSU2 is only the neural machine translation baseline without any image information. From the above results table we found that image information would hurt the performance in some cases. In order to have more detailed analysis, we show some test examples for the translation from English to German on MSCOCO

System	Rank	TER	METEOR	BLEU
LIUMCVC	1	**34.2**	**65.9**	**45.9**
NICT	2	34.7	65.6	45.1
DCU	3	35.2	64.1	44.5
OSU2†	4	36.7	63.8	44.1
OSU1†	6	37.8	61.6	41.2

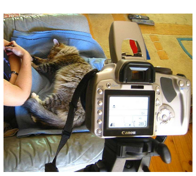
Table 4.5: Experiments on MSCOCO dataset for translation from English to French. 11 systems in total.

input	a finger pointing at a hotdog with cheese , sauerkraut and ketchup .
OSU1	ein finger zeigt auf einen hot dog mit einem messer , wischmobs und napa .
OSU2	ein finger zeigt auf einen hotdog mit hammer und italien .
Reference	ein finger zeigt auf einen hotdog mit kÃďse , sauerkraut und ketchup .

input	a man reaching down for something in a box
OSU1	ein mann greift nach unten , um etwas zu irgendeinem .
OSU2	ein mann greift nach etwas in einer kiste .
Reference	ein mann bÃijckt sich nach etwas in einer schachtel .

Figure 4.4: Two testing examples that image information confuses the NMT model.

input	there are two foods and one drink set on the clear table .
OSU1	da sind zwei speisen und ein getrÃďnk am klaren tisch .
OSU2	zwei erwachsene und ein erwachsener befinden sich auf dem rechteckigen tisch .
Reference	auf dem transparenten tisch stehen zwei speisen und ein getrÃďnk .



input	a camera set up in front of a sleeping cat .
OSU1	eine kameracrew vor einer schlafenden katze .
OSU2	eine kamera vor einer blonden katze .
Reference	eine kamera , die vor einer schlafenden katze aufgebaut ist

Figure 4.5: Two testing examples that image information helps the NMT model.

dataset.

Fig 4.4 shows two examples that NMT baseline model performances better than OSU1 model. In the first example, OSU1 generates several unseen objects from given image, such like knife. The image feature might not represent the image accurately. For the second example, OSU1 model ignores the object “box” in the image.

Fig 4.5 shows two examples that image feature helps the OSU1 to generate better results. In the first example, image feature successfully detects the object “drink” while the baseline completely neglects this. In the second example, the image feature even help the model figure out the action of the cat is “sleeping”.

Bibliography

Bahdanau, D., K. Cho, and Y. Bengio, 2014. Neural Machine Translation by Jointly Learning to Align and Translate. CoRR .

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle, 2007. Greedy Layer-Wise Training of Deep Networks, in: Advances in Neural Information Processing Systems 19, B. Schölkopf, J. Platt, and T. Hoffman, editors, MIT Press, pp. 153–160. URL `http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf`.

Boser, B. E., I. M. Guyon, and V. N. Vapnik, 1992. A Training Algorithm for Optimal Margin Classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory.

Candè, E. J. and M. B. Wakin, 2008. An Introduction To Compressive Sampling, in: Signal Processing Magazine, IEEE, volume 25. URL `http://dx.doi.org/10.1109/msp.2007.914731`.

Cheng, J., L. Dong, and M. Lapata, 2016a. Long Short-Term Memory-Networks for Machine Reading.

—, 2016b. Long Short-Term Memory-Networks for Machine Reading. arXiv preprint arXiv:1601.06733 .

Collins, M., 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms, in: Proceedings of EMNLP.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, 2011. Natural Language Processing (Almost) from Scratch, volume 12, pp. 2493–2537.

Dave, K., S. Lawrence, and D. M. Pennock, 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews, in: Proceedings of the 12th international conference on World Wide Web, ACM, pp. 519–528.

Elliott, D., S. Frank, and E. Hasler, 2015. Multi-Language Image Description with Neural Sequence Models. CoRR .

Elliott, D., S. Frank, K. Sima'an, and L. Specia, 2016. Multi30K: Multilingual English-German Image Descriptions. Proceedings of the 5th Workshop on Vision and Language :70–74.

Fleming, S., D. Chalmers, and I. Wakeman, 2012. A deniable and efficient question and answer service over ad hoc social networks, Springer Netherlands, volume 15, pp. 296–331. URL `http://dx.doi.org/10.1007/s10791-012-9185-0`.

Gamon, M., 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis, in: Proceedings of the 20th international conference on Computational Linguistics, Association for Computational Linguistics, p. 841.

Guo, D., G. Tur, W.-T. Yih, and G. Zweig, 2014. Joint semantic utterance classification and slot filling with recursive neural networks, in: IEEE Workshop on Spoken Language Technology (SLT).

He, K., X. Zhang, S. Ren, and J. Sun, 2016. Deep Residual Learning for Image Recognition. Conference on Computer Vision and Pattern Recognition CVPR .

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014. Improving neural networks by preventing co-adaptation of feature detectors. Journal of Machine Learning Research 15.

Hochreiter, S. and J. Schmidhuber, 1997. Long Short-Term Memory, Cambridge, MA, USA.

Huang, L., K. Zhao, and M. Ma, 2017. Optimal Beam Search for Neural Text Generation (modulo beam size), in: EMNLP 2017.

Irsoy, O. and C. Cardie, 2014. Deep Recursive Neural Networks for Compositionality in Language, in: Advances in Neural Information Processing Systems, pp. 2096–2104.

Jeong, M. and G. G. Lee, 2008. Triangular-Chain Conditional Random Fields. Audio, Speech, and Language Processing, IEEE Transactions on 16.

Judge, J., A. Cahill, and J. van Genabith, 2006. QuestionBank: Creating a Corpus of Parse-Annotated Questions, in: Proceedings of COLING. URL http://www.aclweb.org/anthology/P06-1063.

Kalchbrenner, N., E. Grefenstette, and P. Blunsom, 2014. A Convolutional Neural Network for Modelling Sentences, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Baltimore, Maryland, pp. 655–665. URL http://www.aclweb.org/anthology/P14-1062.

Kim, Y., 2014. Convolutional Neural Networks for Sentence Classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, pp. 1746–1751. URL http://www.aclweb.org/anthology/D14-1181.

Klein, G., Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. ArXiv e-prints .

Koo, T. and M. Collins, 2010. Efficient third-order dependency parsers, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, pp. 1–11.

Kudo, T. and Y. Matsumoto, 2004. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing. URL http://aclweb.org/anthology/W04-3239.

Lafferty, J. D., A. McCallum, and F. C. N. Pereira, 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, in: Proceedings of the Eighteenth International Conference on Machine Learning.

Lavie, A. and M. J. Denkowski, 2009. The Meteor metric for automatic evaluation of machine translation. Machine Translation .

Le, Q. V. and T. Mikolov, 2014. Distributed Representations of Sentences and Documents.

LeCun, Y., L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. MÃČÅŠller, E. SÃČâĆňckinger, P. Simard, and V. Vapnik, 1995. Comparison of Learning Algorithms for Handwritten Digit Recognition, in: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS, pp. 53–60.

Li, X. and D. Roth, 2002. Learning Question Classifiers, in: Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '02, pp. 1–7. URL http://dx.doi.org/10.3115/1072228.1072378.

Lin, T., M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, 2014. Microsoft COCO: Common Objects in Context .

Luong, M.-T., H. Pham, and C. D. Manning, 2015. Effective Approaches to Attention-based Neural Machine Translation. CoRR .

Ma, M., L. Huang, B. Xiang, and B. Zhou, 2015. Dependency-based Convolutional Neural Networks for Sentence Embedding, in: Proceedings of ACL 2015.

Makhzani, A. and B. Frey, 2014. K-Sparse Autoencoders, in: International Conference on Learning Representations.

Manning, C. D., M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, 2014. The Stanford CoreNLP Natural Language Processing Toolkit, in: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60. URL http://www.aclweb.org/anthology/P/P14/P14-5010.

Matsumoto, S., H. Takamura, and M. Okumura, 2005. Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees, in: Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining.

McDonald, R. and F. Pereira, 2006. Online Learning of Approximate Dependency Parsing Algorithms, in: Proceedings of EACL.

Mirkovic, D., L. Cavedon, M. Purver, F. Ratiu, T. Scheideck, F. Weng, Q. Zhang, and K. Xu, 2011. Dialogue management using scripts and combined confidence scores. US Patent :7,904,297.

Nallapati, R., B. Zhou, and M. Ma, 2016. Classify or Select: Neural Architectures for Extractive Document Summarization. CoRR .

Ng, A., 2011. Sparse Autoencoder. URL https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

Pang, B. and L. Lee, 2005. Seeing Stars: Exploiting Class Relationships For Sentiment Categorization With Respect To Rating Scales, in: Proceedings of ACL, pp. 115–124.

Papineni, K., S. Roukos, T. Ward, and W.-J. Zhu, 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting on Association for Computational Linguistics .

Rapaport, W. J., 2012. A History of the Sentence "Buffalo buffalo buffalo buffalo buffalo.". URL http://www.cse.buffalo.edu/~rapaport/buffalobuffalo.html.

Rubinstein, R., A. M. Bruckstein, and M. Elad, 2010. Dictionaries for Sparse Representation Modeling.

Rush, A. M., S. Chopra, and J. Weston, 2015. A Neural Attention Model for Abstractive Sentence Summarization .

Shah, C. and J. Pomerantz, 2010. Evaluating and Predicting Answer Quality in Community QA, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, SIGIR '10, pp. 411–418. URL http://doi.acm.org/10.1145/1835449.1835518.

Shen, Y., X. he, J. Gao, L. Deng, and G. Mesnil, 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search, WWW 2014. URL http://research.microsoft.com/apps/pubs/default.aspx?id=214617.

Silva, J., L. Coheur, A. C. Mendes, and A. Wichert, 2011. From symbolic to subsymbolic information in question classification. Artificial Intelligence Review 35.

Simon, N., J. Friedman, T. Hastie, and R. Tibshirani, 2013. A sparse-group lasso.

Snover, M., B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, 2006. A study of translation edit rate with targeted human annotation. In Proceedings of Association for Machine Translation in the Americas .

Socher, R., J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions, in: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Socher, R., A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Stroudsburg, PA, pp. 1631–1642.

Stent, A., R. Prasad, and M. Walker, 2004. Trainable Sentence Planning for Complex Information Presentation in Spoken Dialog Systems. Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics .

Sutskever, I., O. Vinyals, and Q. V. Le, 2014a. Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, pp. 3104–3112.

—, 2014b. Sequence to Sequence Learning with Neural Networks. Proceedings of the 27th International Conference on Neural Information Processing Systems .

Vinyals, O., A. Toshev, S. Bengio, and D. Erhan, 2015. Show and tell: A neural image caption generator. IEEE Conference on Computer Vision and Pattern Recognition :3156–3164.

Wen, T., M. Gasic, D. Kim, N. Mrksic, P. Su, D. Vandyke, and S. J. Young, 2015. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. CoRR .

Xu, K., J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of the 32nd International Conference on Machine Learning (ICML-15) .

Xu, P. and R. Sarikaya, 2013. Convolutional neural network based triangular CRF for joint intent detection and slot filling., in: ASRU, IEEE, pp. 78–83.

Yang, Z., Y. Yuan, Y. Wu, W. W. Cohen, and R. Salakhutdinov, 2016. Review Networks for Caption Generation. Advances in Neural Information Processing Systems .

Yih, W.-t., X. He, and C. Meek, 2014. Semantic Parsing for Single-Relation Question Answering, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, pp. 643–648. URL `http://aclweb.org/anthology/P14-2105`.

Yuan, M. and Y. Lin, 2006. Model selection and estimation in regression with grouped variables, volume 68, pp. 49–67.

Zeiler, M., 2012. Adadelta: An adaptive learning rate method, *Unpublished manuscript*: `http://arxiv.org/abs/1212.5701`.

Zhu, X., P. Sobhani, and H. Guo, 2015. Long Short-Term Memory over Tree Structures.