# Fine-grained Opinion Mining with Recurrent Neural Networks and Word Embeddings

**Pengfei Liu**[1], **Shafiq Joty**[2] **and Helen Meng**[1]
[1]Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong, Hong Kong SAR, China
[2]Qatar Computing Research Institute - HBKU, Doha, Qatar
{pfliu, hmmeng}@se.cuhk.edu.hk, sjoty@qf.org.qa

## Abstract

The tasks in fine-grained opinion mining can be regarded as either a token-level sequence labeling problem or as a semantic compositional task. We propose a general class of discriminative models based on recurrent neural networks (RNNs) and word embeddings that can be successfully applied to such tasks without any task-specific feature engineering effort. Our experimental results on the task of opinion target identification show that RNNs, without using any hand-crafted features, outperform feature-rich CRF-based models. Our framework is flexible, allows us to incorporate other linguistic features, and achieves results that rival the top performing systems in SemEval-2014.

## 1 Introduction

Fine-grained opinion mining involves identifying the opinion holder who expresses the opinion, detecting opinion expressions, measuring their intensity and sentiment, and identifying the target or aspect of the opinion (Wiebe et al., 2005). For example, in the sentence "John says, the hard disk is very noisy", John, the opinion holder, expresses a very negative (i.e., sentiment with intensity) opinion towards the target "hard disk" using the opinionated expression "very noisy". A number of NLP applications can benefit from fine-grained opinion mining including opinion summarization and opinion-oriented question answering.

The tasks in fine-grained opinion mining can be regarded as either a token-level sequence labeling problem or as a semantic compositional task at the sequence (e.g., phrase) level. For example, identifying opinion holders, opinion expressions and opinion targets can be formulated as a token-level sequence tagging problem, where the task is to label each word in a sentence using the conventional BIO tagging scheme. For example, Table 1 shows a sentence tagged with BIO scheme for opinion target (middle row) and for opinion expression (bottom row) identification tasks. On the other hand, characterizing intensity and sentiment of an opinionated expression can be regarded as a semantic compositional problem, where the task is to aggregate vector representations of tokens in a meaningful way and later use them for sentiment classification (Socher et al., 2013).

| The | **hard** | **disk** | is | *very* | *noisy* |
|---|---|---|---|---|---|
| O | B-TARG | I-TARG | O | O | O |
| O | O | O | O | B-EXPR | I-EXPR |

Table 1: An example sentence annotated with BIO labels for opinion target (`TARG` tags) and for opinion expression (`EXPR` tags) extraction.

Conditional random fields (CRFs) (Lafferty et al., 2001) have been quite successful for different fine-grained opinion mining tasks, e.g., opinion expression extraction (Yang and Cardie, 2012). The state-of-the-art model for opinion target extraction is also based on a CRF (Pontiki et al., 2014). However, the success of CRFs depends heavily on the use of an appropriate feature set and feature function expansion, which often requires a lot of engineering effort for each task in hand.

An alternative approach of deep learning automatically learns latent features as distributed vectors and have recently been shown to outperform CRFs on similar tasks. For example, Irsoy and Cardie (2014) apply deep recurrent neural networks (RNNs) to extract opinion expressions from sentences and show that RNNs outperform CRFs. Socher et al. (2013) propose recursive neural networks for a semantic compositional task to identify the sentiments of phrases and sentences hierarchically using the syntactic parse trees.

Meanwhile, recent advances in word embed-



*Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443,
Lisbon, Portugal, 17-21 September 2015. 

ding induction methods (Collobert and Weston, 2008; Mikolov et al., 2013b) have benefited researchers in two ways: (*i*) they have contributed to significant gains when used as extra word features in existing NLP systems (Turian et al., 2010; Lebret and Lebret, 2013), and (*ii*) they have enabled more effective training of RNNs by providing compact input representations of the words (Mesnil et al., 2013; Irsoy and Cardie, 2014).

Motivated by the recent success of deep learning, in this paper we propose a general class of models based on RNN architecture and word embeddings, that can be successfully applied to fine-grained opinion mining tasks without any task-specific feature engineering effort. We experiment with several important RNN architectures including Elman-RNN, Jordan-RNN, long short term memory (LSTM) and their variations. We acquire pre-trained word embeddings from several external sources to give better initialization to our RNN models. The RNN models then fine-tune the word vectors during training to learn task-specific embeddings. We also present an architecture to incorporate other linguistic features into RNNs.

Our results on the task of opinion target extraction show that word embeddings improve the performance of state-of-the-art CRF models, when included as additional features. They also improve RNNs when used as pre-trained word vectors and fine-tuning them on the task gives the best results. A comparison between models demonstrates that RNNs outperform CRFs, even when they use word embeddings as the only features. Incorporating simple linguistic features into RNNs improves the performance even further. Our best results with LSTM RNN outperform the top performing system on the Laptop dataset and achieve the second best on the Restaurant dataset in SemEval-2014. We make our source code available.[1]

In the remainder of this paper, after discussing related work in Section 2, we present our RNN models in Section 3. In Section 4, we briefly describe the pre-trained word embeddings. The experiments and analysis of results are presented in Section 5. Finally, we summarize our contributions with future directions in Section 6.

## 2 Related Work

A line of previous research in fine-grained opinion mining focused on detecting opinion (subjective) expressions, e.g., (Wilson et al., 2005; Breck et al., 2007). The common approach was to formulate the problem as a sequence tagging task and use a CRF model. Later approaches extended this to jointly identify opinion holders (Choi et al., 2005), and intensity and polarity (Choi and Cardie, 2010).

Extracting aspect terms or opinion targets have been actively investigated in the past. Typical approaches include association mining to find frequent item sets (i.e., co-occurring words) as candidate aspects (Hu and Liu, 2004), classification-based methods such as hidden Markov model (Jin et al., 2009) and CRF (Shariaty and Moghaddam, 2011; Yang and Cardie, 2012; Yang and Cardie, 2013), as well as topic modeling techniques using Latent Dirichlet Allocation (LDA) model and its variants (Titov and McDonald, 2008; Lin and He, 2009; Moghaddam and Ester, 2012).

Conventional RNNs (e.g., Elman type) and LSTM have been successfully applied to various sequence prediction tasks, such as language modeling (Mikolov et al., 2010; Sundermeyer et al., 2012), speech recognition (Graves and Jaitly, 2014; Sak et al., 2014) and spoken language understanding (Mesnil et al., 2013). For sentiment analysis, Socher et al. (2013) propose to use recursive neural networks to hierarchically compose semantic word vectors based on syntactic parse trees, and use the vectors to identify the sentiments of the phrases and sentences. Le and Zuidema (2015) extended recursive neural networks with LSTM to compute a parent vector in parse trees by combining information of both output and LSTM memory cells from its two children.

Most relevant to our work is the recent work of Irsoy and Cardie (2014), where they apply deep Elman-type RNN to extract opinion expressions and show that deep RNN outperforms CRF, semi-CRF and shallow RNN. They used word embeddings from Google without fine-tuning them.

Although inspired, our work differs from the work of Irsoy and Cardie (2014) in many ways. (*i*) We experiment with not only Elman-type, but also with a Jordan-type and with a more advanced LSTM RNN, and demonstrate the performance of various RNN models. (*ii*) We use not only Google embeddings as pre-trained word vectors, but also other embeddings including SENNA and Amazon, and show their performances. (*iii*) We also fine-tune the embeddings for our task, which is shown to be very crucial. (*iv*) We present an RNN ar-

[1] https://github.com/ppfliu/opinion-target

chitecture to include other linguistic features and show its effectiveness. (*v*) Finally, we present a comprehensive experiment exploring different embedding dimensions and hidden layer sizes for all the variations of the RNNs (i.e., including features and bi-directionality).

## 3 Recurrent Neural Models

The recurrent neural models in this section compute compositional vector representations for word sequences of arbitrary length. These high-level (i.e., hidden-layer) distributed representations are then used as features to classify each token in the sentence. We first describe the common properties shared among the RNNs below, followed by the descriptions of the specific RNNs.

Each word in the vocabulary $V$ is represented by a $D$ dimensional vector in the shared look-up table $L \in \mathbb{R}^{|V| \times D}$. Note that $L$ is considered as a model parameter to be learned. We can initialize $L$ randomly or by pre-trained word embedding vectors (see Section 4). Given an input sentence $\mathbf{s} = (s_1, \cdots, s_T)$, we first transform it into a feature sequence by mapping each word token $s_t \in \mathbf{s}$ to an index in $L$. The look-up layer then creates a context vector $\mathbf{x_t} \in \mathbb{R}^{mD}$ covering $m-1$ neighboring tokens for each $s_t$ by concatenating their respective vectors in $L$. For example, given the context size $m = 3$, the context vector $\mathbf{x_t}$ for the word *disk* in Figure 1 is formed by concatenating the embeddings of *hard*, *disk* and *is*. This window-based approach is intended to capture short-term dependencies between neighboring words in a sentence (Collobert et al., 2011).

The concatenated vector is then passed through non-linear recurrent hidden layers to learn high-level compositional representations, which are in turn fed to the output layer for classification using `softmax`. Formally, the probability of $k$-th label in the output for classification into $K$ classes:

$$P(y_t = k|\mathbf{s}, \theta) = \frac{exp\ (\mathbf{w}_k^T \mathbf{h_t})}{\sum_{k=1}^{K} exp\ (\mathbf{w}_k^T \mathbf{h_t})} \quad (1)$$

where, $\mathbf{h}_t = \phi(\mathbf{x}_t)$ defines the transformations of $\mathbf{x}_t$ through the hidden layers, and $\mathbf{w}_k$ are the weights from the last hidden layer to the output layer. We fit the models by minimizing the negative log likelihood (NLL) of the training data. The NLL for the sentence $\mathbf{s}$ can be written as:

$$J(\theta) = \sum_{t=1}^{T} \sum_{k=1}^{K} y_{tk}\ log\ P(y_t = k|\mathbf{s}, \theta) \quad (2)$$

where, $y_{tk} = I(y_t = k)$ is an indicator variable to encode the gold labels, i.e., $y_{tk} = 1$ if the gold label $y_t = k$, otherwise 0.[2] The loss function minimizes the cross-entropy between the predicted distribution and the target distribution (i.e., gold labels). The main difference between the models described below is how they compute $\mathbf{h}_t = \phi(\mathbf{x}_t)$.

### 3.1 Elman-type RNN (Elman, 1990)

In an Elman-type RNN (Fig. 1a), the output of the hidden layer $\mathbf{h}_t$ at time $t$ is computed from a non-linear transformation of the current input $\mathbf{x}_t$ and the previous hidden layer output $\mathbf{h}_{t-1}$. Formally,

$$\mathbf{h}_t = f(U\mathbf{h}_{t-1} + V\mathbf{x}_t + \mathbf{b}) \quad (3)$$

where $f$ is a nonlinear function (e.g., `sigmoid`) applied to the hidden units. $U$ and $V$ are weight matrices between two consecutive hidden layers, and between the input and the hidden layers, respectively, and $\mathbf{b}$ is the bias vector.

This RNN thus creates internal states by remembering previous hidden layer, which allows it to exhibit dynamic temporal behavior. We can interpret $\mathbf{h}_t$ as an intermediate representation summarizing the past, which is in turn used to make a final decision on the current input.

### 3.2 Jordan-type RNN (Jordan, 1997)

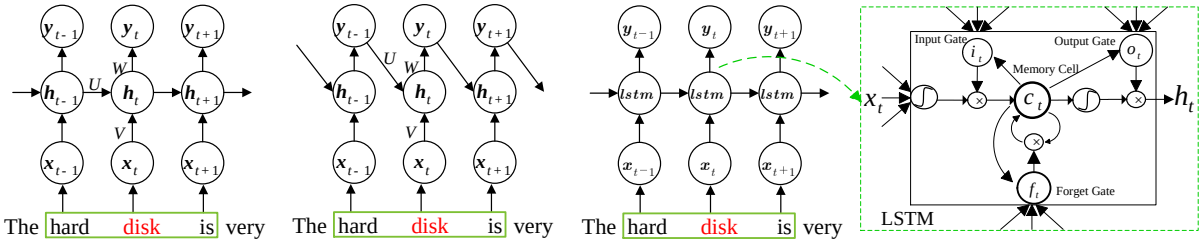
Jordan-type RNNs (Fig. 1b) are similar to Elman-type RNNs except that the hidden layer $\mathbf{h}_t$ at time $t$ is fed from the previous output layer $\mathbf{y}_{t-1}$ instead of the previous hidden layer $\mathbf{h}_{t-1}$. Formally,

$$\mathbf{h}_t = f(U\mathbf{y}_{t-1} + V\mathbf{x}_t + \mathbf{b}) \quad (4)$$

where $U$, $V$, $\mathbf{b}$, and $f$ are similarly defined as before. Both Elman-type and Jordan-type RNNs are known as simple RNNs. These types of RNNs are generally trained using stochastic gradient descent (SGD) with backpropagation through time (BPTT), where errors (i.e., gradients) are propagated back through the edges over time.

One common issue with BPTT is that as the errors get propagated, they may soon become very small or very large that can lead to undesired values in weight matrices, causing the training to fail.

[2] This is also known as one-hot vector representation.




(a) Elman-type RNN (b) Jordan-type RNN (c) Long Short-Term Memory (LSTM) RNN

Figure 1: Elman-type, Jordan-type and LSTM RNNs with a lookup-table layer, a hidden layer and an output layer. The concatenated context vector for the word "disk" at time $t$ is $x_t = [x_{hard}, x_{disk}, x_{is}]$ with a context window of size 3. One memory block in the LSTM hidden layer has been enlarged.

This is known as the *vanishing* and the *exploding* gradients problem (Bengio et al., 1994). One simple way to overcome this issue is to use a truncated BPTT (Mikolov, 2012) for restricting the back-propagation to only few steps like 4 or 5. However, this solution limits the RNN to capture long-range dependencies. In the following, we describe an elegant RNN architecture to address this problem.

### 3.3 Long Short-Term Memory RNN

Long Short-Term Memory or LSTM (Hochreiter and Schmidhuber, 1997) is specifically designed to model long term dependencies in RNNs. The recurrent layer in a standard LSTM is constituted with special (hidden) units called *memory blocks* (Fig. 1c). A memory block is composed of four elements: (*i*) a memory cell $c$ (i.e., a neuron) with a self-connection, (*ii*) an input gate $i$ to control the flow of input signal into the neuron, (*iii*) an output gate $o$ to control the effect of the neuron activation on other neurons, and (*iv*) a forget gate $f$ to allow the neuron to adaptively reset its current state through the self-connection. The following sequence of equations describe how a layer of memory blocks is updated at every time step $t$:

$$\mathbf{i}_t = \sigma(U_i\mathbf{h}_{t-1} + V_i\mathbf{x}_t + C_i\mathbf{c}_{t-1} + \mathbf{b}_i) \quad (5)$$
$$\mathbf{f}_t = \sigma(U_f\mathbf{h}_{t-1} + V_f\mathbf{x}_t + C_f\mathbf{c}_{t-1} + \mathbf{b}_f) \quad (6)$$
$$\mathbf{c}_t = \mathbf{i}_t \odot g(U_c\mathbf{h}_{t-1} + V_c\mathbf{x}_t + \mathbf{b}_c) + \mathbf{f}_t \odot \mathbf{c}_{t-1} \quad (7)$$
$$\mathbf{o}_t = \sigma(U_o\mathbf{h}_{t-1} + V_o\mathbf{x}_t + C_o\mathbf{c}_t + \mathbf{b}_o) \quad (8)$$
$$\mathbf{h}_t = \mathbf{o}_t \odot h(\mathbf{c}_t) \quad (9)$$

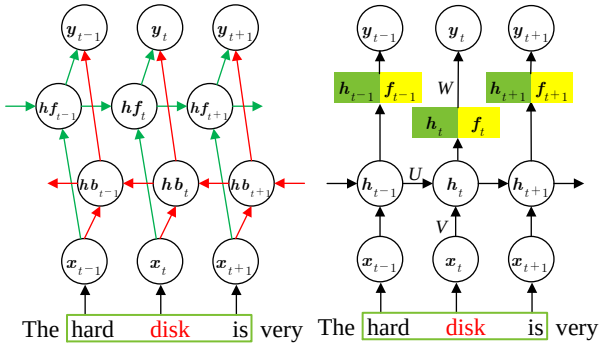
where $U_k$, $V_k$ and $C_k$ are the weight matrices between two consecutive hidden layers, between the input and the hidden layers, and between two consecutive cell activations, respectively, which are associated with gate $k$ (i.e., input, output, forget and cell), and $\mathbf{b}_k$ is the associated bias vector. The symbol $\odot$ denotes a element-wise product of the two vectors. The gate function $\sigma$ is the `sigmoid` activation, and $g$ and $h$ are the cell input and cell output activations, typically a `tanh`. LSTMs are generally trained using truncated or full BPTT.

### 3.4 Bidirectionality

Notice that the RNNs defined above only get information from the past. However, information from the future could also be crucial. In our example sentence (Table 1), to correctly tag the word *hard* as a **B-TARG**, it is beneficial for the RNN to know that the next word is *disk*. Our window-based approach, by considering the neighboring words, already captures short-term dependencies like this from the future. However, it requires tuning to find the right window size, and it disregards long-range dependencies that go beyond the context window, which is typically of size 1 (i.e., no context) to 5 (see Section 5.2). For instance, consider the two sentences: (*i*) *Do you know about the crunchy tuna here, it is to die for.* and (*ii*) *Do you know about the crunchy tuna here, it is imported from Norway.* The phrase *crunchy tuna* is an aspect term in the first (subjective) sentence, but not in the second (objective) one. The RNN models described above will assign the same labels to *crunchy* and *tuna* in both sentences, since the preceding sequences and the context window (of size 1 to 5) are the same.

To capture long-range dependencies from the future as well as from the past, we propose to use bidirectional RNNs (Schuster and Paliwal, 1997), which allow bidirectional links in the network. In an Elman-type bidirectional RNN (Fig. 2a), the forward hidden layer $\overrightarrow{\mathbf{h}_t}$ and the backward hidden layer $\overleftarrow{\mathbf{h}_t}$ at time $t$ are computed as follows:

$$\overrightarrow{\mathbf{h}_t} = f(\overrightarrow{U}\mathbf{h}_{t-1} + \overrightarrow{V}\mathbf{x}_t + \overrightarrow{\mathbf{b}}) \quad (10)$$
$$\overleftarrow{\mathbf{h}_t} = f(\overleftarrow{U}\mathbf{h}_{t-1} + \overleftarrow{V}\mathbf{x}_t + \overleftarrow{\mathbf{b}}) \quad (11)$$




(a) (b)

Figure 2: (a) Bidirectional Elman-type RNN and (b) Linguistic features concatenated with the hidden layer output in Elman-type RNN.

where $\overrightarrow{U}$, $\overrightarrow{V}$ and $\overrightarrow{\mathbf{b}}$ are the forward weight matrices as before; and $\overleftarrow{U}$, $\overleftarrow{V}$ and $\overleftarrow{\mathbf{b}}$ are their backward counterparts. The concatenated vector $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}]$ is passed to the output layer. We can thus interpret $\mathbf{h}_t$ as an intermediate representation summarizing the past and the future, which is then used to make a final decision on the current input.

Similarly, the unidirectional LSTM RNN can be extended to bidirectional LSTM by allowing bidirectional connections in the hidden layer. This amounts to having a backward counterpart for each of the equations from 5 to 9.

Notice that the forward and the backward computations of bidirectional RNNs are independently done until they are combined in the output layer. This means, during training, after backpropagating the errors from the output layer to the forward and to the backward hidden layers, two independent BPTT can be applied – one to each direction.

### 3.5 Fine-tuning of Embeddings

In our RNN framework, we intend to avoid manual feature engineering efforts by using word embeddings as the only features. As mentioned before, we can initialize the embeddings randomly and learn them as part of model parameters by backpropagating the errors to the look-up layer. One issue with random initialization is that it may lead the SGD to get stuck in local minima (Murphy, 2012). On the other hand, one can plug the readily available embeddings from external sources (Section 4) in the RNN model and use them as features without tuning them further for the task, as is done in any other machine learning model. However, this approach does not exploit the automatic feature learning capability of NN models, which is one of the main motivations of using them.

In our work, we use the pre-trained word embeddings to better initialize our models, and we fine-tune them for our task in training, which turns out to be quite beneficial (see Section 5.2).

### 3.6 Incorporating other Linguistic Features

Although NNs learn word features (i.e., embeddings) automatically, we may still be interested in incorporating other linguistic features like part-of-speech (POS) tags and chunk information to guide the training and to learn a better model. However, unlike word embeddings, we want these features to be fixed during training. As shown in Figure 2b, this can be done in our RNNs by feeding these additional features directly to the output layer, and learn their associated weights in training.

## 4 Word Embeddings

Word embeddings are distributed representations of words, represented as real-valued, dense, and low-dimensional vectors. Each dimension potentially describes syntactic or semantic properties of the word. Here we briefly describe the three types of pre-trained embeddings that we use in our work.

### 4.1 SENNA Embeddings

Collobert et al. (2011) present a unified NN architecture for various NLP tasks (e.g., POS tagging, chunking, semantic role labeling, named entity recognition) with a window-based approach and a sentence-based approach (i.e., the input layer is a sentence). Each word in the input layer is represented by $M$ features, each of which has an embedding vector associated with it in a *lookup table*. To give their network a better initialization, they learn word embeddings using a non-probabilistic language model, which was trained on English Wikipedia for about 2 months. They released their 50-dimensional word embeddings (vocabulary size $130K$) under the name SENNA.[3]

### 4.2 Google Embeddings

Mikolov et al. (2013a) propose two log-linear models for computing word embeddings from large corpora efficiently: (*i*) a *bag-of-words* model CBOW that predicts the current word based on the context words, and (*ii*) a *skip-gram* model that predicts surrounding words given the current word.

[3] http://ronan.collobert.com/senna/

They released their pre-trained 300-dimensional word embeddings (vocabulary size $3M$) trained by the skip-gram model on part of Google news dataset containing about 100 billion words.[4]

### 4.3 Amazon Embeddings

Since we work on customer reviews, which are less formal than Wikipedia and news, we have also trained *domain-specific* embeddings (vocabulary size $1M$) using the CBOW model of *word2vec* tool (Mikolov et al., 2013b) from a large corpus of Amazon reviews.[5] The corpus contains $34,686,770$ reviews ($4.7$ billion words) on Amazon products from June 1995 to March 2013 (McAuley and Leskovec, 2013). For comparison with SENNA and Google, we learn word embeddings of $50$- and $300$-dimensions.

## 5 Experiments

In this section, we present our experimental settings and results for the task of opinion target extraction from customer reviews.

### 5.1 Experimental Settings

**Datasets:** In our experiments, we use the two review datasets provided by the *SemEval-2014 task 4: aspect-based sentiment analysis* evaluation campaign (Pontiki et al., 2014), namely the *Laptop* and the *Restaurant* datasets. Table 2 shows some basic statistics about the datasets. The majority of aspect terms have only one word, while about one third of them have multiple words. In both datasets, some sentences have no aspect terms and some have more than one aspect terms. We use the standard train:test split to compare our results with the SemEval best systems. In addition, we show a more general performance of our models on the two datasets based on 10–fold cross validation.

| | Laptop | | Restaurant | |
|---|---|---|---|---|
| | Train | Test | Train | Test |
| Sentences | 3045 | 800 | 3041 | 800 |
| Sentence length | 15 | 13 | 14 | 14 |
| One-word targets | 1494 | 364 | 2786 | 818 |
| Multi-word targets | 864 | 290 | 907 | 316 |
| Total targets | 2358 | 654 | 3693 | 1134 |

Table 2: Corpora statistics.

[4]https://code.google.com/p/word2vec/

[5]https://snap.stanford.edu/data/web-Amazon.html

**Evaluation Metric:** The evaluation metric measures the standard precision, recall and $F_1$ score based on *exact* matches. This means that a candidate aspect term is considered to be correct only if it exactly matches with the aspect term annotated by the human. In all our experiments when comparing two models, we use paired $t$-test on the $F_1$ scores to measure statistical significance and report the corresponding $p$-value.

**CRF Baseline:** We use a linear-chain CRF (Lafferty et al., 2001) of order 2 as our baseline, which is the state-of-the-art model for opinion target extraction (Pontiki et al., 2014). Specifically, the CRF generates (binary) feature functions of order 1 and 2; see (Cuong et al., 2014) for higher order CRFs. The features used in the baseline model include the current word, its POS tag, its prefixes and suffixes between one to four characters, its position, its stylistics (e.g., case, digit, symbol, alphanumeric), and its context (i.e., the same features for the two preceding and the two following words). In addition to the hand-crafted features, we also include the three different types of word embeddings described in Section 4.

**RNN Settings:** We pre-processed each dataset by lowercasing all words and spelling out each digit number as `DIGIT`. We then built the vocabulary from the training set by marking rare words with only one occurrence as `UNKNOWN`, and adding a `PADDDING` word to make context windows for boundary words.

To implement *early stopping* in SGD, we prepared a *validation set* by separating out randomly $10\%$ of the available training data. The remaining $90\%$ is used for training. The weights in the network were initialized by sampling from a small random uniform distribution $\mathcal{U}(-0.2, 0.2)$. The time step in the truncated BPTT was fixed to 6 based on the performance on the validation set; smaller values hurt the performance, while larger values showed no significant gains but increased the training time.

We use a fixed learning rate of $0.01$, but we change the batch size depending on the sentence length following Mesnil et al. (2013). The net effect is a variable step size in the SGD. We run SGD for 30 epochs, calculate the $F_1$ score on the validation set after each epoch, and stop if the accuracy starts to decrease. The size of the context window and the hidden layer are empirically set based on the performance on the validation set. We experimented

with the window size $\in \{1, 3, 5\}$, and found 3 to be the optimal on the validation set. The hidden layer sizes we experimented with are 50, 100, 150, and 200; we report the optimal values in Table 3 (see $|h_l|$ and $|h_r|$ columns).

**Linguistic Features in RNNs:** In addition to the neural features, we also explore the contribution of simple linguistic features in our RNN models using the architecture described in Section 3.6. Specifically, we encode four POS tag classes (*noun, adjective, verb, adverb*) and BIO-tagged chunk information (*NP, VP, PP, ADJP, ADVP*) as binary features. We feed these extra features directly to the output layer of the RNNs and learn their relative weights. Part-of-speech and phrasal information are arguably the most informative features for identifying aspect terms (i.e., aspect terms are generally noun phrases). BIO tags could be useful to find the right text spans (i.e., aspect terms are unlikely to violate phrasal boundaries).

### 5.2 Results and Discussion

Table 3 presents our results of aspect term extraction on the *standard testset* in $F_1$ scores. In Table 4, we show the results on the *whole datasets* based on 10-fold cross validation. RNNs in Table 4 are trained using SENNA embeddings. We perform significance tests on the 10-fold results. In the following, we highlight our main findings.

**Contributions of Word Embeddings in CRF:** From the first group of results in Table 3, we can observe that even though CRF uses a handful of hand-designed features, including word embeddings still leads to sizable improvements on both datasets. The domain-specific Amazon embeddings (300 dim.) yield more general performance across the datasets, delivering the best gain of absolute $3.54\%$ on the Laptop and the second best on the Restaurant dataset. Google embeddings give the best gain on the Restaurant dataset (absolute $3.08\%$). The contribution of embeddings in CRF is also validated by the 10-fold results in Table 4 (see first two rows), where SENNA embeddings yield significant improvements – absolute $1.47\%$ on Laptop ($p < 0.03$) and absolute $1.24\%$ on Restaurant ($p < 0.01$). This demonstrates that word embeddings offer generalizations that complement other strong features, and thus should be considered.

**CRF vs. RNNs:** When we compare the results of

| System | Dim. | $\|h_l\|$ | Laptop | $\|h_r\|$ | Restaurant |
|---|---|---|---|---|---|
| **CRF Base** | - | - | 68.66 | - | 77.28 |
| +SENNA | 50 | - | 71.38 | - | 78.54 |
| +Amazon | 50 | - | 70.61 | - | 79.46 |
| +Google | 300 | - | 68.81 | - | **80.36** |
| +Amazon | 300 | - | **72.20** | - | 79.66 |
| **Jordan-RNN** | | | | | |
| +SENNA | 50 | 200 | 71.41 | 200 | 78.83 |
| +Amazon | 50 | 100 | 73.21 | 150 | 79.01 |
| +Google | 300 | 150 | **73.42** | 200 | **79.89** |
| +Amazon | 300 | 50 | 72.43 | 200 | 78.30 |
| **Elman-RNN** | | | | | |
| +SENNA | 50 | 100 | 73.86 | 150 | 79.89 |
| +Amazon | 50 | 100 | **74.43** | 100 | **80.37** |
| +Google | 300 | 100 | 72.91 | 100 | 79.54 |
| +Amazon | 300 | 200 | 73.67 | 100 | 79.82 |
| **Elman-RNN + Feat.** | | | | | |
| +SENNA | 50 | 50 | 73.70 | 100 | 81.36 |
| +Amazon | 50 | 200 | 73.30 | 50 | **81.66** |
| +Google | 300 | 150 | **74.25** | 100 | 80.57 |
| +Amazon | 300 | 50 | 73.92 | 100 | 80.24 |
| **Bi-Elman-RNN** | | | | | |
| +SENNA | 50 | 100 | 72.38 | 100 | **80.10** |
| +Amazon | 50 | 50 | **73.93** | 50 | 79.97 |
| +Google | 300 | 50 | 72.67 | 100 | 79.52 |
| +Amazon | 300 | 50 | 71.12 | 50 | 79.09 |
| **Bi-Elman-RNN + Feat.** | | | | | |
| +SENNA | 50 | 100 | 73.30 | 50 | 80.34 |
| +Amazon | 50 | 50 | **74.57** | 50 | **82.06** |
| +Google | 300 | 50 | 74.56 | 100 | 78.99 |
| +Amazon | 300 | 50 | 73.56 | 100 | 79.97 |
| **LSTM-RNN** | | | | | |
| +SENNA | 50 | 100 | 73.40 | 150 | 79.43 |
| +Amazon | 50 | 50 | 72.44 | 50 | **79.79** |
| +Google | 300 | 100 | 72.11 | 50 | 79.20 |
| +Amazon | 300 | 50 | **73.52** | 50 | 78.99 |
| **LSTM-RNN + Feat.** | | | | | |
| +SENNA | 50 | 50 | 73.19 | 150 | 80.28 |
| +Amazon | 50 | 100 | **75.00** | 50 | 80.82 |
| +Google | 300 | 50 | 72.19 | 50 | **81.37** |
| +Amazon | 300 | 100 | 72.85 | 100 | 80.60 |
| **Bi-LSTM-RNN** | | | | | |
| +SENNA | 50 | 50 | 72.60 | 150 | **79.89** |
| +Amazon | 50 | 100 | **74.03** | 100 | 79.36 |
| +Google | 300 | 50 | 70.90 | 50 | 78.80 |
| +Amazon | 300 | 150 | 71.25 | 150 | 78.88 |
| **Bi-LSTM-RNN + Feat.** | | | | | |
| +SENNA | 50 | 100 | **74.02** | 150 | **81.06** |
| +Amazon | 50 | 100 | 73.58 | 50 | 80.51 |
| +Google | 300 | 100 | 71.05 | 50 | 79.39 |
| +Amazon | 300 | 100 | 73.81 | 150 | 80.67 |
| **SemEval-14 top systems** | | | | | |
| IHS_RD | - | - | **74.55** | - | 79.62 |
| DLIREC | - | - | 73.78 | - | **84.01** |

Table 3: $F_1$-score performance for CRF baselines, RNNs and SemEval'14 best systems on the standard Laptop and Restaurant testsets. $|h_l|$ and $|h_r|$ columns show the number of hidden units.

| Model | Laptop | | | Restaurant | | |
|---|---|---|---|---|---|---|
| | P | R | $F_1$ | P | R | $F_1$ |
| CRF Base | **79.77** | 64.09 | 70.87 | **82.59** | 74.63 | 78.36 |
| + SENNA | 78.23 | **67.38** | **72.34** | 81.21 | **78.12** | **79.60** |
| Elman-RNN | **82.03** | 72.68 | 76.97 | 81.96 | 78.41 | 80.08 |
| + Feat. | 80.02 | **76.60** | **78.22** | 81.91 | **81.22** | 81.52 |
| + Bidir. | 81.92 | 73.70 | 77.47 | 81.69 | 78.46 | 79.97 |
| + Feat. + Bidir. | 81.00 | 75.70 | 78.17 | **82.80** | 80.44 | **81.57** |
| LSTM-RNN | **81.92** | 73.30 | 77.14 | **83.64** | 77.45 | 80.36 |
| + Feat. | 80.70 | **75.82** | **78.00** | 81.80 | **81.39** | 81.54 |
| + Bidir. | 81.31 | 74.20 | 77.37 | 81.66 | 79.23 | 80.37 |
| + Feat. + Bidir. | 80.81 | 74.48 | 77.27 | 82.96 | 80.42 | **81.56** |

Table 4: 10-fold cross validation results of the models on the two datasets. Elman- and LSTM-RNNs are trained using SENNA embeddings.

| System | Dim. | Laptop | | Restaurant | |
|---|---|---|---|---|---|
| **Elman-RNN** | | -tune | +tune | -tune | +tune |
| +SENNA | 50 | 60.85 | 73.86 | 75.78 | 79.89 |
| +Amazon | 50 | 15.51 | 74.43 | 22.85 | 80.37 |
| +Random | 50 | 38.26 | 72.99 | 56.98 | 78.44 |
| +Google | 300 | 67.91 | 72.91 | 74.73 | 79.54 |
| +Amazon | 300 | 15.51 | 73.67 | 22.85 | 79.82 |
| **Jordan-RNN** | | -tune | +tune | -tune | +tune |
| +SENNA | 50 | 58.81 | 71.41 | 74.68 | 78.83 |
| +Amazon | 50 | 15.51 | 73.21 | 22.85 | 79.01 |
| +Random | 50 | 38.05 | 71.46 | 55.65 | 77.38 |
| +Google | 300 | 69.39 | 73.42 | 77.33 | 79.89 |
| +Amazon | 300 | 15.51 | 72.43 | 22.85 | 78.30 |

Table 5: Effects of fine-tuning in Elman-RNN and Jordan-RNN.

RNNs with those of CRF in Table 3, we see that most of our RNN models outperform CRF models with the maximum absolute gains of $2.80\%$ by **LSTM-RNN+Feat.** on Laptop and $1.70\%$ by **Bi-Elman-RNN+Feat.** on Restaurant. What is remarkable is that RNNs without any hand-crafted features outperform feature-rich CRF models by a good margin – absolute maximum gains of $2.23\%$ by Elman-RNN and $1.83\%$ by **Bi-LSTM-RNN** on Laptop. When we compare their general performance on the 10-folds in Table 4, we observe similar gains, maximum $5.88\%$ on Laptop and $1.97\%$ on Restaurant, which are significant with $p < 6 \times 10^{-6}$ on Laptop and $p < 2 \times 10^{-4}$ on Restaurant. These results demonstrate that RNNs as sequence labelers are more effective than CRFs for fine-grained opinion mining tasks. This can be attributed to RNN's ability to learn better features automatically and to capture long-range sequential dependencies between the output labels.

**Comparison among RNN Models:** A comparison among the RNN models in Table 3 tells that Elman RNN generally outperforms Jordan RNN. However, bi-directionality and LSTM do not provide clear gains over the simple Elman RNN. In fact, bi-directionality hurts the performance in most cases. This finding contrasts the finding of Irsoy and Cardie (2014) in opinion expression detection task, where bi-directional Elman RNNs outperform their uni-directional counterparts. However, when we analyzed the data, we found it to be unsurprising because aspect terms are generally shorter than opinion expressions. For example, the average length of an aspect term in our Restaurant dataset is 1.4, where the average length of an expressive subjective expression in their MPQA corpus is 3.3. Therefore, the information required to correctly identify aspect terms (e.g., *hard disk*) is already captured by the simple (as opposed to LSTM) unidirectional link and the context window covering the neighboring words. LSTM and Bi-directionality increase the number of parameters in the RNNs, which might contribute to overfitting on this specific task.[6]

[6]Bi-directional links double the number of parameters in RNNs.

As a partial solution to this problem, we experimented with a bi-directional Elman-RNN, where both directions share the same parameters. Therefore, the number of parameters remains the same as the uni-directional one. This modification improves the performance over the non-shared one slightly but not significantly. This demands for better modeling of the two sources of information rather than simple concatenation or sharing.

**Contributions of Linguistic Features in RNNs:** Although our linguistic features are quite simple (i.e., POS tags and chunk), they give gains on both datasets when incorporated into Elman and LSTM RNNs. The maximum gains on the standard testset (Table 3) are $0.64\%$ on Laptop and $1.96\%$ on Restaurant for Bi-Elman, and $1.48\%$ on Laptop and $1.58\%$ on Restaurant for LSTM. Similar gains are also observed on the 10-folds in Table 4, where the maximum gains are $1.25\%$ on Laptop and $1.44\%$ on Restaurant. These gains are significant with $p < 0.004$ on Laptop and $p < 6 \times 10^{-5}$ on Restaurant. Linguistic features thus complement word embeddings in RNNs.

**Importance of Fine-tuning in RNNs:** Finally, in order to show the importance of fine-tuning the word embeddings in RNNs on our task, we present in Table 5 the performance of Elman and Jordan RNNs, when the embeddings are used as they are (*'-tune'*), and when they are fine-tuned (*'+tune'*) on the task. The table also shows the contributions of pre-trained embeddings as compared to random initialization. Surprisingly, Amazon embeddings without fine-tuning deliver the worst performance, even lower than the Random initialization. We found that with Amazon embeddings the network gets stuck in a local minimum from the very first epoch.

Other pre-trained (untuned) embeddings improve over the Amazon and Random by providing better initialization. In most cases fine-tuning makes a big difference. For example, the absolute gains for fine-tuning SENNA embeddings in Elman RNN are $13.01\%$ in Laptop and $4.11\%$ in Restaurant. Remarkably, fine-tuning brings both Random and Amazon embeddings close to the best ones.

**Comparison with SemEval-2014 Systems:** When our RNN results are compared with the top performing systems in the SemEval-2014 (last two rows in Table 3), we see that RNNs without using any linguistic features achieve the second best results on both Laptop and Restaurant datasets. Note that these RNNs only use word embeddings, while IHS_RD and DLIREC use complex features like dependency relations, named entity, sentiment orientation of words, word cluster and many more in their CRF models, most of which are expensive to compute; see (Toh and Wang, 2014; Chernyshevich, 2014). The performance of our RNNs improves when they are given access to very simple features like POS tags and chunks, and **LSTM-RNN+Feat.** achieves the best results on the Laptop dataset.

## 6 Conclusion and Future Direction

We presented a general class of discriminative models based on recurrent neural network (RNN) architecture and word embeddings, that can be successfully applied to fine-grained opinion mining tasks without any task-specific manual feature engineering effort. We used pre-trained word embeddings from three external sources in different RNN architectures including Elman-type, Jordan-type, LSTM and their several variations.

Our results on the opinion target extraction task demonstrate that word embeddings improve the performance of both CRF and RNN models, however, fine-tuning them in RNNs on the task gives the best results. RNNs outperform CRFs, even when they use word embeddings as the only features. Incorporating simple linguistic features into RNNs improves the performance further. Our best results with LSTM RNN outperform the top performing system on the Laptop dataset and achieve the second best on the Restaurant dataset in SemEval-2014 evaluation campaign. We made our code publicly available for research purposes.

In the future, we would like apply our models to other fine-grained opinion mining tasks including opinion expression detection and characterizing the intensity and sentiment of the opinion expressions. We would also like to explore to what extent these tasks can be jointly modeled in an RNN-based multi-task learning framework.

## Acknowledgments

We are grateful to the anonymous reviewers for their insightful comments and suggestions to improve the paper. This research is affiliated with the Stanley Ho Big Data Decision Analytics Research Centre of The Chinese University of Hong Kong.

## References


Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2683–2688. Morgan Kaufmann Publishers Inc.

Maryna Chernyshevich. 2014. IHS R&D Belarus: Cross-domain extraction of product features using conditional random fields. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, page 309.

Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 269–274. ACL.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of HLT/EMNLP*, pages 355–362. ACL.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. 2014. Conditional random field with high-order dependencies for sequence labeling and segmentation. *The Journal of Machine Learning Research*, 15(1):981–1009.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of ICML*, pages 1764–1772.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of SIGKDD*, pages 168–177. ACM.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of EMNLP*, pages 720–728.

Wei Jin, Hung Hay Ho, and Rohini K Srihari. 2009. A novel lexicalized HMM-based learning framework for web opinion mining. In *Proceedings of ICML*, pages 465–472. Citeseer.

Michael I Jordan. 1997. Serial order: A parallel distributed processing approach. *Advances in psychology*, 121:471–495.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*, pages 282–289.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the joint Conference on Lexical and Computational Semantics (*SEM)*.

Rémi Lebret and Ronan Lebret. 2013. Word emdeddings through Hellinger PCA. *arXiv preprint arXiv:1312.5542*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of CIKM*, pages 375–384. ACM.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of INTERSPEECH*, pages 3771–3775.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*, pages 1045–1048.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, 2012. *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology.

Samaneh Moghaddam and Martin Ester. 2012. On the design of LDA models for aspect-based opinion mining. In *Proceedings of CIKM*, pages 803–812. ACM.

Kevin Murphy. 2012. *Machine Learning A Probabilistic Perspective*. The MIT Press.

Maria Pontiki, Haris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Hasim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of INTERSPEECH*, pages 338–342.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Shabnam Shariaty and Samaneh Moghaddam. 2011. Fine-grained opinion mining using conditional random fields. In *International Conference on Data Mining Workshops (ICDMW)*, pages 109–114. IEEE.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642. Citeseer.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Proceedings of INTERSPEECH*, pages 194–197.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW*, pages 111–120. ACM.

Zhiqiang Toh and Wenting Wang. 2014. DLIREC: Aspect term extraction and term polarity classification system. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, page 235.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of ACL*, pages 384–394. ACL.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP*, pages 347–354. ACL.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-Markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345. ACL.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1640–1649. ACL.