International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-017-0088-4

REGULAR PAPER

Deep learning for detecting inappropriate content in text

Harish Yenala[1] · Ashish Jhanwar[1] · Manoj K. Chinnakotla[1] · Jay Goyal[1]

Received: 1 September 2017 / Accepted: 11 December 2017

Abstract
Today, there are a large number of online discussion fora on the internet which are meant for users to express, discuss and exchange their views and opinions on various topics. For example, news portals, blogs, social media channels such as youtube. typically allow users to express their views through comments. In such fora, it has been often observed that user conversations sometimes quickly derail and become *inappropriate* such as hurling abuses, passing rude and discourteous comments on individuals or certain groups/communities. Similarly, some virtual agents or bots have also been found to respond back to users with inappropriate messages. As a result, inappropriate messages or comments are turning into an online menace slowly degrading the effectiveness of user experiences. Hence, *automatic detection and filtering* of such inappropriate language has become an important problem for improving the quality of conversations with users as well as virtual agents. In this paper, we propose a novel deep learning-based technique for automatically identifying such inappropriate language. We especially focus on solving this problem in two application scenarios—(a) Query completion suggestions in search engines and (b) Users conversations in messengers. Detecting inappropriate language is challenging due to various natural language phenomenon such as spelling mistakes and variations, polysemy, contextual ambiguity and semantic variations. For identifying inappropriate query suggestions, we propose a novel deep learning architecture called *"Convolutional Bi-Directional LSTM (C-BiLSTM)"* which combines the strengths of both Convolution Neural Networks (CNN) and Bi-directional LSTMs (BLSTM). For filtering inappropriate conversations, we use LSTM and Bi-directional LSTM (BLSTM) sequential models. The proposed models do not rely on hand-crafted features, are trained *end-end* as a single model, and effectively capture both local features as well as their global semantics. Evaluating C-BiLSTM, LSTM and BLSTM models on real-world search queries and conversations reveals that they significantly outperform both pattern-based and other hand-crafted feature-based baselines.

Keywords Query classification · Deep learning · Query autosuggest · Web search · Conversations · CNN + Bi-directional LSTM · Supervised learning

1 Introduction

There are a large number of online discussion fora on the internet today which are meant for users to express, discuss and exchange their views and opinions on various topics. For example, news portals, blogs, social media channels such as youtube typically allow users to express their views through comments. In such fora, it has been often observed that user conversations often derail and become *inappropriate* such as hurling abuses, passing rude and discourteous comments on individuals or certain groups/communities. In this context, we define any textual message or conversation as *inappropriate* as follows:

Definition 1 A given textual information is defined as *inappropriate* if its intent is any of the following–(a) rude or discourteous or exhibiting lack of respect toward certain indi-

This paper is an extended version of the long paper which appeared in PAKDD'2017—"Convolutional Bi-Directional LSTM for Detecting Inappropriate Query Suggestions in Web Search" [26].

✉ Manoj K. Chinnakotla
manojc@microsoft.com

Harish Yenala
hayenala@microsoft.com

Ashish Jhanwar
asjhanwa@microsoft.com

Jay Goyal
jaygoyal@microsoft.com

[1] Microsoft, Hyderabad, India

Published online: 27 December 2017

Fig. 1 Sample inappropriate query suggestions from popular web search engines

viduals or group of individuals (b) to cause or capable of causing harm (to oneself or others) (c) related to an activity which is illegal as per the laws of the country or (d) has extreme violence.

Similarly, some virtual agents or bots have also been found to respond back to users with inappropriate messages. As a result, inappropriate messages or comments are turning into an online menace slowly degrading the effectiveness of user experiences. Hence, *automatic detection and filtering* of such inappropriate language has become an important problem for improving the quality of conversations with users as well as virtual agents.

In this paper, we propose a novel deep learning-based technique for automatically identifying such inappropriate language. We especially focus on solving this problem in two application scenarios—(a) Query completion suggestions in search engines and (b) Users conversations in messengers. In the following subsections, we describe more about these application scenarios.

1.1 Query completion suggestions in SEs

Web search engines have become indispensable tools for people seeking information on the web. Query autocompletion (QAC) [1,19,22] feature improves the user search experience by providing the most relevant query completions matching the query prefix entered by the user. QAC has many advantages such as saving user time and keystrokes required to enter the query, avoiding spelling mistakes and formulation of better queries [3]. The candidates for query completion are usually selected from search query logs which record what other users have searched for [1].[1] During query time, those candidates which match the given query prefix are ranked based on various relevance signals such as time, location and user search history to finally arrive at the top k relevant completions to be displayed [3,7]. The value of k usually varies from 4–10 depending on each web search engine.

With more than a billion searches per day, search queries have become a mirror of the society reflecting the attitudes and biases of people. Hence, besides queries with a clean intent, search query logs also contain queries expressing violence, hate speech, racism, pornography, profanity and illegality. As a result, while retrieving potential completions from search logs, search engines may inadvertently suggest query completions which are *inappropriate* to the users.

Figure 1 shows a few inappropriate query completions currently shown in some popular web search engines. For example, the query completions for *"christianity is "* are *christianity is fake*, *christianity is not a religion*, *christianity is a lie* and *christianity is the true religion*. Out of these, the first three suggestions are *inappropriate* since they are likely to offend and hurt the sentiments of christians. Similarly, the first query completion suggestion for *"angelina jolie is "* is *angelina jolie is a heroin addict*. This suggestion is inappropriate as it tries to characterize her entire personality with heroin addiction. *Inappropriate* queries are different from queries with an *adult* intent such as pornography and sexuality. Adult queries are much more coherent when compared to inappropriate class which includes a large number of sub-categories within it.

Although, users still have the right to search whatever they want, a search engine offering such inappropriate completions/suggestions inadvertently may—(a) be considered as endorsing those views thereby tarnishing the brand image or (b) damage the reputation of certain individuals or communities leading to legal complications or (c) help a person who is trying to harm oneself or others. In the past, there have been some instances[2] where search engines were dragged into legal tussles over such inappropriate suggestions. Hence, due to their potential to negatively influence their users and brand, it is imperative for search engines to *identify and filter or block* such inappropriate suggestions during QAC. Once the inappropriate queries are automatically identified, they can be pruned out during candidate generation phase thereby

[1] http://googleblog.blogspot.com/2004/12/ive-gotsuggestion.html.

[2] http://searchengineland.com/google-trouble-racist-autocomplete-suggestions-uk-184031.

Table 1 Examples of misspelling problems

Type of misspelling	Examples
Leet converted words	@$$hole, 81tch (referring to bitch)
Repeated characters	Fuckkk, fucccckkkk, niggger
Short forms	F U, pls , stfu
Similar sounding words	Bich, fack off, suk my dik
Vowels removed words	Btch, fckd

blocking their passage into the next modules of the QAC pipeline.

Automatic detection of inappropriate search queries is challenging due to lack of sufficient context and syntactic structure in web queries, presence of spelling mistakes and natural language ambiguity. For instance, a query like *"defecate on my face video"* may sound extremely offensive and hence inappropriate but it is the name of a famous song. Similarly, *"what to do when tweaking alone"* is a query where the term *tweaking* refers to the act of consuming meth—a drug which is illegal and hence the suggestion is inappropriate. A query like *"hore in bible"* has a spelling mistake where *hore* refers to *whore* which makes the query inappropriate. Previous approaches [16,20,23,24] have focused on identifying offensive language or flames in the messages posted on online or social networking forums such as twitter and facebook. They mainly rely on the presence of strong offensive keywords or phrases and grammatical expressions. They also explored the use of supervised machine learning-based classification techniques, which require hand-crafted features, such as—Naive Bayes, SVMs and Random Forests, to automatically learn the target pattern with the help of labeled training data. However, such techniques are not suited well for inappropriate search query detection since the ambiguity is high, context is less and there is no linguistic structure to be exploited.

1.2 User conversations in messengers

For our current work, we collected real-world conversational data from gaming application and chat bots. Conversations are different in nature than search queries and quite diverse. The difference between query and conversations could be understood by the following characteristics:

1. In general, we found conversations to be longer than queries. Average number of words in search queries are around 3.5 words, while average number of words in conversations are around 8. Also, conversations may vary from a simple chat conversation like "hi" to a long comment. In our data, after removing top 0.1% longest conversations, we found that maximum length of conversation as 250 words. By applying the similar logic for queries, we observed the maximum length as 18 words.
2. Conversations are natural language sentences, often saying or asking something to other entity (person, bot, etc.), while queries will be mostly limited to keywords.
3. It is common to see misspelled terms in conversations. In our data, 70% of the conversations had at least one misspelled term. We observed several type of misspellings, some of the examples shown in Table 1.
4. Sometimes, the intent cannot be inferred from the user message alone and have to take context, from previous turns of conversations, into account as well.
5. Conversations also include smileys and symbols. For example,

   – you are great _/_
   – Thank you :) :D
   – Awesome job (y)

In this paper, we use deep learning-based techniques to solve both problems. For query completion suggestions, we combine the strengths of both Convolutional Neural Networks (CNN) and Bi-directional LSTM (BLSTM) deep learning architectures and propose a novel architecture called *"Convolutional, Bi-Directional LSTM (C-BiLSTM)"* for automatic inappropriate query detection. Given a query, C-BiLSTM uses a convolutional layer for learning a feature representation for the query words which is then fed as input to the BLSTM which captures the sequential patterns and outputs a richer representation encoding those patterns. The query representation thus learnt passes through a deep fully connected network which predicts the target class.

As conversations are different, we found LSTM and BLSTM with character gram embedding doing better. All three proposed models do not require any hand-crafted feature engineering, are trained *end-end* as a single model and effectively capture both local features as well as their global semantics. Evaluating them on real-world search queries and conversations reveal that they significantly outperform both pattern-based and other hand-crafted feature -based baseline models. In this context, the following are our major contributions:

- We introduce the research problem of automatic identification of inappropriate content in text.
- We propose a novel deep learning-based approach called "Convolutional, Bi-Directional LSTM (C-BiLSTM)" for identifying inappropriate query suggestions in web search.
- We evaluate the various techniques proposed so far on query suggestions, including standard deep learning techniques (such as CNN, LSTM and BLSTM), on a real-world datasets and compare their effectiveness.
- We also evaluate performance of sequential models like LSTM and BLSTM to identify inappropriate conversations.

The rest of the paper is organized as follows: Sect. 2 discusses the related work in this area, Sects. 3 and 4 present our approaches to the problem of detecting inappropriate text in search queries and conversations, respectively. In the end, Sect. 5 concludes the paper.

2 Related work

Vandersmissen et al. [20] applied machine learning (ML) techniques to automatically detect messages containing offensive language on Dutch social networking site Netlog. They report that a combination of SVM and word list-based classifier performs best and observed their models perform badly with less context which is usually the case with search queries as well. Xiang et al. [23] used statistical topic modeling (LDA) [2] and lexicon-based features to detect offensive tweets in a large scale twitter corpus. These try various models (SVMs, Logistic Regression (LR) and Random Forests (RF) with these features and report that LR performs best. It is pretty hard to extract topical features from search queries which are usually short and have less context. Razavi et al. [16] detect flames (offensive/abusive rants) from text messages using a multi-level classification approach. They use a curated list of 2700 words, phrases and expressions denoting various degrees of flames and then used them as features for a two-staged Naive Bayes classifier. Xu et al. [24] use grammatical relations and a curated offensive word list to identify and filter inappropriate/offensive language in online social forums. Chuklin et al. [5] automatically classify queries with adult intent into three categories—*black (adult intent), gray, white (clean intent)*. They use gradient boosted decision trees for classification.

Shen et al. [25] use a series of convolution and max-pooling layers to create a Convolution Latent Semantic Model (CLSM) aimed at learning low-dimensional semantic representations of search queries and web documents. CLSM uses character trigram and word n-gram features and is trained using click-through data. Results on a real-world dataset showed better performance over state of the art methods such as DSSM [12].

Researchers have tried combining CNN and LSTM architectures in the context of other text mining problems such as Named Entity Recognition (NER) and Sentiment Analysis. Zhou et al. [28] combined CNN and LSTM architectures to create a hybrid model C-LSTM and apply it for sentiment analysis of movie reviews and question type classification. Although, the combination of CNN and LSTM is similar to our current model, there are some minor differences—(a) Through Convolutional layer, we are interested in learning a better representation for each input query word and hence we do not use max-pooling since it reduces the number of input words and (b) We use a Bi-directional LSTM layer instead of LSTM layer since it can model both forward and backward dependencies and patterns in the query. Sainath et al. [18] also sequentially combine convolutional, LSTM and fully connected layers into a single architecture named CLDNN for the problem of speech recognition. Chiu et al. [4] combined Bi-directional LSTM and CNN models for NER. They augment the features learnt by the CNN with additional word features like capitalization and lexicon features for forming a complete feature vector. This complete feature vector is then fed into Bi-directional LSTM layer for sequentially tagging the words with their NER tags. Unlike them, we only use the final outputs of forward and backward layers of the Bi-Directional LSTM since we are interested in query classification.

To the best of our knowledge, we are first one to introduce the research problem of detecting search queries with inappropriate/offensive intents and also to propose an end-end deep learning model for solving it.

3 Inappropriate text detection on web search queries

3.1 C-BiLSTM for inappropriate query detection

The architecture of our proposed C-BiLSTM model is shown in Fig. 2. C-BiLSTM takes an input search query and outputs the probability of the query belonging to the inappropriate class. The input search query is fed into the model in the form of a word embedding matrix. C-BiLSTM model consists of three sequential layers—(a) Convolution (CONV) Layer (b) Bi-directional LSTM (BLSTM) Layer and (c) Fully Connected (FC) layer. Given the input query embedding matrix, the CONV Layer learns a new lower-dimensional feature representation for the input query which is then fed into the BLSTM layer. The BLSTM layer takes the CONV layer query representation as input and in turn outputs a feature representation which encodes the sequential patterns in the query from both forward and reverse directions. This fea-

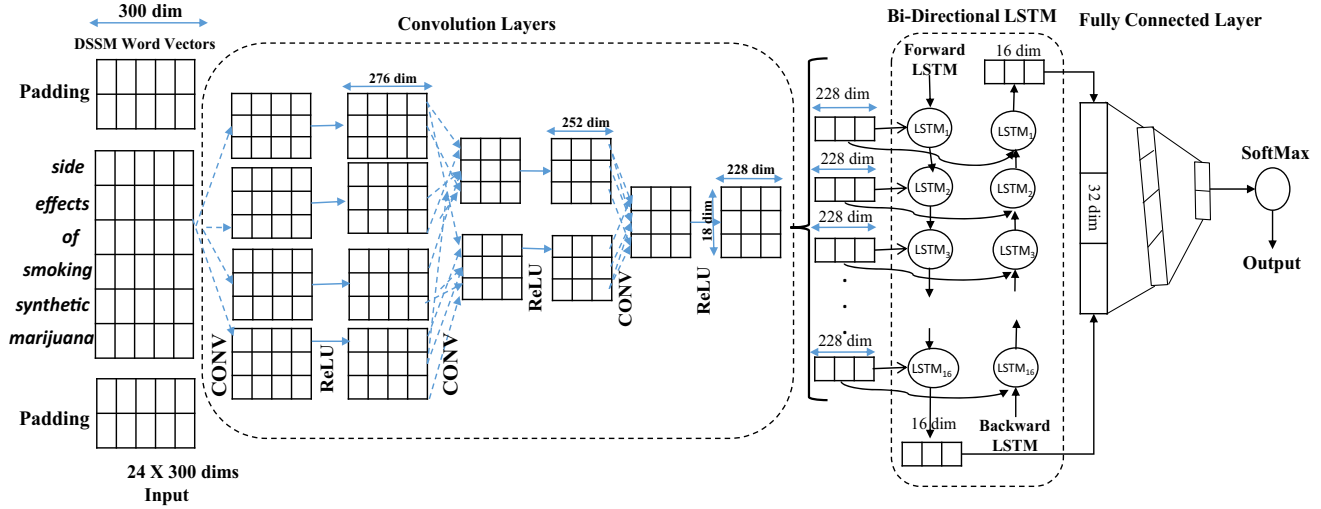


Fig. 2 Architecture of convolutional Bi-directional LSTM (C-BiLSTM) model

ture representation then passes through the FC layer, which models the various interactions between these features and finally outputs the probability of the query belonging to the inappropriate class. We share more details about each of the above layers below.

3.1.1 Input query embedding and padding

For each word in the input query, we obtain its DSSM [12] word embedding in 300 dimensions and use it to form the input query matrix. We chose DSSM for representing the words since—(a) it has been specifically trained on web search queries and (b) it uses character grams as input and hence can also generate embeddings for unseen words such as spelling mistakes and other variations which are common in search queries. As the CONV layer requires fixed-length input, we pad each query with special symbols, indicating unknown words, at the beginning and at the end to ensure the length is equal to *maxlen* (in our case 24). We randomly initialize the DSSM word vectors for these padded unknown words from the uniform distribution $[-0.25, 0.25]$.

3.1.2 Learning feature representations using convolution layer

Let $w \in \mathbb{R}^{MaxLen \times d}$ denote the entire query where $MaxLen$ is the length of the final padded query where $w_i \in \mathbb{R}^d$ be the DSSM word representation of the ith word of the input query in d dimensions. In our case, $MaxLen = 24$ and $d = 300$. Let $k \times l$ be the size of the 2-D convolution filter with weight $m \in \mathbb{R}^{k \times l}$ then each filter will produce a feature map $v \in \mathbb{R}^{(MaxLen-k+1)\times(d-l+1)}$. We consider multiple such filters and if there are n filters then $C = [v_1, v_2, \ldots, v_n]$ will be the combined feature representation of these filters. After each convolution operation, we apply a nonlinear transformation using a Rectified Linear Unit (ReLU) [14] as it simplifies back propagation and makes learning faster while avoiding saturation. In our case, we used four 3×25 filters. As shown in Fig. 2, we apply three successive steps of convolution and nonlinearity to arrive at the final feature representation which has a dimension of 18×228.

3.1.3 Capturing sequential patterns with Bi-directional LSTM

Long Short-Term Memory (LSTMs) [11] are variants of Recurrent Neural Networks (RNN) [17,21] architectures which—(a) overcome the vanishing gradient problem of conventional RNNs and (b) have the ability to capture long-term dependencies present in a sequential pattern due to their gating mechanisms which control information flow. While LSTMs can only utilize previous contexts, Bi-Directional LSTMs (BLSTMs) overcome this limitation by examining the input sequence from both forward and backward directions and then combine the information from both ends to derive a single representation. This enables them to capture much richer sequential patterns from both directions and also helps in learning a much better feature representation for the input query.

The 18×228 feature representation from the previous convolution layer is fed as a sequence of 18 words, each with a 228-dimensional representation, to the BLSTM layer. The LSTM cells inside the forward and backward LSTM networks of BLSTM read the word representations in the forward and reverse orders and each of them output a 16-dimensional representation which is then combined to produce a 32-dimensional feature representation which encodes the various semantic patterns in the query.

Table 2 Optimal hyper-parameter set for all models after tuning on validation set

Parameter	CNN	LSTM	BLSTM	C-BiLSTM
Batch size	1000	1000	1000	1000
Max len.	24	24	24	24
WordVecDim.	300	300	300	300
CNN depth	4	NA	NA	3
Filter size	2×20	NA	NA	3×25
Pooling	Max-pooling	NA	NA	NA
Nonlinearity	ReLU	NA	NA	ReLU
LSTM cells	NA	40	40	32
Optimizer	Adagrad	Adagrad	Adagrad	Adagrad
Learning rate	0.01	0.05	0.05	0.05
Epsilon	1e−08	1e−08	1e−08	1e−08

Table 3 Statistics of inappropriate categories in our evaluation dataset

Category	# Samples	Sample queries
ExtremeViolence/SelfHarm /IllegalActivity	1619	*Woman beheaded video how many pills does it take to kill yourself growing marijuana indoors for beginners*
Race/Religion/Sexual Orientation/Gender	2241	*New zealanders hate americans anti islam shirts gays are destroying this country butch clothing for women*
OtherOffensive/Celebrity	1124	*Jokes about short people louie gohmert stupid quotes*
Clean	74,057	*20 adjectives that describe chocolate*
		what is the order of the planets
Total	**79,041**	

The output of the BLSTM layer (32-dimensional feature vector) is given as input to a Fully Connected (FC) layer which models the interactions between these features. The final softmax node in the FC layer outputs the probability of the query belonging to the inappropriate class.

3.2 Model training

We train the parameters of C-BiLSTM with an objective of maximizing their predication accuracy given the target labels in the training set. We randomly split the given dataset into train, validation and test sets. We trained the models using the training set and tuned the hyper-parameters using the validation set. The optimal hyper-parameter configuration thus found for various models is shown in Table 2. If t is the true label and o is the output of the network with the current weight configuration, we used Binary Cross-Entropy (BCE) as the loss function which is calculated as follows:

$$BCE(t, o) = -(t \cdot log(o) + (1 - t) \cdot log(1 - o))$$

We use Adagrad [8] as the optimization routine and the models were trained by minimizing the above loss function in a batch size of n which was tuned differently across models as given in Table 2.

3.3 Experimental setup

In this section, we describe the details of our dataset, baseline approaches used for comparison and evaluation metrics.

3.3.1 Dataset details

For evaluating this task, we created a dataset comprising of 79,041 unique web search queries along with their class labels (inappropriate/clean) provided by human judges. The details of the dataset along with the statistics are shown in Table 3. These queries were sampled from the query suggestions being currently served through the autocompletion service of a well-known commercial search engine in the US market. The sampled queries were then judged by human judges through a crowd-sourcing platform. Each query had to be classified into one of two categories–(a) Clean (b) Inappropriate. In case of *Inappropriate*, they were also asked to mark one of the following three finer categories—(a)

Table 4 Evaluation dataset label distribution across train, validation and test sets

Label	Training	Validation	Test	*Total*
Inappropriate	4594	212	178	4984
Clean	65, 447	4788	3822	74, 057
Total	70, 041	5000	4000	79, 041

Violence/Illegal/SelfHarm (b) Race/Religion/Sexual/Gender and (c) Other/Profane. To avoid any individual bias, each query was judged by nine judges. An odd number of judges was chosen to easily decide the majority vote. The judges were given clear guidelines, along with sample query label pairs, on how to classify the queries. A query was finally labeled *inappropriate* if a majority of the judges labeled it as *inappropriate*. The inter-annotator agreement based on fleiss' kappa value [9] was found to be 0.632 which is substantial. As shown in Table 4, we randomly split the dataset into train, validation and test sets. The train set was used for training the models while the validation set was used for tuning the parameters of individual models and the test set for evaluation.

3.4 Baseline approaches

We compare our approach with three baselines—(a) Pattern and Keyword-based Filtering (PKF) and (b) Support Vector Machine (SVM) [6]-based classifier and (c) Gradient Boosted Decision Trees (BDT) [10]-based classifier. In PKF, based on manual query analysis, we curated a list of typical inappropriate query patterns and high-confidence inappropriate/offensive keywords. If a given query matches any of the inappropriate/offensive query patterns or contains one of the inappropriate/offensive keywords then it will be labeled as *inappropriate*. Some sample patterns and inappropriate/offensive keywords are shown in Table 5. SVM and BDT classifiers are supervised learning models and we implemented both of them using Scikit Learn [15]. We use the query words as features and the labeled training set for training. In order to test the efficacy of deep learning-based word embeddings, we also implemented variants of SVM and BDT approaches called SVM-DSSM and BDT-DSSM where besides regular word features, DSSM embeddings of query words were also included as additional features. We optimized the performance of all models by tuning their parameters using the validation set. In the case of SVMs, we tuned the parameter C and also tried various kernels. We also handled class imbalance in all the models by appropriately setting the class weight parameter. The best performance was found with $C = 0.5$ and linear kernel. In case of BDT, the optimal parameter choice was found to be no. of trees = 50, max. depth = 7, min. samples split=350 and min. samples leaf=10. We use the standard classification evaluation metrics—Precision, Recall and F1 score [13].

3.5 Results and discussion

Table 6 presents the overall results of various techniques on the test set of the evaluation dataset. C-BiLSTM gives the best performance on F1 score which measures the overall quality of classification and also shows significant improvement over the baseline approaches which employ either pattern/keyword-based or hand-crafted feature-based identification techniques. It is also interesting to note that the other deep learning techniques such as CNN, LSTM and BLSTM also perform significantly better than the baseline approaches. The results show that DL approaches, with the help of automatically learnt feature representations, can perform better than competitively trained hand-crafted base-

Table 5 Sample patterns and keywords used in PKF baseline

Sample Patterns	Sample Matching Queries	Sample Words in Category
Proper Noun *AND* ***NegativePersonalityWord***	*hillary clinton is terrible marie harf is an idiot*	***NegativePersonalityWord:*** *terrible, idiot, moron, miser..*
Proper Noun AND *'not'/'no' AND* ***PositivePersonalityWord***	*ellen degeneres is not funny*	***PositivePersonalityWord:*** *calm, affectionate, charming..*
SelfHarmPrefix *AND* ***SelfHarmSuffix***	*how can i commit suicide methods to kill myself*	***SelfHarmPrefix:*** *how can I, how should I, ways of…* ***SelfHarmSuffix:*** *hang myself, shoot myself, commit suicide…*
Ethnicity/Religion *AND* ***CommunityDislikeWord***	*americans hate black people muslims murdered christians*	***Ethnicity/Religion:*** *americans, jews, muslims..* ***CommunityDislikeWord:*** *hate, disrespect, kill…*
CoreOffensiveWord	*slut shaming quotes the bitch is back*	***CoreOffensiveWord:*** *fuck, asshole, bitch, slut..*

Table 6 Final results of various models on Test Set

Model	Precision	Recall	F1 score
PKF	0.625	0.2142	0.3190
BDT	0.7926	0.2784	0.4120
BDT-DSSM	**0.9474**	0.3051	0.4615
SVM	0.8322	0.3593	0.5019
SVM-DSSM	0.9241	0.4101	0.5680
CNN	0.7148	**0.8952**	0.7949
LSTM	0.8862	0.7047	0.7850
BLSTM	0.8018	0.8285	0.8149
C-BiLSTM	0.9246	0.8251	**0.8720**

C-BiLSTM and BLSTM results are found to be statistically significant with $p < 0.005$
Best values are in bold

lines. We can also observe that BDT and SVM perform better when we provide them DSSM word embedding features instead of word features alone. The results also prove that combining convolutional layer and BLSTM architectures in C-BiLSTM is better than individual CNN and BLSTM models and is especially helpful in improving precision as shown in the significant improvement of precision (more than 29% when compared to CNN). Although BDT baseline shows the highest precision of 0.9474, the recall is poor (0.3051) which means it is precise only for a small class of inappropriate queries. Figure 3 shows a comparison of the relative running times taken by the various models during train and test phases.

3.5.1 Qualitative analysis

Table 7 presents a qualitative analysis of C-BiLSTM results vis-à-vis other baseline approaches using queries from the test set. Since, C-BiLSTM uses DSSM embeddings for query words, it understands the spelling mistakes and variations of a word. Due to this, it correctly classifies the queries—*"hore in the bible"* and *"a**monkey"*, whereas the baseline approaches fail since the particular words may not have been observed in the training data or in the pattern dictionary. Similarly, C-BiLSTM correctly classifies the query *"niger-please.com"* although the offensive word is used with another word without any space. This shows that C-BiLSTM has effectively captured the character grams associated with the target label. However, there are still some queries where C-BiLSTM performs badly and needs to be improved. For example, it incorrectly classifies the query *"why do asians speak the ching chong"* which was perfectly classified by PKF due to the presence of the word *"ching chong"* in its inappropriate/offensive word list.

3.5.2 Query autocompletion filtering task

In order to demonstrate the effectiveness of the proposed techniques for query autocompletion filtering task, we randomly selected 200 unique inappropriate/offensive queries from the pool of inappropriate queries excluding training set (i.e., from Validation (212) and Test (178)). From the above set, for each query, we generated a prefix of random length. Later, for each of these 200 query prefixes, we retrieved the current top 5 autocompletion suggestions offered by a popular commercial search engine and got them labeled from human judges using the same process described earlier in Sect. 3.3.1. We call this dataset of 200 query prefixes with 5 query suggestions each as the *"Query Prefix Based Autocompletions Dataset (QPBAD)"*. We ran all the models on QPBAD and reported their average precision (at 5), average recall (at 5) and F1 scores across queries in Fig. 4. In tune with earlier observation, C-BiLSTM shows better performance in identifying inappropriate query completion suggestions based on query prefixes than the baseline and individual deep learning models.

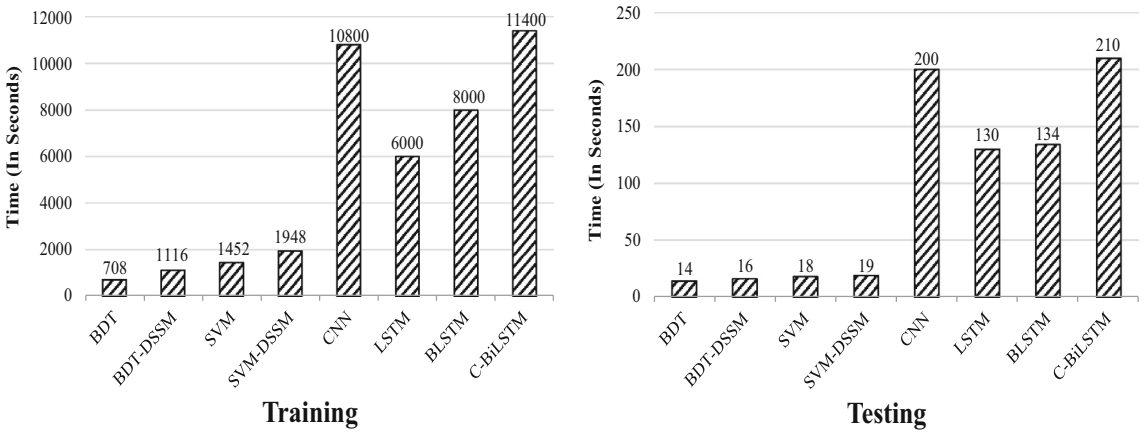


Fig. 3 Run-time performance of various models during train and test phases

Table 7 Qualitative analysis of C-BiLSTM results vis-a-vis other baseline approaches

Query	True Label	C-BLSTM Label	Judgment Explanation	Comments
nigerplease.com	Inappropriate	Inappropriate	The word niger is an Inappropriate word	Since nigerplease.com is a single word, PKF, SVM, BDT models fail. Other deep learnt models also misclassify this query. C-BLSTM alone correctly classifies this one.
shake and bake meth instructions	Inappropriate	Inappropriate	“meth” is a drug which is illegal in USA and some other parts of world	"meth" is a short form for Methamphetamine and hence PKF, SVM, BDT models fail. Other deep learnt models also misclassify this query. C-BLSTM alone correctly classifies this one.
*a**monkey*	Inappropriate	Inappropriate	It refers to the Inappropriate word—“assmonkey”	C-BLSTM perfectly classifies it. PKF, SVM, BDT fail because it includes “**”. Other deep learnt models also fail in this case.
hore in the bible	Inappropriate	Inappropriate	It is a spell mistake of the word "whore" and is Inappropriate to christians	PKF, SVM, BDT fail because of not catching the spell mistake. Other deep learnt models also fail except for C-BLSTM.
marvin gaye if i should die tonight download	Clean	Clean	Not Inappropriate since it is a song download	PKF misclassifies it as “Inappropriate” due to the presence of “die tonight” pattern. Remaining models classify correctly.
asshat in sign language	Inappropriate	Clean	An Inappropriate term in sign language	BDT perfectly classifies it. Remaining all models misclassify it.
why do asians speak the ching chong	Inappropriate	Clean	Ching chong is a pejorative term for chinese language	PKF classifies it correctly since “ching chong” is included in list of core Inappropriate words. Remaining classifiers fail.

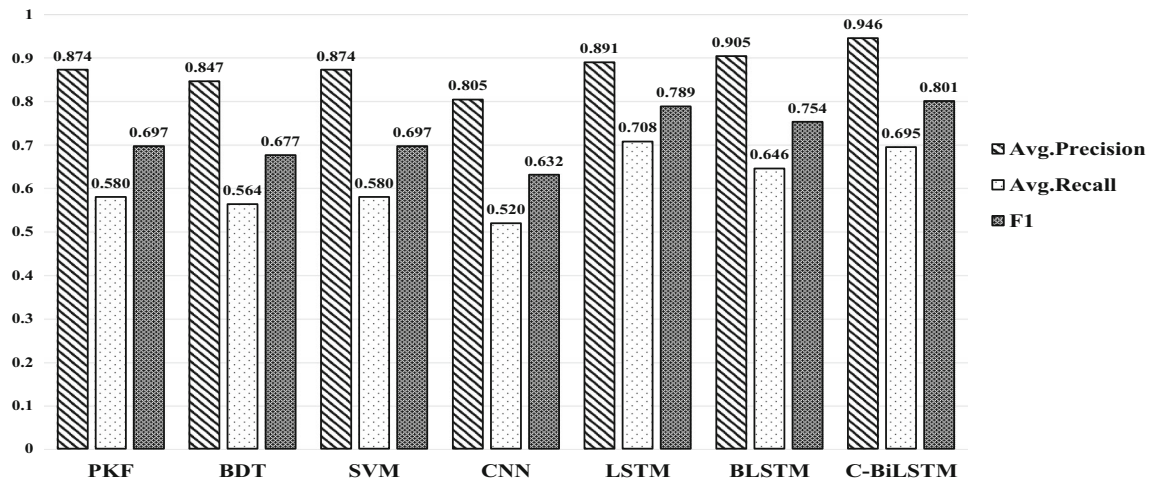


Fig. 4 Results on QPBAD dataset using top five query completion suggestions

4 Inappropriate text detection for conversations

As mentioned in Sect. 3.5, C-BiLSTM performs well on search engine queries. In this section, we focus on solving the problem of identifying inappropriate conversations with various techniques.

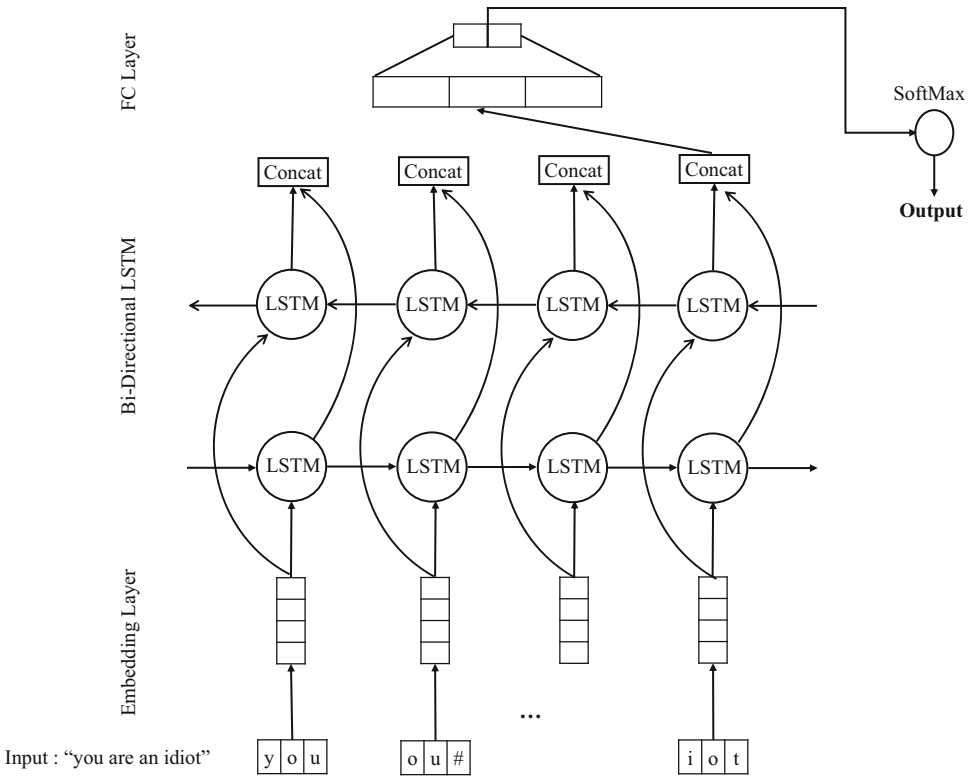


Fig. 5 Architecture of Bi-directional LSTM (BLSTM) model

4.1 LSTM and BLSTM for inappropriate conversation detection

Since, conversations are typically longer than queries and sometimes may even be as large as 200 words, we do not apply C-BiLSTM architecture since—(a) we need to fix some maximum length for the row dimension of the input matrix and if some input conversation (esp. during test time) is beyond the maximum length, we may have to discard those words. In case of query suggestions, the maximum length was fixed to 20 words and hence we could use C-BiLSTM. (b) A large input matrix size may significantly reduce the run-time efficiency of the model.

Due to the aforementioned reasons, we use sequence model architectures such as LSTM and BLSTM with character grams as inputs. Since, the number of unique character grams are limited in size, the model input vocabulary size would be always limited. The architecture of the BLSTM model is shown in Fig. 5. Our LSTM model also has a similar architecture except that they do not have bi-directional dependencies. BLSTM and LSTM take an input conversation and output the probability of the conversation belonging to the inappropriate class. Sequence of character trigrams obtained from the input conversation are fed into the model with their one-hot representation. BLSTM and LSTM models consists of three sequential layers—(a) Embedding Layer (b) Bi-directional LSTM (BLSTM) Layer or LSTM Layer and (c) Fully Connected (FC) layer. Given the input conversation, the Embedding Layer learns a new lower-dimensional feature representation for each character trigram in the conversation which are then sequentially fed into the BLSTM or LSTM layer. The BLSTM then encodes the sequential patterns in the query from both forward and reverse directions. In case of LSTM, it encodes the sequential patterns from left to right direction. This feature representation then passes through the FC layer, which models the various interactions between these features and finally outputs the probability of the query belonging to the inappropriate class. Following subsections give more details about each layer of the above architecture.

4.1.1 Input conversation embedding

As described above conversation data have different characteristics such as spelling mistakes, symbol usage and leet conversions. To deal with complex structured conversation data, we have used character level trigrams to represent each conversation. Sequence of character trigrams sent through an embedding layer which takes one-hot vector of each character trigram in the sentence and learns the lower dimension representation for it. In our case, dimension of each input character trigram (N) is 24,543 and the Embedding dimension(E) is 150. The Embedding layer here creates a matrix of dimensions NxE. It initializes the matrix with random weights and those will be learned while training the network similar to the other layer parameters.

Table 8 Optimal hyper-parameter set for models after tuning on conversation validation set

Parameter	LSTM	BLSTM
Batch size	1000	1000
Embedding dim.	150	150
LSTM cells	50	50
Optimizer	SGD	SGD
Learning rate	0.01	0.05
Epsilon	1e−08	1e−08

Table 9 Distributions of labels in conversation datasets

Dataset	Offensive	Clean	Total
Training data	27,311	220,663	247,974
Validation set	2000	8000	10,000
Zo test set	1805	2146	3951
Xbox test set	2303	7697	10,000

4.1.2 LSTM

LSTMs capture the sequential patterns in the input from left to right direction. Each character trigram of dim 150 from the previous embedding layer is fed as a sequence of trigrams, each with a 150-dimensional representation, to the LSTM layer. The LSTM cells read the character trigram representations in the forward direction and each of them outputs a 50-dimensional representation which encodes various semantic patterns in the query. The output of the LSTM layer (50-dimensional feature vector) is given as input to a Fully Connected (FC) layer which models the interactions between these features. The final softmax node in the FC layer outputs the probability of the Conversation belonging to the inappropriate class.

4.1.3 Bi-directional LSTM (BLSTM)

As described in Sect. 3.1.3, Bi-Directional LSTMs have the ability to capture much richer sequential patterns from both directions of a sequence. Each character trigram of dim 150 from the previous embedding layer is fed as a sequence of trigrams , each with a 150-dimensional representation, to the BLSTM layer. The LSTM cells inside the forward and backward LSTM networks of BLSTM read the character trigram representations in the forward and reverse orders and each of them outputs a 25-dimensional representation which are then combined to produce a 50-dimensional feature representation which encodes various semantic patterns in the query. The output of the BLSTM layer (50-dimensional feature vector) is given as input to a Fully Connected (FC) layer which models the interactions between these features. The final softmax node in the FC layer outputs the probability of the Conversation belonging to the inappropriate class.

Table 10 Examples of offensive and clean labels in conversation dataset

Category	Examples
Offensive	dude i hope you burn in hell
	your awful drink bleach and delete smite
	go fuck yourself fuckin faggot
	hindu are shit sweaty cunts
	f off u american prick go eat some tacobell
	i bet you guys have aids
Clean	you're a great guy. you know that
	u being a good boy
	im in ur game join my party
	wat should i buy

Table 11 Final results of various models on Zo Set

Model	Precision	Recall	F1 score
PKF	0.865	0.427	0.572
BDT	**0.964**	0.451	0.614
LSTM	0.93	0.833	0.879
BLSTM	0.921	**0.867**	**0.893**

LSTM and BLSTM results are found to be statistically significant with $p < 0.005$
Best values are in bold

4.2 Model training

We train the parameters of LSTM and BLSTM with an objective of maximizing their predication accuracy given the target labels in the training set. We randomly split the given dataset into train, validation and test sets. We trained the models using the training set and tuned the hyper-parameters using the validation set. The optimal hyper-parameter configuration thus found for various models is shown in Table 8. If t is the true label and o is the output of the network with the current weight configuration and BCE as the loss function. We use Stochastic Gradient Descent [27] as the optimization routine and the models were trained by minimizing the above loss function in a batch size of n which was tuned differently across models as given in Table 8.

4.3 Details of conversation datasets

We obtained data from 2 sources:

1. Xbox Conversations (Xbox is gaming console device)

Table 12 Final results of various models on Xbox set

Model	Precision	Recall	F1 score
PKF	0.789	0.413	0.542
BDT	**0.935**	0.544	0.687
LSTM	0.857	0.731	**0.789**
BLSTM	0.828	**0.749**	0.786

LSTM and BLSTM results are found to be statistically significant with $p < 0.005$
Best values are in bold

2. Zo Conversation (Zo is a chat bot.)[3]

We have provided label distribution in Table 9 and examples in Table 10.

4.4 Quantitative analysis on conversation sets

Table 11 shows the Precision, Recall and F1 scores with various models on Zo dataset. Table 12 shows the results on Xbox dataset with various techniques. We notice that the LSTM and BLSTM models perform significantly better than the feature-based ML baseline methods in both the datasets—Zo and Xbox.

4.5 Error analysis

We looked at our errors and categorized them in multiple categories. Since it is deep learning model, it is hard to pin point the exact reason behind error. Our analysis is based on looking at outputs and finding the likely reason for the errors. The distribution of errors is shown in Fig. 6.

1. Lack of Semantic Understanding—There are two type of sentences in this category. First type of sentences do not have any specific offensive term but entire sentence becomes offensive, which makes this category as hardest problem. Some examples are as following:

   (a) *i highly doubt that you are not stupid*
   (b) *you are the tickle to my pickle*

   The second type of sentences is classified incorrectly as offensive by classifier. Like any machine learning algorithm, our model also faced false positives. Some examples are as following:

   (a) *and they had to saw the dumbbell off*
   (b) *link me a picture i cannot sleep*

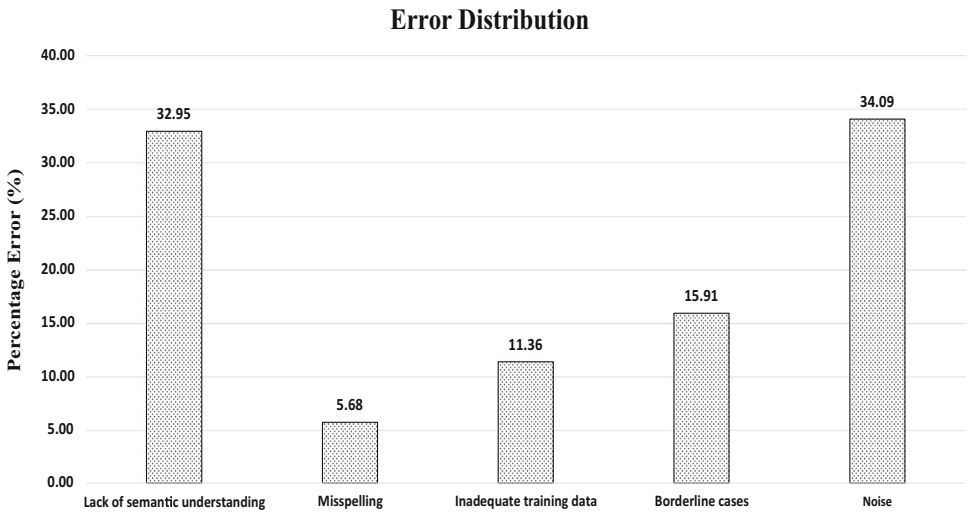
2. Misspellings—As we have mentioned earlier that misspelling is one of the major problem. Even though we could identify most of the misspelled offensive terms, we missed a few of them. Some examples are as following:

   (a) *i believe everything says it might not be vivid but it feels pr0n related*—the term "pr0n" is misspelling of term "porn".
   (b) *only cause you stoopid as fook*—terms "stoopid" and "fook" are misspellings of terms "stupid" and "fuck," respectively.

3. Inadequate training data—It is not feasible to get good representation of all offensive terms in training data. Some offensive terms are used rarely and hence we could not find their presence in training data. As a result, our models could not classify such terms or sentences containing such terms as offensive.

[3] https://www.zo.ai/

Fig. 6 Error distribution chart



(a) *yep coontastic*—term "coontastic" is urban slang term, which has offensive meaning.
(b) *if your gimpy face turned up to do my house i would make you do it too*—term "gimpy" is offensive here.

4 Borderline cases—Since we rely on human judges to get labels, many times different judges give different judgments for the same text. For example, in sentence *what the frick i was at the pool, shut up you harsh, etc.* may be considered nonoffensive by some judges, while offensive by other judges. While we consider such texts as offensive and have specified it in judgment guideline, still sometimes judges mark such text as nonoffensive due to their own perspective. This results into ambiguous training data and we end up misclassifying such cases.
5 Noise—This is judgment noise. In these cases, we do not agree with judges.

5 Conclusions

We introduced the problem of automatically detecting inappropriate content in text. We considered two application scenarios for solving problem—(a) query completion suggestions in web search and (b) User conversations in messengers. We proposed novel deep learning-based techniques for automatically identifying such inappropriate language. The proposed models do not rely on hand-crafted features, are trained *end-end* as a single model, and effectively capture both local features as well as their global semantics.

For identifying inappropriate query suggestions, we proposed a novel deep learning architecture called *"Convolutional Bi-Directional LSTM (C-BiLSTM)"* which combines the strengths of both Convolution Neural Networks (CNN) and Bi-directional LSTMs (BLSTM). Given a query, C-BiLSTM used a convolutional layer for extracting a sequence of high-level phrase representations from query words and then fed it into a BLSTM which then captured the sequential patterns in the given query. The final output from the BLSTM was a representation for the entire query which is then passed through a fully connected network for predicting the target class. Evaluation on real-world search queries and their prefixes from a large scale commercial search engine revealed that it significantly outperforms both pattern-based and other hand-crafted feature-based baselines. C-BiLSTM also proved to be better than other individual deep learning-based architectures CNN, LSTM and BLSTM.

For filtering inappropriate conversations, we use LSTM and BLSTM sequential models with character gram inputs. Evaluating LSTM and BLSTM models on real-world conversational data reveals that LSTM and BLSTM significantly outperform both pattern-based and other hand-crafted feature-based baselines.

References

1. Bar-Yossef, Z., Kraus, N.: Context-sensitive query auto-completion. In: WWW'11, pp. 107–116. ACM, New York, NY, USA (2011)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J Mach Learn Res 3:993–1022. http://dl.acm.org/citation.cfm?id=944919.944937 (2003)
3. Cai, F., de Rijke, M.: A survey of query auto completion in information retrieval. Found. Trends® Inf. Retr. **10**(4), 273–363 (2016). https://doi.org/10.1561/1500000055
4. Chiu, J.P.C., Nichols, E.: Named Entity Recognition with Bidirectional LTM-CNNs. CoRR abs/1511.08308. http://arxiv.org/abs/1511.08308 (2015)
5. Chuklin, A., Lavrentyeva, A.: Adult query classification for web search and recommendation. In: Search and Exploration of X-Rated Information (WSDM'13) (2013)
6. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
7. Di Santo, G., McCreadie, R., Macdonald, C., Ounis, I.: Comparing approaches for query autocompletion. In: SIGIR '15, ACM, New York, NY, USA (2015)
8. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-24.html (2010)
9. Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychol. Bull. **5**, 378–382 (1971)
10. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning, vol. 1. Springer series in statistics Springer, Berlin (2001)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
12. Huang, P.S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using Clickthrough Data. In: CIKM'13 (2013)
13. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
14. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: ICML-10, June 21–24, 2010, pp. 807–814. Haifa, Israel (2010)
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Vand, M.B., Thirion, G.O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
16. Razavi, A.H., Inkpen, D., Uritsky, S., Matwin, S.: Offensive language detection using multi-level classification. In: AI'10, Springer-Verlag, pp. 16–27 (2010)

17. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cogn. Model. **5**(3), 1 (1988)
18. Sainath, T., Senior, W.A., Vinyals, O., Sak, H.: Convolutional, long short-term memory, fully connected deep neural networks. US Patent App. 14/847,133, http://www.google.com/patents/US20160099010 (2016)
19. Shokouhi, M., Radinsky, K.: Time-sensitive query auto-completion. In: SIGIR '12, pp. 601–610. ACM, New York, NY, USA (2012)
20. Vandersmissen, F.D.T, Baptist., Wauters, T.: Automated Detection of Offensive Language Behavior on Social Networking Sites, pp. xiv, 81 p.: ill (2012)
21. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proc. IEEE **78**(10), 1550–1560 (1990). https://doi.org/10.1109/5.58337
22. Whiting, S., Jose, J.M.: Recent and robust query auto-completion. In: WWW'14 (2014)
23. Xiang, G., Fan, B., Wang, L., Hong, J., Rose, C.: Detecting offensive tweets via topical feature discovery over a large scale twitter corpus, pp. 1980–1984. http://doi.acm.org/10.1145/2396761.2398556 (2012) https://doi.org/10.1145/2396761.2398556
24. Xu, Z., Zhu, S.: Filtering offensive language in online communities using grammatical relations. In: Proceedings of the Seventh Annual CEAS 2010 (2010)
25. Yelong, S., Xiaodong, H., Jianfeng, G., Li, D., Gregoire, M.: A latent semantic model with convolutional-pooling structure for information retrieval. In: CIKM (2014)
26. Yenala, H., Chinnakotla, M., Goyal, J.: Convolutional Bi-directional LSTM for Detecting Inappropriate Query Suggestions in Web Search. Springer International Publishing, Cham, pp 3–16 (2017). https://doi.org/10.1007/978-3-319-57454-7_1
27. Zhang, T.: Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: Proceedings of the Twenty-first International Conference on Machine Learning, ACM, New York, NY, USA, ICML '04, p. 116,(2004). https://doi.org/10.1145/1015330.1015332
28. Zhou, C., Sun, C., Liu, Z., Lau, FCM.: A C-LSTM Neural Network for Text Classification. CoRR abs/1511.08630 (2015)