Sentiment Classification with Convolutional Neural Networks: an Experimental Study on a Large-scale Chinese Conversation Corpus

Lei Zhang
Email: lei.zhang@xiaoi.com

Chengcai Chen
Email: arlene.chen@xiaoi.com

***Abstract*—Previous research on sentiment classification by machine learning algorithms has shown that they usually work well with large-scale dataset. However, most open datasets for Chinese sentiment classification are quite small. In this paper we build a large-scale annotated Chinese sentiment dataset by filtering a vast amount of human-computer conversations. We conduct thorough experiments by using Convolutional Neural Networks (CNNs) and other classical machine learning methods. The experimental evaluation is made on a human-annotated dataset and COAE2014 task. The extensive experiments demonstrate that Chinese sentiment classification task can benefit from our dataset and CNNs model can achieve better performance than classical machine learning approaches.**

***Keywords*-deep learning; Chinese sentiment classification; Convolutional Neural Networks**

I. Introduction

Sentiment analysis or option mining is the automated extraction of writer's attitude from the text, typically a sentence or a review. In the simplest settings, sentiment analysis can be seen as a binary classification between positive and negative sentiment. In this paper, we focus on the problem of binary sentiment classification on short text, typically a sentence or a short paragraph.

Sentiment classification is not a straightforward task. Generally speaking, existing works may be camped into two major approaches, namely sentiment knowledge based approaches and machine learning based approaches. The former approaches mainly utilize the sentiment lexicon, rules and pattern matching for sentiment analysis. They build a lexicon of words with positive and negative polarities, and identify the attitude of the author by comparing words in the text with the lexicon. However, the rules and patterns of sentiment are very difficult to achieve.

The machine learning approaches employ machine learning algorithms to build classifiers from texts with manually annotated sentiment polarity. The traditional machine learning approaches are heavily based on engineered features, but it is very difficult to handcraft features to extract properties mentioned. This becomes an even harder problem especially in cases when the amount of labelled data is relatively small, e.g., thousands of examples.

Convolutional Neural Networks (CNNs) have recently been shown to achieve impressive results on sentiment classification across multiple datasets [1], [2]. CNNs can capture the underlying semantic information of input texts, such as dependency relations, co-references and negation scopes. CNNs-based models require practitioners to specify many accompanying hyper-parameters and exploring the space of possible configurations is extremely expensive.

Previous research on machine learning areas has shown that the quality and quantity of annotated dataset are the keys to the algorithms. However, most open datasets for sentiment classification are quite small, especially Chinese. Most of these datasets are originated from user-generated content such as microblogs or user reviews. In this paper, we utilize another type of user-generated content: human-computer conversation data. Xiaoi robot[1], as an intelligent human-machine interaction robot for more than 10 years, has chatted with humans over billions of times. We build a large-scale dataset for sentiment classification by selecting the sentiment sentences from the massive amounts of conversation data.

In this paper, we conduct experiments on sentiment classification using CNNs model. Our model is most similar to the deep learning systems presented in [1]. The model is trained on top of pre-trained word embeddings obtained by unsupervised learning on a large text corpus. The training of CNNs requires exhaustive tuning of many parameters, which is very time-consuming. Our aim is not to propose a model to obtain state-of-the-art performance but to derive practical advice from our experimental results for those interested in CNNs in real world settings. The experimental evaluation is made on several datasets.

The main contributions of this work can be summarized as follows: First, to the best of our knowledge, no previous work exploits the human-computer conversation data for sentiment classification. We leverage massive chat logs with emotional terms as supervised corpus with little human effort. Second, we conduct thorough experiments and provide practical guidance for Chinese sentiment classification tasks with CNNs model.

II. Our Approach for Sentiment Classification

The architecture of our approach is inspired by Kim [1]. Since our training process requires to run the network on

[1]http://www.xiaoi.com

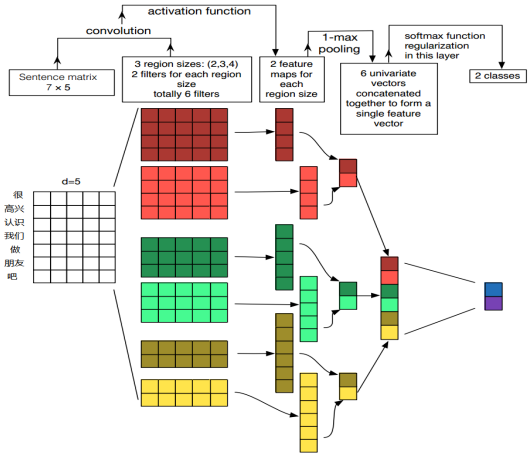

Figure 1. Illustration of our CNNs model, most similar to [2], [1]. Three filter region sizes (filter heights) are given as examples: $(2, 3, 4)$, each of which has 2 filters. Filters perform convolution on the sentence matrix and generate feature maps. Each convolutional layer is followed by a non-linear activation function. 1-max pooling is performed over each map and the largest number of each feature map is reserved. Then a concatenated feature vector is generated from 6 maps. The final softmax layer receives this feature to classify the sentences.

a rather large corpus, we adopt a single level architecture, which has been shown in [1], [2] to perform quite well. Our model is composed of a single convolutional layer, followed by a max-over-time pooling layer and a standard softmax fully connected layer at the end. Fig 1 illustrates the model.

The input to our model are sentences each treated as a sequence of words:$[w_1, w_2, ..., w_n]$ where each word is represented by a distributional vector. These vectors can be computed by word2vec [3]. We denote the dimensionality of the word vectors by d. For each input sentence we build a sentence matrix. We can perform convolution on the sentence matrix via linear filters. Filters are applied to every possible window of words in the sentence and a feature map is produced as a result. For each of the filters, a weight matrix and bias term are learned. The width of the filters are set to d. We can modify the height(size) of the filter, i.e. the number of adjacent rows considered jointly. One may use multiple filters for the same size to learn complementary features from the same regions.

To enable the learning of non-linear decision boundaries, the convolutional layer is typically followed by a non-linear activation function. The most common choices of activation functions are: sigmoid, hyperbolic tangent (tanh) and a rectified linear unit (ReLU) function defined as simply $max(0, x)$ to ensure that feature maps are always positive. We use ReLU as activated function since its efficiency.

The output from the convolutional layer (passed through the activation function) is then passed to the pooling layer, whose goal is to aggregate the information and reduce the representation. We use max pooling in our model. The output of the penultimate convolutional and pooling layers is passed to a fully connected softmax layer. It computes the probability distribution over the labels.

III. Large-scale Chinese Sentiment Dataset

The success of supervised systems largely depends on the amount and quality of training data. In this section, our goal is to construct an annotated sentiment dataset for training a smart robot able to identify human's emotion, so we exploit large-scale human-computer conversion corpus.

Sentiment lexicons are lists of words with associations to positive and negative sentiments. The manually created lexicons we used include the Chinese Vocabulary for Sentiment Analysis (VSA) released by HowNet[2] and a lexicon released by Songbo Tan. We rechecked the two lexicons and removed some inappropriate words, especially single-character words. The final sentiment lexicon in our work includes 3986 positive terms and 3904 negative terms. Negation lexicon includes terms that are used to reverse the semantic polarity of a particular term. By mining through the conversation logs, we get our negation words including 不(not), 无(not have), 没(no), 否(not), 木有(not have), 非(no), 灭有(not have), 幕友(not have), 莫有(not have), 毋(no), 勿(not).

Algorithm 1 Acquire annotated sentiment dataset from human-computer conversation

Inputs: human-computer conversation logs; sentiment lexicon; negation lexicon
Output: annotated sentiment dataset

1: Reserve only human's sentences and remove all questions
2: Reserve sentences satisfied $Min \leq length \leq Max$
3: A sentence is considered positive if it has one of the positive term, and negative if it has one of the negative term
4: Remove the sentences which are both positive and negative
5: Flip the polarity if a negation word is found immediately before a sentiment term. Remove the sentence if a negation word is found several words before a sentiment term.
6: Segment sentences and remove non-Chinese words
7: Remove all one-word sentences and de-duplicate the sentences

The basic idea of Algorithm 1 is that since the conversation logs are huge, we only need reserve the sentences with high possibility of sentiment. All the dubious sentences are removed straightforwardly including the sentence has both positive and negative terms. We do not utilize syntactic structure or intensifier word since people always use simple syntax when talking to a robot. The question sentence can be determined by whether it has one of the interrogative words, such as "怎么" (how, why), "什么" (what), "好不好" (good or not), or whether it ends with question mark "?" . In our experiment, we set $Max = 20$ and $Min = 2$. One-word sentences are removed because we found these sentences are always ambiguous.

[2]http://www.keenage.com/html/e_index.html

Table I
THE SUMMARY STATISTICS FOR OUR DATASETS

Dataset	Size	$\#Positive$	$\|Vocabulary\|$	$Length$
XS_30k	30,000	17,462	18,865	8.23
XS_100k	100,000	58,458	38,590	8.26
XS_1m	1,000,000	586,008	143,942	8.26
XS_3m	3,324,300	1,949,476	280,575	8.26
XS_test	11,576	5307	8513	6.09
XS_logs	89,060,315		2,658,133	5.40

Table II
THE QUALITY OF OUR SENTIMENT DATASET(RANDOMLY 2000 EXAMPLES FROM XS_3M FOR HUMAN ANNOTATION)

Dataset \ Actual	$Positive$	$Negative$	Neutral
$Positive$	929	23	48
$Negative$	13	946	41

Through Algorithm 1 we get about 3 million sentiment data from about 10 billion conversation logs which were generated before 2015. We randomly take about 15 thousand for human annotation and get a dataset named XS_test[3] for test purpose. We can characterize XS_test as accurate and trustworthy. The remaining data compose XS_3m. We randomly select some data from XS_3m to obtain XS_30k, XS_100k, XS_1m. We also get a dataset named XS_logs by dumping human's chats from the conversation logs in the year of 2015. The summary statistics of these datasets are listed in Table I. The average length of XS_logs is obviously shorter than those in XS_3m due to without filtering out one-word sentences.

To judge the quality of our sentiment dataset, we randomly select 2000 instances from XS_3m and annotate them humanly, then we get Table II. The accuracy of our dataset is about 93.8%. If the neutral or ambiguous data are not considered as wrong data, the accuracy is as high as 98.2%.

The most frequent notional words in XS_3m are listed in Table III. It is not surprising that the word "喜欢(like)" is frequent as both positive and negative word. Our dataset has many instances match "negation+sentiment" style, such as " 不喜欢" (not like). The word " 机器人(robot)" is frequent because many people express their opinions about robot when they chat to a robot. Some examples in XS_3m are listed in Table IV. These sentences are always vivid and oral and most of them express sentiment or opinion clearly. The data tend to be short since the long and complex sentences are likely to be removed.

[3]Released at https://github.com/z17176/Chinese_conversation_sentiment.

Table III
THE MOST FREQUENT NOTIONAL WORDS IN XS_3M SORTED BY FREQUENCY FROM HIGH TO LOW. 喜欢(LIKE) IS ABOUT 275,000 TIMES WHILE 最好(BEST) IS ABOUT 29,000 TIMES

Positive	喜欢(like),好(good),说(say),美女(beauty), 机器人(robot), 谢谢(thank), 开心(joyful),好玩(fun), 可爱(lovely), 好看(good-looking),漂亮(pretty), 知道(know),聪明(clever), 好吃(delicious), 好听(songful), 最好(best)
Negative	不(no),不好(bad),无聊(boring),机器人(robot), 不行(no way), 讨厌(disgusting), 死(die),骂人(abuse), 生气(angry), 喜欢(like), 垃圾(rubbish), 伤心(sad), 没用(useless), 不好意思(excuse me), 麻烦(trouble), 傻(idiot)

Table IV
SOME EXAMPLES FROM XS_3M(ENGLISH TRANSLATION THROUGH WORD BY WORD).

Positive	Negative
很 满意 很 满意(very satisfied very satisfied)	看 很 难过(look very sad)
最 喜欢 那个 男主角(mostly like that actor)	那 不行 得(so not allowed)
今天 工作 不错(today work well)	无 语 不好玩(noting to say no fun)
觉得 挺 聪明 再见(feel so smart bye)	脑袋 有病 还是 咋 滴(head sick or else)
好 漂亮 附件(so pretty accessory)	妈 笨小二 坑 钱(fuck stupid waiter scam)
但 喜欢 搂 着 异性 (but like embrace opposite sex)	晕死 改 个 签 而已 (faint die change sign just so so)
惊喜 给 什么 最好 (pleasantly surprised give what best)	第一次 想 骂人 冲动 (first time want criticize impulse)
这样 就 很 不错(this is so good)	就 无聊 去 吧(so boring go)
去 欢迎 去 继续 搞对象 (go welcome go continue dating)	问东 答西 狗日 就是 欠揍 (irrelevant answer fuck spanking)
好 欢迎 你常来 坐坐 (ok welcome you often come sit)	哥们 现在 心情 不 美丽 (buddy now mood not good)
好看 武侠 电影(good looking swordplay film)	美美 花心(meimei (name) flirting)
你 喜欢 小车 开去 试试 就 知道 这 感觉 (you like car drive try just know the feel)	尼玛 被 屌丝 骗 这货 根本 不会 说话 (fuck cheated by loser this stink can't talk)

IV. EXPERIMENTS

All the experiments are carried out on a workstation equipped with Intel Xeon CPU E5-2407 @2.20GHz and 192G RAM. To evaluate the quality of word vectors, we first learn word embeddings from text corpus then apply them to produce result via our CNNs model.

A. Pre-trained Word Vectors

We use word2vec to compute distributed word vectors. Our preliminary work has compared the skip-gram and continuous bag-of-words(CBOW) architectures, suggesting that skip-gram is slightly better than CBOW. The same conclusion can be found in [4]. Thus, in this study we only use embeddings derived with the structured skip-gram approach. We fix the iter parameter to 10 (default is 5) since running more training iterations can get better performance. The parameters in CNNs model can be seen in Section IV-B.

1) Effects of Corpus Size and Dimension.: First we investigate how the corpus size and word vectors dimension (-size parameter) influence the performance. We use two large-scale corpora to compute word vectors: XS_3m and XS_logs. In the meantime we switch the dimension size among $\{50, 100, 200, 300, 500, 800\}$. Table V(a) shows that by increasing the value of dimension we acquire progressive improvements and the performance reaches to the peak point at 300. After that, better validation performance is achieved but test performance drops. By comparing the performance between XS_3m and XS_logs, we found that training on a large corpus generally improves the quality of word embeddings. The word vectors trained from XS_3m are easier to overfitting than those from XS_logs. For example, when $d = 300$, the validation accuracy by XS_3m is better than by XS_logs, but the test performance is opposite. This result indicates that a large unlabelled corpus is useful for the text representation learning process. In the following

Table V
EFFECTS OF PARAMETERS IN WORD2VEC

(a) Effects of dimension and training corpus. The upper value in each cell reports validation accuracy as mean value calculated via 10-fold cross-validation on XS_100k and the lower value is test accuracy on XS_test. We will use this format for all following tables.

Training set	$d = 50$	100	200	300	500	800
XS_3m	97.41	98.10	98.46	98.51	98.78	98.69
	85.32	86.83	86.68	**87.43**	86.11	85.71
XS_logs	94.93	96.17	96.90	97.47	97.55	97.70
	87.23	87.83	88.34	**88.80**	88.76	88.75

(b) Effect of size parameter

1	2	3	4	5	6	7	8
97.45	97.28	97.46	97.24	97.47	97.43	97.47	97.43
88.82	88.88	**89.24**	88.54	88.80	88.50	88.44	88.67

(c) Effect of sample parameter

0	$1e-1$	$1e-2$	$1e-3$	$1e-4$	$1e-5$
97.39	97.45	97.52	97.47	97.10	97.08
88.51	89.12	88.79	88.80	**88.95**	88.92

(d) Effect of minimum count parameter

Training set	1	2	3	5	7	9
XS_30k	96.37	96.20	96.40	96.57	96.30	96.37
	87.92	87.65	88.04	88.32	88.01	**88.46**
XS_100k	97.64	97.62	97.65	97.57	97.72	97.65
	88.57	**88.77**	88.70	88.56	88.76	88.64

Table VI
EFFECTS OF PARAMETERS IN CNNS

(a) Effects of filter sizes (fix the number of feature maps at 100)

Filter sizes	Val accuracy	Test accuracy	Training time(hours)
(1)	96.1	85.4	2.5
(2)	96.6	87.7	4
(3)	96.0	86.9	6
(5)	95.9	88.1	10
(7)	95.9	88.4	13.5
(9)	95.8	87.4	20
(1,2)	97.0	87.7	6
(2,3)	96.8	87.8	10
(1,2,3)	97.1	**88.2**	12
(2,3,4)	96.6	87.6	18
(2,3,5,7)	96.5	88.8	35
(2,3,4,5,6,7)	96.7	88.7	55
(1,2,3,5,7,9)	96.8	**89.0**	60

(b) Effect of the number of feature maps (fix filter sizes as (1,2,3))

Feature maps	Val accuracy	Test accuracy	Training time(hours)
50	95.3	85.8	0.5
100	95.4	86.3	1
200	95.7	86.8	2.5
400	95.5	**87.1**	5
800	95.5	86.7	10.5
1000	95.3	86.4	11

experiments we use 300-dimensional word vectors trained on XS_logs.

2) Effects of Window and Sample.: The window parameter determines the max skip length between words when training and the default value is 5. The sample parameter sets threshold for occurrence of words and the default value is 0.001. Those that appear with higher frequency in the training set will be randomly down-sampled. As Table V(b) and Table V(c) show, the default values of these two parameters can get impressive performance, slightly worse than the best value. Tuning these two parameters is dispensable since the default value works so well. The best value for window parameter value is 3 partly because the sentences in our dataset are generally short.

3) Effect of Min-count.: We can see that the performance is not sensitive to this parameter from Table V(d). Setting a low value is not much helpful and we can suggest two possible reasons. First, more rows in the sentence matrix make the matrix sparse. Second, the words with low count have much less context windows to compute, so the word vectors are inaccurate and have less semantic information.

In the following experiments, word vectors are trained on XS_logs with $size = 300, window = 3, iter = 10, cbow = 0$. For words that are not present in the vocabulary, we use random initialization. To encode the sentiment information in word embeddings, we use non-static word vectors in our last model.

B. CNNs Experiments Setup

The choice of hyper-parameters of CNNs model may affect the final performance. In our experiments we reused some of hyper-parameters reported in [1], such as a mini-batch size of 50, MLP hidden units as 50. We use stochastic gradient descent (SGD) to train the network and use back-propagation algorithm to compute the gradients. We opt for the Adadelta update rule to automatically tune the learning rate. We set initial learning rate as 0.001. We also use Dropout [5] to help the model converge better (dropout ratio as 0.5). The network is trained for 50 epochs without early stopping. We conducted the following experiments to choose two important parameters: filter sizes and feature maps.

1) Effects of Filter Sizes: We first perform a coarse line-search over a single filter size to find the "best" size for the dataset. From Table VI(a) we conclude the best candidate single filter size could be 7, 5 or 2. The best multiple filter sizes combination is (1,2,3,5,7,9). However, the training time for this filter combination is 60 hours on XS_100k dataset, making it impossible for applying to XS_3m dataset. Considering the performance and efficiency we choose (1,2,3) in our last model. This filter size combination is like (uni,bi,tri-gram) features in classical models.

2) Effects of Feature Maps: Table VI(b) shows that increasing the number of feature maps beyond 400 often hurts performance (likely due to overfitting). Another salient practical point is that it takes longer to train the model when the number of feature maps is increased. We choose 100 in our last model.

C. Baseline Methods

In this work the baseline methods for sentiment classification include:

1) SVC: Support Vector Classification by Crammer and Singer[6]. We modified the implementation in [7]
2) LR: Logistic Regression implemented in [7]
3) NBSVM: Naive Bayes SVM [8]

D. Results and Discussions

We first report the performance achieved on XS_test. To test the influence of the size of labelled data, we trained models on XS_30k, XS_100k, XS_1m and XS_3m datasets individually. From Table VII(a) we can see that more labelled data we use, better accuracy we get. For each training dataset, CNNs model achieves the best performance. It proves that CNNs model can effectively compose the semantic representation of texts and capture more contextual information of features compared with traditional methods based on bag-of-words model.

By comparing the 2nd column and the last column in Table VII(a), an interesting observation can be made. When the dataset has 30k training instances, CNNs model gets much better performance than $SVC_{uni,bi}$ but the advantage is not so obvious on XS_3m dataset. Furthermore, there is notable difference in training time between CNNs and SVC models. It took less than 6 hours to train $SVC_{uni,bi,tri}$ model on XS_3m dataset, while 40 days(50 epochs) for CNNs. So the experimental results demonstrate CNNs can work better but some classical models can work well and efficiently when the dataset is big enough.

We also did experiments on the following test datasets:

1) COAE2014: the dataset used in COAE2014 task 4 having 5000 microblogs with annotated polarity. The average length is 21.3.
2) Hotel_15k: a Chinese hotel reviews dataset presented in [9]. We randomly select 15000 positive and negative instances half and half to build this dataset.

All the sentences in COAE2014 and Hotel_15k are segmented then fed into the $SVC_{uni,bi}$ and CNNs models both trained on XS_3m dataset, without any other preprocessing. Table VII(b) shows that each accuracy is better than 71% and CNNs can get as high as 76.92% for COAE2014. It is important to note that sentences in these two test datasets are much lengthier than the training dataset XS_3m. Furthermore, our models trained on XS_3m have no extra knowledge about microblogs or hotel reviews. So in these two experiments the training and test data are drawn from different underlying distribution. Considering these are challengeable cross-domain classification tasks, the accuracy over 71% is acceptable.

Table VII
RESULTS FOR XS_TEST, COAE2014 AND HOTEL_15K DATASETS

(a) XS_test as test dataset(models trained on different training datasets)

	XS_30k	XS_100k	XS_1m	XS_3m
$SVC_{uni,bi}$	81.29	83.86	88.99	95.34
$SVC_{uni,bi,tri}$	81.62	84.21	89.15	**95.54**
$SVC_{uni,bi,tri,four}$	81.89	84.22	89.18	95.32
$LR_{uni,bi}$	80.90	83.85	88.01	91.71
$LR_{uni,bi,tri}$	81.88	83.40	87.98	91.71
$LR_{uni,bi,tri,four}$	81.84	83.06	87.84	91.55
$NBSVM_{uni,bi}$	81.18	83.74	86.57	86.34
$NBSVM_{uni,bi,tri}$	81.04	83.77	85.90	86.20
$CNNs$	**87.12**	**88.75**	**91.94**	**96.05**

(b) COAE2014 and Hotel_15k as test datasets (models trained on XS_3m)

Model	COAE2014	Hotel_15k
$SVC_{uni,bi}$	74.14	71.94
CNNs	76.92	72.96

V. CONCLUSION

In this paper we propose an effortless method to extract sentiment instances from large-scale human-computer conversation logs as labelled data. We evaluate our dataset with CNNs model and some classical machine learning models. We conclude that CNNs model performs well on the task of sentiment classification. When the data is big, classical method, such as SVC, can also achieve good performance with less time-consuming. Experiments on COAE2014 and a hotel review datasets prove that our dataset can benefit sentiment classification task.

REFERENCES

[1] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, 2014.

[2] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[4] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification." in *ACL (1)*, 2014, pp. 1555–1565.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, pp. 1929–1958, 2014.

[6] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *JMLR*, vol. 2, pp. 265–292, 2002.

[7] H. Yu, C. Ho, Y. Juan, and C.-J. Lin, "Libshorttext: A library for short-text classification and analysis," *Software available at http://www.csie.ntu.edu.tw/˜ cjlin/libshorttext*, 2013.

[8] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *ACL*, 2012.

[9] Y. Lin, H. Lei, J. Wu, and X. Li, "An empirical study on sentiment classification of chinese review using word embedding," *arXiv preprint arXiv:1511.01665*, 2015.