Deep Convolutional Neural Network for Sentiment Analysis of Short Texts

Abstract

Sentiment analysis of short texts such as single sentences and Twitter messages is challenging because of the limited contextual information that they normally contain. Effectively solving this task requires strategies that combine the small text content with prior knowledge and use more than just bag-of-words. In this work we propose a new deep convolutional neural network that exploits from character- to sentence-level information to perform sentiment analysis of short texts. We apply our approach for two corpora of two different domains: the Stanford Sentiment Treebank (SSTb), which contains sentences from movie reviews; and the Stanford Twitter Sentiment corpus (STS), which contains Twitter messages. For the SSTb corpus, our approach achieves state-of-the-art results for single sentence sentiment prediction in both binary positive/negative classification

Introduction

The advent of online social networks has produced a crescent interest on the task of sentiment analysis for short text messages . However, sentiment analysis of short texts such as single sentences and and microblogging posts, like Twitter messages, is challenging because of the limited amount of contextual data in this type of text. Effectively solving this task requires strategies that go beyond bag-of-words and extract information from the sentence/message in a more disciplined way. Additionally, to fill the gap of contextual information in a scalable manner, it is more suitable to use methods that can exploit prior knowledge from large sets of unlabeled texts. In this work we propose a deep convolutional neural network that exploits from character- to sentence level information to perform sentiment analysis of short texts. The proposed network, named Character to Sentence Convolutional Neural Network (CharSCNN), uses two convolutional layers to extract relevant features from words and sentences of any size. The proposed network can easily explore the richness of word embeddings produced by unsupervised pre-training . We perform experiments that show the effectiveness of CharSCNN for sentiment analysis of texts from two domains: movie review sentences; and Twitter messages (tweets). CharSCNN achieves state-of-the-art results for the two domains. Additionally, in our experiments we provide information about the usefulness of unsupervised pre-training; the contribution of character-level features; and the effectiveness of sentence-level features to detect negation.

Neural Network Architecture

the network takes as input the sequence of words in the sentence, and passes it through a sequence of layers where features with increasing levels of complexity are extracted. The network extracts features from the character-level up to the sentence-level. The main novelty in our network architecture is the inclusion of two convolutional layers, which allows it to handle words and sentences of any size.

Initial Representation Levels

The first layer of the network transforms words into real-valued feature vectors (embeddings) that capture morphological, syntactic and semantic information about the words. We use a fixed-sized word vocabulary (Vwrd), and we consider that words are composed of characters from a fixed-sized character vocabulary (Vchr)

Word-Level Embeddings

Word-level embeddings are encoded by column vectors in an embedding matrix. which has value 1 at index w and zero in all other positions. The matrix (Wwrd) is a parameter to be learned, and the size of the word-level embedding

Character-Level Embeddings

Robust methods to extract morphological and shape information from words must take into consideration
all characters of the word and select which features are more important for the task at hand .
the convolutional approach produces local features around each character of the word and then combines them using a max operation to create a fixed-sized character-level embedding of the

Sentence-Level Representation and Scoring

Methods to extract a sentence-wide feature set most deal with two main problems: sentences have different sizes; and important information can appear at any position in the sentence. We tackle these problems by using a convolutional layer to compute the sentence-wide feature vector r sent. This second convolutional layer in our neural network architecture works in a very similar way to the one used to extract character-level features for words. This layer produces local features around each word in the sentence and then combines them using a max operation to create a fixed-sized feature vector for the sentence

Experimental Setup and Results

Sentiment Analysis Datasets

We apply CharSCNN for two different corpora from two different domains: movie reviews and Twitter posts. The movie review dataset used is the recently proposed Stanford Sentiment Treebank (SSTb) , which includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences. In our experiments we focus in sentiment prediction of complete sentences. However, we show the impact of training with sentences and phrases instead of only sentences. The second labeled corpus we use is the Stanford Twitter Sentiment corpus (STS) .

Dataset	Set	# sentences / tweets	# classes
SSTb	Train	8544	5
	Dev	1101	5
	Test	2210	5
STS	Train	80K	2
	Dev	16K	2
	Test	498	3

Unsupervised Learning of Word-Level Embeddings

Word-level embeddings play a very important role in the (CharSCNN) architecture.
They are meant to capture syntactic and semantic information, which are very important to sentiment analysis. Recent work has shown that large improvements in terms of model accuracy can be obtained by performing unsupervised pre-training of word embeddings
we perform unsupervised learning of word-level embeddings using the word2vec
which implements the continuous bag-of-words and skip-gram architectures for computing vector representations of word

Results for SSTb Corpus

we present the result of CharSCNN and SCNN for different versions of the SSTb corpus. Note that SSTb corpus is a sentiment treebank, hence it contains sentiment annotations for all phrases in all sentences in the corpus. In our experiments, we check whether using examples that are single phrases, in addition to complete sentences, can provide useful information for training the proposed NN. However, in our experiments the test set always includes only complete sentences. In Table 3, the column Phrases indicates whether all phrases (yes) or only complete sentences (no) in the corpus are used for training. The Fine-Grained column contains prediction results for the case where 5 sentiment classes (labels) are used (very negative, negative, neutral, positive, very positive). The Positive/Negative column presents prediction results for the case of binary classification of sentences, i.e, the neutral class is removed, the two negative classes are merged as well as the two positive classes.

Model	Phrases	Fine-Grained	Positive/Negative
CharSCNN	yes	**48.3**	**85.7**
SCNN	yes	**48.3**	85.5
CharSCNN	no	43.5	82.3
SCNN	no	43.5	82.0
RNTN (Socher et al., 2013b)	yes	45.7	85.4
MV-RNN (Socher et al., 2013b)	yes	44.4	82.9
RNN (Socher et al., 2013b)	yes	43.2	82.4
NB (Socher et al., 2013b)	yes	41.0	81.8
SVM (Socher et al., 2013b)	yes	40.7	79.4

Results for STS Corpus

we present the results of CharSCNN and SCNN for sentiment prediction using the STS corpus. As expected, character-level information has a greater impact for Twitter data. Using unsupervised pre-training, CharSCNN provides an absolute accuracy improvement of 1.2 over SCNN. Additionally, initializing word-embeddings using unsupervised pre-training gives an absolute accuracy increase of around 4.5 when compared to randomly initializing the word-embeddings.

Model	Accuracy (unsup. pre-training)	Accuracy (random word embeddings)
CharSCNN	**86.4**	81.9
SCNN	85.2	82.2
LProp (Speriosu et al., 2011)	84.7	
MaxEnt (Go et al., 2009)	83.0	
NB (Go et al., 2009)	82.7	
SVM (Go et al., 2009)	82.2	

Recommendation System

A **recommender system** or a **recommendation system** (sometimes replacing "system" with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item.

Recommender systems have become increasingly popular in recent years, and are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts, collaborators, jokes, restaurants, financial services, life insurance, romantic partners (online dating), and Twitter pages.

➔ **Recommendation System Approaches**

Recommender systems typically produce a list of recommendations in one of three ways

1- Collaborative filtering
2- Content-based filtering
3- Hybrid recommender systems

Collaborative filtering

One approach to the design of recommender systems that has wide use is collaborative filtering. Collaborative filtering methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users.

→A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself.

→Many algorithms have been used in measuring user similarity or item similarity in recommender systems. For example, the k-nearest neighbor (k-NN) approach and the Pearson Correlation as first implemented by Allen.

➔ There are two senses of collaborative filtering

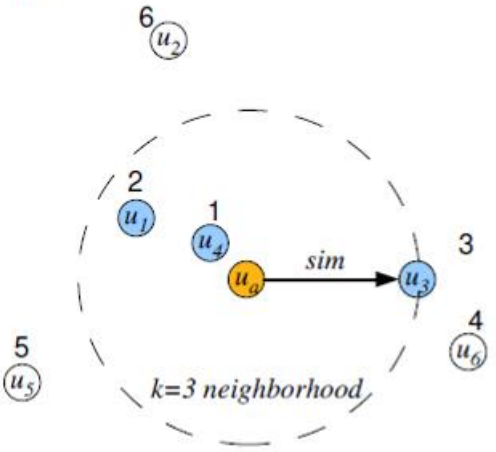
1) User-based Collaborative Filtering Recommendation
2) Item-Based Collaborative Filtering Recommendation

User-based Collaborative Filtering (CF)

Produce recommendations based on preferences of similar users (Goldberg, Nichols, Oki, and Terry, 1992; Resnick, Iacovou, Suchak, Bergstrom, and Riedl, 1994; Mild and Reutterer, 2001).

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_a	0	0	1	1	0	1	0	1
u_1	0	1	1	1	1	0	0	0
u_2	1	0	0	0	1	1	0	0
u_3	0	0	0	1	0	1	0	1
u_4	0	0	0	1	1	1	0	1
u_5	1	1	0	0	0	0	0	1
u_6	0	1	0	0	1	1	0	1
	0	1	1	1	2	2	0	2

Recommendation: i_5, i_2



1. Find k nearest neighbors based on similarity between users.
2. Generate recommendation based on the items liked by the k nearest neighbors. E.g., recommend most popular items or use a weighing scheme.

Item-based CF

Produce recommendations based on item similarities (Kitts, Freed, and Vrieze, 2000; Sarwar, Karypis, Konstan, and Riedl, 2001)

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
i_1	1	0.1	0	**0.3**	**0.2**	**0.4**	0	0.1
i_2	0.1	1	**0.8**	**0.9**	0	**0.2**	0.1	0
i_3	0	**0.8**	1	0	**0.4**	0.1	0.3	**0.5**
i_4	**0.3**	**0.9**	0	1	0	**0.3**	0	0.1
i_5	**0.2**	0	**0.4**	0	1	**0.1**	0	0
i_6	**0.4**	**0.2**	0.1	**0.3**	0.1	1	0	0.1
i_7	0	**0.1**	**0.3**	0	0	0	1	0
i_8	**0.1**	0	**0.5**	**0.1**	0	0.1	0	1
	0.3	0	0.9	0.4	0.2	0.5	0	0

$k=3$

$u_a=\{i_1, i_5, i_8\}$

Recommendation: i_3, i_6, i_4

1. Calculate similarities between items and keep for each item only the values for the k most similar items.
2. For each item add the similarities with the active user's items.
3. Remove the items of the active user and recommend the N items with the highest score.

→problems with collaborative filtering

- Cold start: These systems often require a large amount of existing data on a user in order to make accurate recommendations.
- Scalability: In many of the environments in which these systems make recommendations, there are millions of users and products. Thus, a large amount of computation power is often necessary to calculate recommendations.
- Sparsity: The number of items sold on major e-commerce sites is extremely large. The most active users will only have rated a small subset of the overall database. Thus, even the most popular items have very few ratings.

Content-based filtering

Content-based filtering methods are based on a description of the item and a profile of the user's preference. In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. The system generates recommendations from two sources: the features associated with products and the ratings that a user has given them. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on product features.

→the problems with Content-based filtering

A key issue with content-based filtering is whether the system is able to learn user preferences from users' actions regarding one content source and use them across other content types. When the system is limited to recommending content of the same type as the user is already using, the value from the recommendation system is significantly less than when other content types from other services can be recommended

Hybrid recommender systems

Recent research has demonstrated that a hybrid approach, combining collaborative filtering and content-based filtering could be more effective in some cases. Hybrid approaches can be implemented in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and vice versa); or by unifying the approaches into one model (a complete review of recommender systems). Several studies empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrate that the hybrid methods can provide more accurate recommendations than pure approaches. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem

Risk-aware recommender systems

The majority of existing approaches to recommender systems focus on recommending the most relevant content to users using contextual information and do not take into account the risk of disturbing the user in specific situation. However, in many applications, such as recommending personalized content, it is also important to consider the risk of upsetting the user so as not to push recommendations in certain circumstances, for instance, during a professional meeting, early morning, or late at night. Therefore, the performance of the recommender system depends in part on the degree to which it has incorporated the risk into the recommendation process.

Reference

C´ıcero Nogueira dos Santos Brazilian Research Lab
Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In Proceedings of the Conference on Empirical Methods in NLP, pages 647–657.

Pandora Lead Rise of Recommendation Engines - TIME". TIME.com. 27 May 2016. Retrieved 1 June 2015.
Jump up ^ H. Chen, A. G. Ororbia II, C. L. Giles ExpertSeer: a Keyphrase Based Expert Recommender for Digital Libraries, in arXiv preprint 2015