A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

Ye Zhang
Dept. of Computer Science
University of Texas at Austin
yezhang@utexas.edu

Byron C. Wallace
iSchool
University of Texas at Austin
byron.wallace@utexas.edu

Abstract

Convolutional Neural Networks (CNNs) have recently achieved remarkably strong performance on sentence classification tasks (Kim, 2014; Kalchbrenner et al., 2014). However, these models require practitioners to specify the exact model architecture and accompanying hyper-parameters, e.g., the choice of filter region size, regularization parameters, and so on. It is currently unknown how sensitive model performance is to changes in these configurations for the task of sentence classification. We thus conduct a sensitivity analysis of one-layer CNNs to explore the effect of architecture components on model performance; our aim is to distinguish between important and comparatively inconsequential design decisions for sentence classification. We focus on one-layer CNNs (to the exclusion of more complex models) due to their comparative simplicity and strong empirical performance (Kim, 2014). We derive practical advice from our extensive empirical results for those interested in getting the most out of CNNs for sentence classification. One important observation borne out by our experimental results is that researchers should report performance variances, as these can be substantial due to stochastic initialization and inference.

1 Introduction

Our focus in this work is on the practically important task of sentence categorization. Convolutional Neural Networks (CNNs) have recently been shown to perform quite well for this task (Kim, 2014; Kalchbrenner et al., 2014; Wang et al., 2015; Goldberg, 2015; Iyyer et al., 2015). Such models capitalize on distributed representations of words by first converting the tokens comprising each instance into a vector, forming a matrix to be used as input to the CNN (Figure 1). Empirical results have been impressive. And the models need not be complex to realize strong results: e.g., Kim (2014) proposed a straight forward one-layer CNN architecture that achieved consistent state of the art (or comparable) results across several tasks. Thus there is now compelling support to prefer CNNs over sparse linear models for sentence classification tasks.

However, a downside to CNNs is that they require practitioners to specify the exact model architecture to be used and to set the accompanying hyper-parameters. To the uninitiated, making such decisions can seem like something of a black art, especially because there are many 'free parameters' in the model that one could explore. This is in contrast to the linear models widely used for text classification, such as regularized logistic regression and linear Support Vector Machines (SVMs) (Joachims, 1998). Such models are typically induced over sparse 'bag-of-words' representations of text and require comparatively little tuning: often one needs only to set the parameter encoding the regularization strength (i.e., the model bias). Conducting a line-search to identify this parameter using training data provides a practical means of setting this hyper-parameter.

The recent work on CNNs for sentence classification alluded to above have provided the settings used to achieve reported results. However these configurations were selected via an unspecified tuning procedure performed using a development set. But in practice, exploring the space of possible configurations for CNNs is extremely expensive, for at least two reasons: (1) training these models is relatively slow, even using GPUs. For example, it takes about 1 hour to run 10-fold cross validation on SST-1 dataset (Socher et al., 2013) using a similar configuration to that described in (Kim, 2014).[1] (2) the space of possible model architectures and hyper-parameter settings is vast. For example, the simple CNN architecture we consider requires, at a minimum, specifying the following: the input word vector representations; the filter region size(s); the number of feature maps; the activation function(s); the pooling strategy; the dropout rate (if any); and the $l2$ norm constraint (if any).

In practice, tuning all of these parameters is simply not feasible, especially in light of the runtime required for parameter estimation. Our aim is thus to identify empirically the settings that practitioners should expend effort tuning, and those that are either inconsequential with respect to performance or that seem to have a 'best' setting independent of the specific dataset. We take inspiration from previous empirical analyses of neural models due to Coates et al. (2011) and Breuel (Breuel, 2015), which investigated factors that effect performance of unsupervised feature learning and the effects of Stochastic Gradient Descent (SGD) hyper-parameters on training, respectively. Here we consider the effects of configurations of the model architecture and hyper-parameter values of one-layer CNNs specifically for the task of sentence categorization. We report the results of a large number of experiments exploring different configurations of this model, run over seven sentence classification datasets.

For those interested in only the punchlines, we summarize our empirical findings and derive from these practical advice presented in Section 6.

2 Background and Preliminaries

Deep learning methods are now well established in machine learning (LeCun et al., 2015; Bengio, 2009). They have been especially successful (and popular) for image and speech processing tasks. However, recently such methods have begun to overtake traditional sparse, linear models for natural language processing (NLP) tasks (Goldberg, 2015). Much of the interest in this space has been focused on inducing distributed representations of words (Bengio et al., 2003; Mikolov et al., 2013) and jointly embedding such 'internal' representations into models for token classification (Collobert and Weston, 2008; Collobert et al., 2011) or sentence modeling (Kalchbrenner et al., 2014; Socher et al., 2013).

In (Kalchbrenner et al., 2014), the authors constructed a CNN architecture with multiple convolution layers. Their model used dynamic k-max pooling. Their model posits latent, dense, low dimensional word vectors (initialized to random values prior to inference).

Kim (2014) defined a much simpler architecture that achieves comparable results to (Kalchbrenner et al., 2014) on the same datasets. This model also represents each word as a dense, low dimensional vector (Mikolov et al., 2013). They used pre-trained word vectors, and considered two approaches: *static* and *non-static*. In the former approach, word vectors are treated as static inputs, while in the latter one dynamically adjusts (or 'tunes') the word vectors for a specific task.

Elsewhere, Johnson and Zhang (2014) introduced a similar model, but swapped in high dimensional one-hot vector representations of words. They considered two variants of this approach, *seq-CNN* and *bow-CNN*. The former completely preserves sequential structure (at the cost of operating in a very high dimensional input space) while the latter preserves some sequence, but loses order within small regions. Their focus was on classification of longer texts, rather than sentences (but of course the model could be used for sentence classification).

The relative simplicity of Kim's architecture – which is largely the same as that proposed by Johnson and Zhang (2014), modulo the word vectors – coupled with observed strong empirical performance across several datasets makes this an appealing approach for sentence classification. However, in practice one is faced with making several model architecture decisions and setting various hyper-parameters. At present, very little empirical data is available to guide such decisions; addressing this gap is our aim here.

2.1 CNN Architecture

We first describe the relatively simple CNN architecture we use in this paper. We begin with a tokenized sentence which we then convert to a *sentence matrix*, the rows of which are inferred word vectors for each token. For example, these might be outputs from the Google word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) models. We denote the dimensionality of the word vec-

[1] We ran all experiments reported in this paper using Theano package, on an NVIDIA K20 GPU.

tors by d. If the length of a given sentence (i.e., token count) is s, then the dimensionality of the sentence matrix is $s \times d$.[2] Following (Collobert and Weston, 2008), we can then effectively treat the sentence matrix as an 'image', and perform convolution on it via linear filters. In NLP applications there is inherent sequential structure to the data. Intuitively, because rows represent discrete symbols (namely, words), it is reasonable to use filters with widths equal to the dimensionality of the word vectors (i.e., d). We can then think of varying only the 'height' of the filter, which refers to the number of adjacent rows (word vectors) considered jointly. From this point on, we will refer to the height of the filter as the region size of the filter.

Suppose that there is a filter parameterized by the weight vector $\mathbf{w} \in \mathbb{R}^{h \times d}$ with region size h; $\mathbf{w}$ will contain $h \cdot d$ parameters to be estimated. We denote the sentence matrix by $\mathbf{A} \in \mathbb{R}^{s \times d}$, and use $\mathbf{A}[i : j]$ to represent the sub-matrix of $\mathbf{A}$ from row i to row j. The output sequence of the convolution operator is obtained by repeatedly applying the filter on sub-matrices of $\mathbf{A}$:

$$o_i = \mathbf{w} \cdot \mathbf{A}[i : i + h - 1], \tag{1}$$

where $i = 1 \ldots s - h + 1$, and $\cdot$ is the dot product between the sub-matrix and the filter (first do element-wise multiplication, and then summed to a single value), and the length of the output sequence $\mathbf{o}$ is $s - h + 1$. We include a bias term $b \in \mathbb{R}$ and an activation function f to each o_i, inducing the *feature map* $\mathbf{c} \in \mathbb{R}^{s-h+1}$ for this filter, where:

$$c_i = f(o_i + b). \tag{2}$$

Note that one may use multiple filters for the same region size, with the aim being that each filter learns complementary features from the same regions. One may also specify multiple kinds of filters with different region sizes (i.e., 'heights').

The dimensionality of the feature map generated by each filter will very as a function of the sentence length and the filter region size. Then a pooling function is applied to each feature map to reduce the dimension and the number of parameters to be estimated. Commonly the pooling operator is the *1-max pooling* function (Boureau et al., 2010b), which generates a uni-dimensional feature from each feature map. Alternatively, the pooling strategy can be modified to operate within equal-sized regions in the feature map, encoding salient features corresponding to each region. Together, the outputs generated from each filter map can be concatenated into a 'top-level' feature vector, the size of which will be independent of individual sentence lengths, in case of 1-max pooling.

This representation is then fed through a softmax function to generate the final classification. At this softmax layer, one may opt to apply a 'dropout strategy' (Hinton et al., 2012) as means of regularization. This entails randomly setting some values in the vector to 0. We may also choose to impose an $l2$ norm constraint, i.e., linearly scale the $l2$ norm of the vector to a specified threshold when it exceed this. During training, the objective to be minimized is the categorical cross-entropy loss, and the parameters to be estimated include the weight vector(s) of the filter(s), the bias term in the activation function, and the weight vector of the softmax function. Note that one may either opt to treat the word vectors as fixed (we will refer to this as 'static') or as additional parameters of the model, to be tuned (we will refer to this approach as 'non-static'). We explore both variants.

Figure 1 provides a simple schematic to illustrate the model architecture just described.

3 Datasets

We use the same seven datasets as in (Kim, 2014), summarized briefly as follows:

- **MR**: Sentence polarity dataset from (Pang and Lee, 2005).
- **SST-1**: Stanford Sentiment Treebank (Socher et al., 2013). Note that to make input representations consistent across tasks, we only train and test on sentences. This is in contrast to (Kim, 2014), wherein the authors trained models on both phrases and sentences.
- **SST-2**: Derived from SST-1, but paring to only two classes. We again only train and test models on sentences, excluding phrases.
- **Subj**: Subjectivity dataset from (Pang and Lee, 2005).
- **TREC**: Question classification dataset from (Li and Roth, 2002).

[2] We use the same zero-padding strategy as in (Kim, 2014).

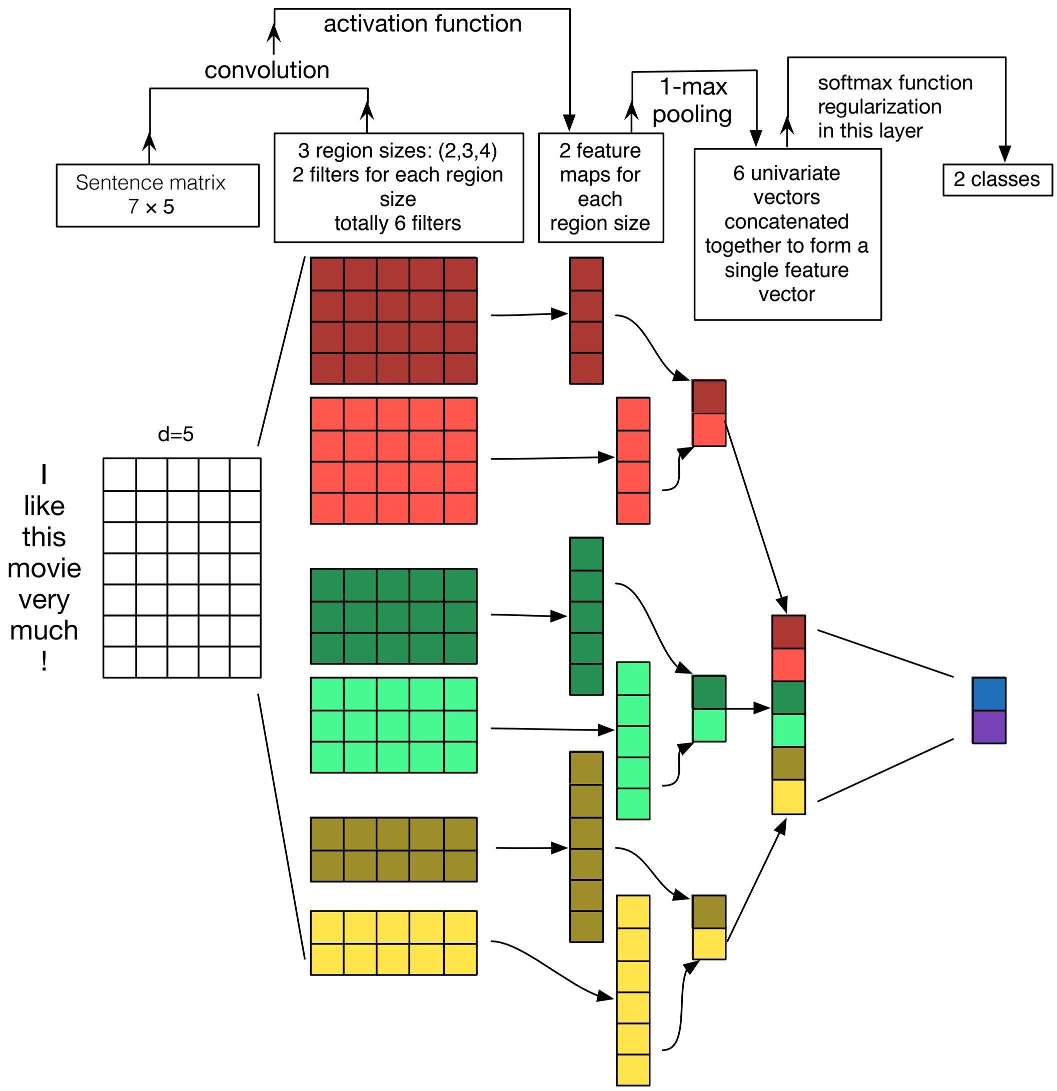

Figure 1: Illustration of a Convolutional Neural Network (CNN) architecture for sentence classification. Here we depict three filter region sizes: 2, 3 and 4, each of which has 2 filters. Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps. Then 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence; here we assume binary classification and hence depict two possible output states.

- **CR**: Customer review dataset (Hu and Liu, 2004).

- **MPQA**: Opinion polarity dataset (Wiebe et al., 2005)

We report the average length and the maximum length of the tokenized sentences for all seven datasets in Table1. For more details on these datasets, please refer to (Kim, 2014).

Dataset	Average length	Maximum length
MR	20	56
SST-1	18	53
SST-2	19	53
Subj	23	120
TREC	10	37
CR	19	105
MPQA	3	36

Table 1: Average length and maximum length of the 7 datasets

4 Performance of Baseline Models

To provide a point of reference for the CNN results, we first report the performance achieved using sparse regularized SVM for sentence classification. We used uni-gram and bi-gram features, keeping only the most frequent 30k n-grams for all datasets.

We also wanted to explore the relative gains achieved by naively incorporating embedding information directly into these models. To this end, we augmented this representation with averaged word vectors (from Google word2vec[3] or GloVe[4]) calculated over the words comprising the sentence, similar to the method in (Lai et al., 2015). We then use an RBF-kernel SVM as the classifier operating in this dense feature space. We also experiment with combining the uni-gram, bi-gram and word2vec as the feature of the sentence and use linear SVM as the classifier.

We tuned the regularization hyper-parameters via nested cross-fold validation, optimizing for accuracy. We report means from 10-folds over all datasets in Table 2.[5] For consistency, we use the same pre-processing steps for the data as described in previous work (Kim, 2014).

One thing that is immediately notable from these results is the consistent performance increase realized by even navely incorporating word2vec output into feature vectors.

5 Sensitivity Analysis of CNNs

We now report results from our main analysis, which aims to interrogate the sensitivity of CNNs for sentence classification as a function of specific architecture and hyper-parameter settings. To this end, we take as our starting point a baseline configuration (described below) which has been shown to work well in previous work (Kim, 2014). We then explore the effects of modifying components of this baseline configuration in turn, holding other settings constant.

We performed experiments using both 'static' and 'non-static' word vectors; in the former case, word vectors are not updated during inference, while in the latter case the vectors are 'tuned' to the task at hand. The non-static configuration uniformly outperformed the static variant. Therefore, we report only non-static results in this paper, although we provide results for the static configuration in the Appendix.

5.1 Baseline Configuration

We now consider the performance of a baseline model configuration of CNN. Specifically, we start with the architectural decisions and hyper-parameters used in previous work (Kim, 2014). To contextualize the variance in performance attributable to various architecture decisions and hyper-parameter settings, it is critical to assess the variance due strictly to the parameter estimation procedure. Most prior work, unfortunately, has not reported such variance, despite a highly stochastic inference procedure. This variance is attributable to estimation via Stochastic Gradient Descent (SGD), random dropout, and random weight parameter initialization. We show that mean performance calculated via 10-fold cross validation exhibits relatively high variance over repeated runs.

We first use the original parameter settings depicted in Table 3 and replicate the experiments 100 times for each dataset, in which each replication is a 10-fold CV[6], and the folds over the

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

[4] Pre-trained on common crawls of 840B tokens http://nlp.stanford.edu/projects/glove/

[5] Note that when folds are fixed, parameter estimation for SVM via QP is deterministic, which is why we report only means here.

[6] We run 10-fold CV for all datasets, which is different

Dataset	bow-SVM	wv-SVM	bow+wv-SVM	gv-SVM	bow+gv-SVM
MR	78.24	78.53	79.67	78.09	80.07
SST-1	37.92	44.34	43.15	44.31	44.29
SST-2	80.54	81.97	83.30	81.92	83.52
Subj	89.13	90.94	91.74	92.61	92.22
TREC	87.95	83.61	87.33	80.28	88.71
CR	80.21	80.79	81.31	81.75	82.27
MPQA	85.38	89.27	89.70	88.97	89.27

Table 2: Performance of SVM with different feature sets. **bow-SVM** uses only uni- and bi-gram features. **wv-SVM** uses a naive word2vec-based representation, i.e., the average (300-dimensional) word vector, calculated over the words constituting the sentence. **bow-wv-SVM** combines these feature sets by concatenating bow vectors with the average word2vec representations. For completeness, we also report analogous results using GloVe (denoted by **gv**).

replications are fixed. 'ReLU' in Table 3 refers to rectified linear unit (Maas et al., 2013), which is a commonly used activation function in CNN. We record the average accuracy over the 10-folds for each replication and report the mean, minimum and maximum mean values observed over 100 replications. We do this for both static and non-static methods. This provides a sense of the variance we might observe without any changes to the model. Results are shown in Table 4. Figure 2 provides density plots of the mean accuracy of 10-fold CV over the 100 replications for both methods on all the datasets. For clarity in presentation, we exclude SST-1, because accuracy is substantially lower on this dataset (results can be found in the tables, however). Since we split and process some datasets differently from the previous work as we have described previously, the results are also different from the original ones. And since in this work, we're only interested in the sensitivity and effect of each component of CNN on the performance, we don't care much about the absolute accuracy and won't compare the results we got with the ones in previous works.

Description	Values
input word vectors	Google word2vec
filter region size	(3,4,5)
feature maps for each region size	100
activation function	ReLU
pooling	1-max pooling
dropout rate	0.5
$l2$ norm constraint on weight vector	3

Table 3: Baseline configuration.

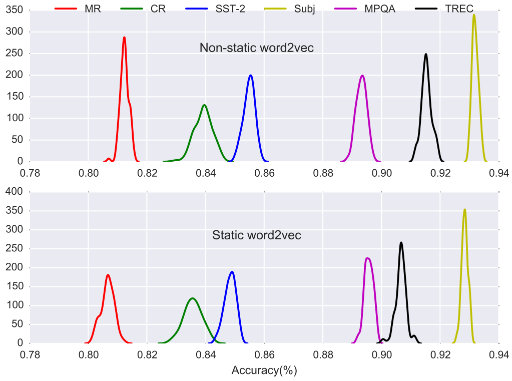

Figure 2: Density curve of accuracy using static and non-static word2vec-CNN

Having established a baseline performance for CNNs, we now consider the effect of different architecture decisions and hyper-parameter settings. To this end, we hold all other settings constant (as per Table 3) and vary only the component of interest. For every configuration that we consider, we replicate the experiment 10 times, where each replication constitutes a run of 10-fold CV.[7] Like the 100 replications of the original parameter settings, we also report the average mean, minimum mean, and maximum mean of 10 fold CV over 10 replications. For all experiments, we use the same preprocessing steps for the data as in (Kim, 2014). Similarly, we use the ADADELTA update rule for SGD (Zeiler, 2012), and set the minibatch size as 50 .

from (Kim, 2014)

[7] Running 100 replications for every configuration that we consider was simply not feasible.

Dataset	Non-static Word2vec-CNN	Static Word2vec-CNN
MR	81.24 (80.69, 81.56)	80.66 (80.16, 81.22)
SST-1	47.08 (46.42,48.01)	45.54 (45.03,46.27)
SST-2	85.49 (85.03, 85.90)	84.84 (84.34,85.20)
Subj	93.20 (92.97, 93.45)	92.84 (92.56,93.06)
TREC	91.54 (91.15, 91.92)	90.66 (90.02, 91.18)
CR	83.92 (82.95, 84.56)	83.57 (82.78, 84.28)
MPQA	89.32 (88.84, 89.73)	89.57 (89.18, 89.85)

Table 4: Performance on several datasets using non-static and static word2vec-CNN. In each cell we report: mean (min, max); we will use this format for all tables involving replications.

5.2 Effect of input word vectors

A nice property of sentence classification models that start with distributed representations of words as inputs is the flexibility the architecture affords to swap in different pre-trained word vectors. Therefore, we first explore the sensitivity of CNNs for sentence classification with respect to the input representations used. In particular, we replace Google word2vec with GloVe representations. Google word2vec uses a local context window model trained on 100 billion words from Google News (Mikolov et al., 2013), while GloVe proposes a model which leverages global word-word co-occurrence statistics in a certain very large corpus (Pennington et al., 2014). In this paper, we use a GloVe version which is trained from a corpus containing 840 billion tokens of web data and also has 300 dimensions. We keep all other settings the same as in the original configuration. We report results in Table 5. (Note that we also report results for SVM augmented with average GloVe vectors in Table 2.)

Dataset	Non-static GloVe-CNN	Static GloVe-CNN
MR	81.03 (80.68,81.48)	80.10 (79.55,80.51)
SST-1	45.65 (45.09,45.94)	44.76 (44.09,45.09)
SST-2	85.22 (85.04,85.48)	84.15 (83.94,84.33)
Subj	93.64 (93.51,93.77)	93.44 (93.28,93.60)
TREC	90.38 (90.19,90.59)	89.68 (89.26,90.05)
CR	84.33 (84.00,84.67)	83.50 (82.84,83.98)
MPQA	89.57 (89.31,89.78)	89.17 (88.98,89.46)

Table 5: Performance of GloVe-CNN

As a potentially simple means of realizing the best performance across all datasets, we also considered an approach that capitalizes jointly on both of these pre-trained representations. Specifically, we concatenated word2vec and GloVe vectors for each word, resulting in 600-dimensional word vectors, and we used these as input for the CNN. Pre-trained vectors may not always be available for specific words (either in word2vec or GloVe, or both); in such cases, we randomly initialized the corresponding sub-vectors, as described above.

Results are shown Table 6. Here we report results only for the non-static variant, given its general superiority.

Dataset	Non-static GloVe+Word2vec CNN
MR	81.02 (80.75,81.32)
SST-1	45.98 (45.49,46.65)
SST-2	85.45 (85.03,85.82)
Subj	93.66 (93.39,93.87)
TREC	91.37 (91.13,91.62)
CR	84.65 (84.21,84.96)
MPQA	89.55 (89.22,89.88)

Table 6: Performance of non-static GloVe + word2vec CNN

From these results, one can see that the relative performance when using GloVe versus word2vec depends on the dataset, and, unfortunately, that simply concatenating these representations is not necessarily helpful. Practically, when faced with a new dataset, it is probably worth experimenting with different pre-trained word vectors using the training data.

We also experimented with using long, sparse one-hot vectors as input word representations, in the spirit of (Johnson and Zhang, 2014). In this strategy, each word is encoded as a one-hot vector, which is a sparse high dimensional vector . In this case, the width of the sentence matrix is equal to the vocabulary size. The one-hot vector is fixed during the training, since this method acts like it searches each word in a pre-built dictionary. The performance is shown in Table 7.

Comparing the result with the ones with word2vec and GloVe, , we can see that under the same basic CNN configuration, one-hot vector uniformly performs worse than word2vec or

Dataset	One-hot vector CNN
MR	77.83 (76.56,78.45)
SST-1	41.96 (40.29,43.46)
SST-2	79.80 (78.53,80.52)
Subj	91.14 (90.38,91.53)
TREC	88.28 (87.34,89.30)
CR	78.22 (76.67,80.00)
MPQA	83.94 (82.94,84.31)

Table 7: Performance of one-hot vector CNN

GloVe. We do not exclude the possibility that with specific configurations, one-hot CNN may be able to outperform other input representations for sentence classification. But our evidence here suggests that one-hot CNN may not be suitable for sentence classification. This may be due to sparsity; the sentences are perhaps too brief to provide enough information for this high-dimensional encoding (whereas this may be less of a problem for longer documents).

5.3 Effect of filter region size

We first explore the effect of filter region size when using only one region size, and we set the number of feature maps for this region size as 100 (as in the original configuration). We consider region sizes of 1,3,5,7,10,15,20,25, and 30, and record the mean, minimum, and maximum accuracies over 10 replications of 10-fold CV for each region size, and show the performance in Table 8.

Figure 3 shows the difference between mean accuracy over 10 replications of each region size and the region size 3. Because we are only interested in the trend of the accuracy as we alter the region size or other components of the CNN (and not the absolute performance on each task), we show only the change in accuracy from an arbitrary baseline point (here, a region-size of 3). We follow this convention for all figures in this paper to ease interpretation.

From the figure, we can conclude that each dataset has its own optimal range of filter region size. Practically, this suggests performing a coarse grid search over specified range; the data here suggests that a reasonable range for sentence classification might be from 2 to 25. However, for datasets comprising longer sentences, such as CR (maximum sentence length is 105), the optimal region size might be even larger. This might also due to the fact that in CR, it's easier to predict the positive/negative customer reviews given a larger context.

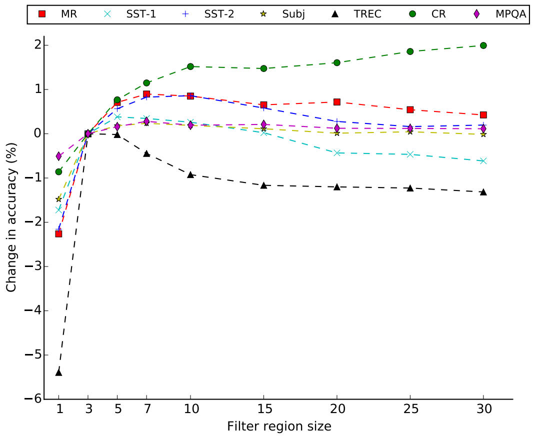

Figure 3: Effect of region size using only one region size in non-static word2vec-CNN

We also explore the effect of combining several different filter region sizes, while keeping the number of feature maps for each region size fixed at 100. Here we find that combining several filters with region size close to the optimal single region size can improve the performance, but adding region sizes far outside the optimal range may hurt performance. For example, from Figure 3, one can observe that the optimal single region size for the MR dataset is 7. We therefore combine several different filter region sizes close to this optimal range, and compare this to approaches that use region sizes outside of this range. From Table 9, we can see that using (5,6,7),and (7,8,9) and (6,7,8,9)– the sets near the best single region size – produce the best results. The difference is especially pronounced when comparing to the baseline setting of (3,4,5). Note that even only using a single good filter region size 7 results in better performance than combining different sizes (3,4,5). The best performing strategy here is to simply use many feature maps (here, 400) all with region size equal to 7, i.e., the single best region size.

We then give another empirical result using several region sizes on TREC dataset in Table 10. From Fig 3, we see that the best single filter region size for TREC is 3 and 5, so we explore the region size around them, and compare with the multiple region sizes far away from them. From TREC result, we see that (3,3,3) and (3,3,3,3) are worse than (2,3,4) and (3,4,5). However, the result still shows that combination of the region sizes near the optimal single region size can perform better than the region sizes far away from the optimal single

Region size	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
1	77.85 (77.47,77.97)	44.91 (44.42,45.37)	82.59(82.20,82.80)	91.23 (90.96,91.48)	85.82 (85.41,86.12)	80.15 (79.27,80.89)	88.53 (88.29,88.86)
3	80.48 (80.26,80.65)	46.64 (46.21,47.07)	84.74 (84.47,85.00)	92.71 (92.52,92.93)	91.21 (90.88,91.52)	81.01 (80.64,81.53)	89.04 (88.71,89.27)
5	81.13 (80.96,81.32)	47.02 (46.74,47.40)	85.31 (85.04,85.71)	92.89 (92.64,93.07)	91.20 (90.96,91.43)	81.78 (80.75,82.52)	89.20 (88.99,89.37)
7	81.65 (81.45,81.85)	46.98 (46.70,47.37)	85.57 (85.16,85.90)	92.95 (92.72,93.19)	90.77 (90.53,91.15)	82.16 (81.70,82.87)	89.32 (89.17,89.41)
10	81.43 (81.28,81.75)	46.90 (46.50,47.56)	85.60 (85.33,85.90)	92.90 (92.71,93.10)	90.29 (89.89,90.52)	82.53 (81.58,82.92)	89.23 (89.03,89.52)
15	81.26 (81.01,81.43)	46.66 (46.13,47.23)	85.33 (84.96,85.74)	92.82 (92.61,92.98)	90.05 (89.68,90.28)	82.49 (81.61,83.06)	89.25 (89.03,89.44)
20	81.06 (80.87,81.30)	46.20 (45.40,46.72)	85.02 (84.94,85.24)	92.72(92.47,92.87)	90.01 (89.84,90.50)	82.62 (82.16,83.03)	89.16 (88.92,89.28)
25	80.91 (80.73,81.10)	46.17 (45.20,46.92)	84.91 (84.49,85.39)	92.75 (92.45,92.96)	89.99 (89.66,90.40)	82.87 (82.21,83.45)	89.16 (88.99,89.45)
30	80.91 (80.72,81.05)	46.02 (45.21,46.54)	84.94 (84.63,85.25)	92.70 (92.50,92.90)	89.90 (89.58,90.13)	83.01 (82.44,83.38)	89.15 (88.93,89.41)

Table 8: Effect of single filter region size using non-static CNN.

Multiple region size	Accuracy (%)
(7)	81.65 (81.45,81.85)
(3,4,5)	81.24 (80.69, 81.56)
(4,5,6)	81.28 (81.07,81.56)
(5,6,7)	81.57 (81.31,81.80)
(7,8,9)	81.69 (81.27,81.93)
(10,11,12)	81.52 (81.27,81.87)
(11,12,13)	81.53 (81.35,81.76)
(3,4,5,6)	81.43 (81.10,81.61)
(6,7,8,9)	81.62 (81.38,81.72)
(7,7,7)	81.63 (81.33,82.08)
(7,7,7,7)	**81.73 (81.33,81.94)**

Table 9: Effect of filter region size with several region sizes using non-static word2vec-CNN on MR dataset

Multiple region size	Accuracy (%)
(3)	91.21 (90.88,91.52)
(5)	91.20 (90.96,91.43)
(2,3,4)	91.48 (90.96,91.70)
(3,4,5)	91.56 (91.24,91.81)
(4,5,6)	91.48 (91.17,91.68)
(7,8,9)	90.79 (90.57,91.26)
(14,15,16)	90.23 (89.81,90.51)
(2,3,4,5)	**91.57 (91.25,91.94)**
(3,3,3)	91.42 (91.11,91.65)
(3,3,3,3)	91.32 (90.53,91.55)

Table 10: Effect of filter region size with several region sizes using non-static word2vec-CNN on TREC dataset

region size, and a single good region size (3) outperforms combination of several bad region sizes (7,8,9) and (14,15,16).

In light of these observations, we believe it advisable to first perform a coarse line-search over a single filter region size to find the 'best' size for the dataset under consideration, and then explore the combining filters with region sizes nearby this single best size.

5.4 Effect of number of feature maps for each filter region size

We explore the effect of number of feature maps for each filter region size. Again, we keep other configurations the same, where there are three filter region sizes: 3, 4 and 5, and only the number of feature maps for each of these region size is changed. We keep the number of feature maps for 3, 4, and 5 the same. The results for non-static CNN are shown in Table 11. The change in accuracy from the baseline 100 is shown in Fig 4.

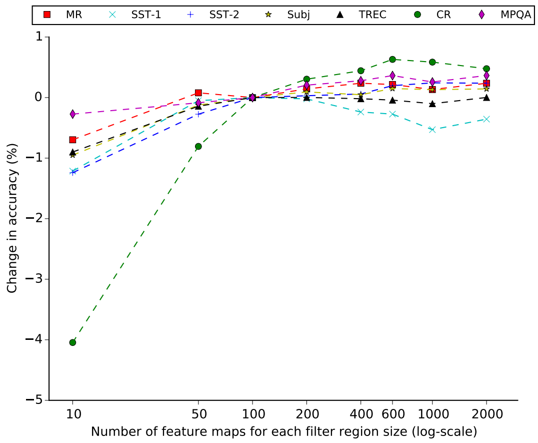

Figure 4: Effect of number of feature maps for each filter region size using non-static word2vec-CNN

One can see that the 'best' number of feature maps for each filter region size depends on the dataset. However, as a practical observation, it would seem that increasing the number of maps beyond 600 yields at best very marginal returns, and often hurts performs (likely due to overfitting). Another point to notice is that it takes a longer time to train the model when the number of feature maps is increased. We provide results pertaining to running time as a function of the number of feature maps in the Appendix. In practice, the evidence here suggests perhaps searching over a range of 50 to 600.

5.5 Effect of activation function

We explore the effect of seven different activation functions in the convolution layer, including: ReLU (as per the original setting), hyperbolic tangent (tanh), Sigmoid function (Maas et al., 2013), SoftPlus function (Dugas et al., 2001), Cube func-

	10	**50**	**100**	**200**	**400**	**600**	**1000**	**2000**
MR	80.47 (80.14,80.99)	81.25 (80.90,81.56)	81.17 (81.00,81.38)	81.31 (81.00,81.60)	81.41 (81.21,81.61)	81.38 (81.09, 81.68)	81.30 (81.15,81.39)	81.40 (81.13,81.61)
SST-1	45.90 (45.14,46.41)	47.06 (46.58,47.59)	47.09 (46.50,47.66)	47.09 (46.34,47.50)	46.87 (46.41,47.43)	46.84 (46.29,47.47)	46.58 (46.26,47.14)	46.75 (45.87,47.67)
SST-2	84.26 (83.93,84.73)	85.23 (84.86,85.57)	85.50 (85.31,85.66)	85.53 (85.24,85.69)	85.56 (85.27,85.79)	85.70 (85.57,85.93)	85.75 (85.53,86.01)	85.74 (85.49,86.02)
Subj	92.24 (91.74,92.43)	93.07 (92.94,93.28)	93.19 (93.08,93.45)	93.29 (93.07,93.38)	93.24 (92.96,93.39)	93.34 (93.22,93.44)	93.32 (93.17,93.49)	93.34 (93.05,93.49)
TREC	90.64 (90.19,90.86)	91.40 (91.12,91.59)	91.54 (91.17,91.90)	91.54 (91.23,91.71)	91.52 (91.30,91.70)	91.50 (91.23,91.71)	91.44 (91.26,91.56)	91.54 (91.28,91.75)
CR	79.95 (79.36,80.41)	83.19 (82.32,83.50)	83.86 (83.52,84.15)	84.30 (83.80,84.64)	84.44 (84.14,85.02)	84.62 (84.31,84.94)	84.58 (84.35,84.85)	84.47 (83.84,85.03)
MPQA	89.02 (88.89,89.19)	89.21 (88.97,89.41)	89.21 (88.90,89.51)	89.50 (89.27,89.68)	89.57 (89.13,89.81)	89.66 (89.35,89.90)	89.55 (89.22,89.73)	89.66 (89.47,89.94)

Table 11: Performance of number of feature maps for each filter using non-static word2vec-CNN

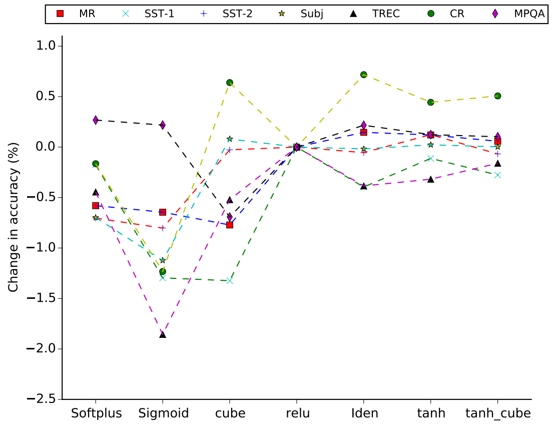

Figure 5: Effect of activation function in non-static word2vec-CNN

tion(Chen and Manning, 2014), and tanh cube function (Pei et al., 2015). We use 'Iden' to denote the identity function, which means not using any activation function. The effect of different activation functions in non-static CNN is reported in Table 12, and the change in accuracy from baseline 'ReLU' is shown in Fig 5.

From the figure, we can see that for 6 out of 7 datasets, the best activation function is one of Iden, ReLU and tanh. Softplus and Sigmoid functions outperform these only on one dataset (MPQA). Practically, we therefore suggest experimenting with each of Iden, ReLU and tanh.

5.6 Effect of pooling strategy

We investigate the effect of the pooling strategy and the pooling region size. We keep the filter region size the same as (3,4,5) as in the baseline configuration, and we use 100 feature maps for each region size, thus we change only the pooling strategy and the pooling region size.

In the original configuration, 1-max pooling was performed over entire feature maps, inducing a feature vector of length 1 for each filter. Alternatively, however, pooling may also be performed over small equal sized local regions rather than over the entire feature map (Boureau et al., 2011). Each small local region on the feature map will generate a single number from pooling, and all of these numbers can be concatenated together to form a feature vector for one feature map. The following step is the same as 1-max pooling: we concatenate all the feature vectors together to form a single feature vector for the classification layer. Figure 6 illustrates an example of max pooling performed over local regions (with size 3) on a feature map of length 9. This feature map will generate a feature vector of length 3. Here we use the same pooling strategy (max pooling or average pooling) for all of the feature maps. Because we have 100 feature maps for each of the three filter region sizes, if each feature map generates a feature vector of length n, then the final feature vector will have length $300 * n$ rather than 300, as in the case 1-max pooling. When the length of the feature map is not a multiple of the pooling region size, we simply take the maximum value from the rest of the feature map.

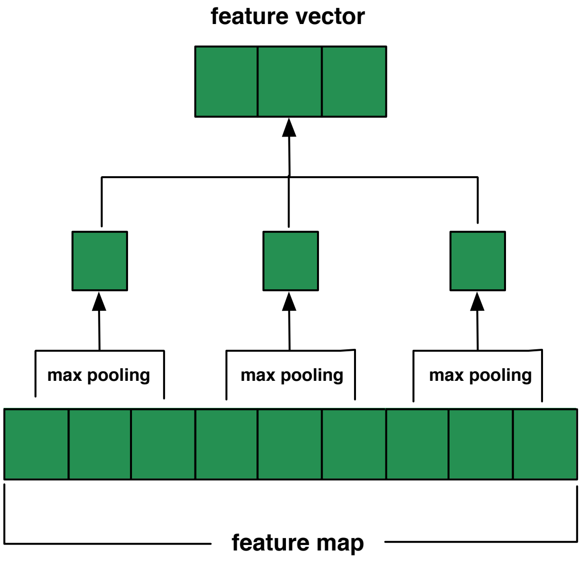

Figure 6: Local max pooling within 3 sized region

The results for non-static CNN are shown in Table 13. Here 'max,3' means that we perform max pooling within local region of size of 3 on the feature map for all of the filter region sizes. Interest-

	Sigmoid	**tanh**	**Softplus**	**Iden**	**ReLU**	**Cube**	**tahn-cube**
MR	80.51 (80.22, 80.77)	81.28 (81.07, 81.52)	80.58 (80.17, 81.12)	81.30 (81.09, 81.52)	81.16 (80.81, 83.38)	80.39 (79.94,80.83)	81.22 (80.93,81.48)
SST-1	45.83 (45.44, 46.31)	47.02 (46.31, 47.73)	46.95 (46.43, 47.45)	46.73 (46.24,47.18)	47.13 (46.39, 47.56)	45.80 (45.27,46.51)	46.85 (46.13,47.46)
SST-2	84.51 (84.36, 84.63)	85.43 (85.10, 85.85)	84.61 (84.19, 84.94)	85.26 (85.11, 85.45)	85.31 (85.93, 85.66)	85.28 (85.15,85.55)	85.24 (84.98,85.51)
Subj	92.00 (91.87, 92.22)	93.15 (92.93, 93.34)	92.43 (92.21, 92.61)	93.11 (92.92, 93.22)	93.13 (92.93, 93.23)	93.01 (93.21,93.43)	92.91 (93.13,93.29)
TREC	89.64 (89.38, 89.94)	91.18 (90.91, 91.47)	91.05 (90.82, 91.29)	91.11 (90.82, 91.34)	91.54 (91.17, 91.84)	90.98 (90.58,91.47)	91.34 (90.97,91.73)
CR	82.60 (81.77, 83.05)	84.28 (83.90, 85.11)	83.67 (83.16, 84.26)	84.55 (84.21, 84.69)	83.83 (83.18, 84.21)	84.16 (84.47,84.88)	83.89 (84.34,84.89)
MPQA	89.56 (89.43, 89.78)	89.48 (89.16, 89.84)	89.62 (89.45, 89.77)	89.57 (89.31, 89.88)	89.35 (88.88, 89.58)	88.66 (88.55,88.77)	89.45 (89.27,89.62)

Table 12: Performance of different activation functions using non-static word2vec-CNN

ingly, 1-max pooling ('max,all'; the last column) uniformly outperforms local max pooling.

We also consider a k-max pooling strategy similar to (Kalchbrenner et al., 2014), in which the maximum k values are extracted from the entire feature map, and the relative order of these values is preserved. We explore the effect of k on the performance, and show the results for non-static CNN in Table 14. Again we see 1-max pooling fares best, consistently outperforming k-max pooling.

Next, we replace the max operation in the local max pooling with the average operation (Boureau et al., 2010a), that is, we take the average value within a region size rather than the maximum value. The rest of the architecture is the same. We try local average pooling region size 3, 10, 20, and 30, and find that the average pooling uniformly performs (much) worse than max pooling, at least on the CR and TREC dataset as shown in Table 15.[8]

Pooling region	**CR**	**TREC**
3	81.01 (80.73,81.28)	88.89 (88.67,88.97)
10	80.74 (80.36,81.09)	88.10 (87.82,88.47)
20	80.69 (79.72,81.32)	86.45 (85.65,86.42)
30	81.13 (80.16,81.76)	84.95 (84.65,85.14)
all	80.17 (79.97,80.84)	83.30 (83.11,83.57)

Table 15: Performance of local average pooling region size using non-static word2vec-CNN ('all' means average pooling over the whole feature map resulting in one number)

The take-away from our analysis of pooling strategies is that 1-max pooling consistently performs better than alternative strategies for the task of sentence classification. This may be because the location of predictive contexts does not matter, and certain n-grams in the sentence can be more predictive on their own than the entire sentence considered jointly.

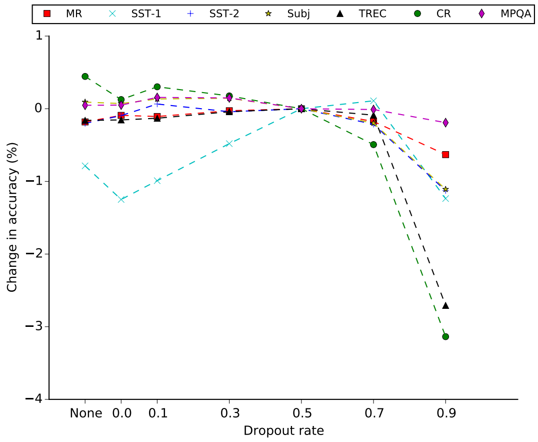

Figure 7: Effect of dropout rate in non-static word2vec-CNN

5.7 Effect of regularization

As mentioned above, 'dropout' is one means of regularization, and is applied to the input to the penultimate layer. We experiment with varying the dropout rate from 0.0 to 0.9, while fixing the $l2$ norm constraint to 3, as per the baseline configuration. The results for non-static CNN are shown in Table 16. We also report the accuracy achieved when we remove both dropout and the $l2$ norm constraint (i.e., when no regularization is performed); this is dentoed as 'None'. The change in accuracy from baseline 0.5 is shown in Fig 7.

Separately, we considered the effect of the $l2$ norm imposed on the weight vectors that parametrize the softmax function. Recall that the $l2$ norm of a weight vector is linearly scaled to a constraint c when it exceeds this threshold, so smaller c implies stronger regularization. (Note that this strategy also applies only to the penultimate layer like the dropout.) We show the relative effect of varying c on non-static CNN in Table 17 and Figure 8, where we have fixed the dropout rate to 0.5; 3 is the baseline here (again, arbitrarily).

From Figures 7 and 8, one can see that non-zero dropout rates can help (though very little) at some points from 0.1 to 0.5, depending on datasets. But the $l2$ norm constraint either does not help much,

[8] Due to the substantially worse performance and slow running time when using average versus max pooling, we did not repeat experiments on all datasets.

	max,3	max,10	max,20	max,30	max,all (1-max)
MR	79.75 (79.47,80.03)	80.20 (80.02,80.35)	80.68 (80.14,81.21)	80.99 (80.65,81.30)	81.28 (81.16,81.54)
SST-1	44.98 (44.06,45.68)	46.10(45.37,46.84)	46.75 (46.35,47.36)	47.02 (46.59,47.59)	47.00 (46.54,47.26)
SST-2	83.69(83.46,84.07)	84.63 (84.44,84.88)	85.18 (84.64,85.59)	85.38 (85.31,85.49)	85.50 (85.31,85.83)
Subj	92.60 (92.28,92.76)	92.87 (92.69,93.17)	93.06 (92.81,93.19)	93.13 (92.79,93.32)	93.20 (93.00,93.36)
TREC	90.29 (89.93,90.61)	91.42 (91.16,91.71)	91.52 (91.23,91.72)	91.47 (91.15,91.64)	91.56 (91.67,91.88)
CR	81.72 (81.21,82.20)	82.71 (82.06,83.30)	83.44(83.06,83.90)	83.70 (83.31,84.25)	83.93 (83.48,84.39)
MPQA	89.15 (88.83,89.47)	89.39 (89.14,89.56)	89.30 (89.16,89.60)	89.37 (88.99,89.61)	89.39 (89.04,89.73)

Table 13: Performance of local max pooling using non-static word2vec-CNN

	1 (1-max)	5	10	15	20
MR	81.25 (81.00,81.47)	80.83 (80.69,80.91)	80.05 (79.69,80.41)	80.11 (79.89,80.36)	80.05 (79.72,80.25)
SST-1	47.24 (46.90,47.65)	46.63 (46.31,47.12)	46.04 (45.27,46.61)	45.91 (45.16,46.49)	45.31 (44.90,45.63)
SST-2	85.53 (85.26,85.80)	84.61(84.47,84.90)	84.09 (83.94,84.22)	84.02 (83.57,84.28)	84.04 (83.74,84.34)
Subj	93.18 (93.09,93.31)	92.49 (92.33,92.61)	92.66 (92.50,92.79)	92.52 (92.33,92.96)	92.58 (92.50,92.83)
TREC	91.53 (91.26,91.78)	89.93 (89.75,90.09)	89.73 (89.61,89.83)	89.49(89.31,89.65)	89.05(88.85,89.34)
CR	83.81 (83.44,84.37)	82.70 (82.14,83.11)	82.46 (82.17,82.76)	82.26 (81.86, 82.90)	82.09 (81.74,82.34)
MPQA	89.39 (89.14, 89.58)	89.36 (89.17,89.57)	89.14 (89.00,89.45)	89.31 (89.18,89.48)	88.93 (88.82,89.06)

Table 14: Performance of global k-max pooling using non-static word2vec-CNN

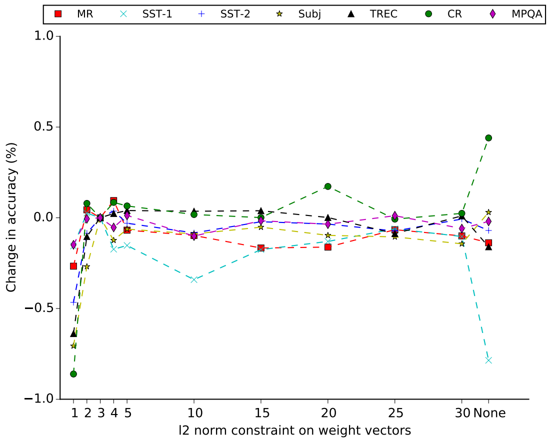

Figure 8: Effect of $l2$ norm constraint on weight vectors in non-static word2vec-CNN

and even adversely effects performance on at least one dataset (CR). This observation is in contrast to results reported for image and speech tasks (Srivastava et al., 2014), and also in contrast to (Peng et al., 2015), which uses recursive neural network (RNN) for sentiment analysis and applies regularization in all layers. We would suggest not using strong regularization: perhaps using a dropout rate ranging from 0 to 0.5, and $l2$ norm constraint no less than 3. Further, it is probably worth exploring dropping reuglarization entirely.

6 Conclusions

6.1 Summary of Main Empirical Findings

From our experimental analysis we draw several conclusions that we hope will guide future work and be useful for researchers new to using CNNs for sentence classification.

- Prior work has tended to report only the mean performance on datasets achieved by models. But this overlooks variance due solely to the stochastic inference procedure used. This can be substantial: holding everything constant (including the folds), so that variance is due exclusively to the stochastic inference procedure, we find that mean performance (calculated via 10 fold cross-validation) has a range of up to 1.5 points (Table 4). More replication should be performed in future work, and ranges/variances should be reported, to prevent potentially spurious conclusions regarding relative model performance.

- Surprisingly (in our view) regularization – i.e., dropout and $l2$ constraints on parameter weights – seems to have little effect on model performance on some datasets.

- We find that, even when tuning them to the task at hand, the choice of input word vector representation (e.g., between word2vec and GloVe) has an impact on performance, however different representations perform better for different tasks. At least for sentence classification, both seem to perform better than using one-hot vectors directly.

- The filter region size can have a large effect on performance, and should be tuned on a per-task basis.

	None	0.0	0.1	0.3	0.5	0.7	0.9
MR	81.15 (80.95,81.34)	81.24 (80.82, 81.63)	81.22 (80.97 ,81.61)	81.30 (81.03 ,81.48)	81.33 (81.02, 81.74)	81.16 (80.83, 81.57)	80.70 (80.36, 80.89)
SST-1	46.30 (45.81,47.09)	45.84 (45.13 ,46.43)	46.10 (45.68, 46.36)	46.61 (46.13, 47.04)	47.09 (46.32, 47.66)	47.19 (46.88 ,47.46)	45.85 (45.50, 46.42)
SST-2	85.42 (85.13,85.23)	85.53 (85.12 ,85.88)	85.69 (85.32, 86.06)	85.58 (85.30, 85.76)	85.62 (85.25, 85.92)	85.41 (85.18, 85.65)	84.49 (84.35, 84.82)
Subj	93.23 (93.09,93.37)	93.21 (93.09 ,93.31)	93.27 (93.12 ,93.45)	93.28 (93.06, 93.39)	93.14 (93.01, 93.32)	92.94 (92.77 ,93.08)	92.03 (91.80 ,92.24)
TREC	91.38 (91.18,91.59)	91.39 (91.13 ,91.66)	91.41 (91.26, 91.63)	91.50 (91.22 ,91.76)	91.54 (91.41, 91.68)	91.45 (91.17, 91.77)	88.83 (88.53 ,89.19)
CR	84.36 (84.06,84.70)	84.04 (82.91, 84.84)	84.22 (83.47, 84.60)	84.09 (83.72, 84.51)	83.92 (83.12, 84.34)	83.42 (82.87, 83.97)	80.78 (80.35, 81.34)
MPQA	89.30 (88.91,89.68)	89.30 (89.01, 89.56)	89.41 (89.19, 89.64)	89.40 (89.18, 89.77)	89.25 (88.96, 89.60)	89.24 (88.98, 89.50)	89.06 (88.93, 89.26)

Table 16: Effect of dropout rate using non-static word2vec-CNN

	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
1	81.02 (80.75 ,81.29)	46.93 (46.57, 47.33)	85.02 (84.76,85.22)	92.49 (92.35 92.63)	90.90 (90.62 91.20)	83.06 (82.50 83.42)	89.17 (88.97 89.36)
2	81.33 (81.04 ,81.71)	47.11 (46.77, 47.43)	85.40 (84.98,85.67)	92.93 (92.82 93.15)	91.44 (91.20 91.60)	84.00 (83.57 84.34)	89.31 (89.17 89.54)
3	81.29 (80.96, 81.59)	47.29 (46.90 ,47.82)	85.47 (85.17,85.77)	93.21 (93.03 93.37)	91.44 (91.18 91.68)	83.89 (83.24 84.47)	89.18 (88.84 89.40)
4	81.38 (81.21, 81.68)	46.91 (46.22 ,47.38)	85.33 (85.25,85.72)	93.08 (92.96 93.22)	91.56 (91.26 91.90)	84.00 (83.21 84.60)	89.27 (89.11 89.41)
5	81.22 (81.03, 81.49)	46.93 (46.44 ,47.38)	85.46 (84.98,85.73)	93.14 (92.90 93.33)	91.58 (91.39 91.87)	83.99 (83.73 84.31)	89.33 (89.02 89.55)
10	81.19 (80.94 ,81.42)	46.74 (46.19, 47.12)	85.41 (85.04,85.83)	93.11 (92.99 93.32)	91.58 (91.29 91.81)	83.94 (83.04 84.61)	89.22 (89.01 89.40)
15	81.12 (80.87, 81.29)	46.91 (46.58 ,47.48)	85.47 (85.23,85.74)	93.15 (92.99 93.29)	91.58 (91.37 91.84)	83.92 (83.40 84.54)	89.30 (88.93 89.66)
20	81.13 (80.64, 81.33)	46.96 (46.62 ,47.31)	85.46 (85.17,85.64)	93.10 (92.98 93.19)	91.54 (91.28 91.73)	84.09 (83.59 84.53)	89.28 (88.92 89.43)
25	81.22 (80.82, 81.66)	47.02 (46.73, 47.67)	85.42 (85.16,85.78)	93.09 (92.95 93.25)	91.45 (91.22 91.62)	83.91 (83.24 84.40)	89.33 (89.05 89.61)
30	81.19 (80.79 ,81.43)	46.98 (46.63 ,47.59)	85.48 (85.27,85.79)	93.06 (92.84 93.43)	91.55 (91.26 91.84)	83.94 (83.02 84.35)	89.26 (89.10 89.54)
None	80.19(79.95,80.39)	46.30 (45.81,47.09)	85.42 (85.13,85.23)	93.23 (93.09,93.37)	91.38 (91.18,91.59)	84.36 (84.06,84.70)	89.30 (88.91,89.68)

Table 17: Effect of constraint on $l2$ norm using non-static word2vec-CNN

- The number of features can also play an important role in the performance, however, trying to find a good number might be time consuming.

- 1-max pooling uniformly beats other pooling strategies.

6.2 Advice to practitioners

Drawing from our empirical results, we provide the following guidance regarding CNN architecture and hyper-parameters for practitioners looking to apply CNNs to a new sentence classification task.

- Consider starting with the basic configuration described in Table 3 and use the non-static word2vec or GloVe rather than one-hot vector CNN.

- Line-search over the single filter region size to find the 'best' single region size. A reasonable range might be 2∼10. However, for datasets with very long sentences like CR, it may be worth exploring larger filter region sizes. Once this 'best' region size is identified, it may be worth exploring combining multiple filters using regions sizes near this single best size, given that empirically multiple 'good' region sizes always outperformed using only the single best region size.

- Alter the number of feature maps for each filter region size from 50 to 600. Note that increasing the number of feature maps will increase the running time, so there is a trade-off to consider.

- Consider different activation functions if possible: ReLU and tanh are the best overall candidates. And it might also be worth trying no activation function at all.

- Use 1-max pooling; it does not seem necessary to expend resources evaluating alternative strategies.

- Regarding regularization: If using regularization, perform a line search over the dropout rate parameter from 0.0 to 0.5. Using an $l2$ norm constraint is probably not necessary; but if one is to be used, it should be small; it may also be worth trying no regularization at all.

- When summarizing the performance of a model (or a particular configuration thereof), it is imperative to consider variance. Therefore, replications of the cross-fold validation procedure should be performed and variances and ranges should be reported.

Of course, the above suggestions are applicable only to datasets comprising sentences with similar properties to the seven considered in this work. And there may be examples that run counter to our findings here. Nonetheless, we believe these suggestions are likely to provide a reasonable starting point for researchers or practitioners looking to apply a simple one-layer CNNs to a new sentence classification task.

References

[Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

[Bengio2009] Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127.

[Boureau et al.2010a] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. 2010a. Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2559–2566. IEEE.

[Boureau et al.2010b] Y-Lan Boureau, Jean Ponce, and Yann LeCun. 2010b. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 111–118.

[Boureau et al.2011] Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. 2011. Ask the locals: multi-way local pooling for image recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2651–2658. IEEE.

[Breuel2015] Thomas M Breuel. 2015. The effects of hyperparameters on sgd training of neural networks. *arXiv preprint arXiv:1508.02788*.

[Chen and Manning2014] Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750.

[Coates et al.2011] Adam Coates, Andrew Y Ng, and Honglak Lee. 2011. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223.

[Collobert and Weston2008] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

[Dugas et al.2001] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. 2001. Incorporating second-order functional knowledge for better option pricing. *Advances in Neural Information Processing Systems*, pages 472–478.

[Goldberg2015] Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

[Hinton et al.2012] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[Hu and Liu2004] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

[Iyyer et al.2015] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification.

[Joachims1998] Thorsten Joachims. 1998. *Text categorization with support vector machines: Learning with many relevant features*. Springer.

[Johnson and Zhang2014] Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

[Kalchbrenner et al.2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[Lai et al.2015] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[LeCun et al.2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

[Li and Roth2002] Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

[Maas et al.2013] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Pang and Lee2005] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

[Pei et al.2015] Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proc. of ACL*.

[Peng et al.2015] Hao Peng, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2015. A comparative study on regularization strategies for embedding-based neural networks. *arXiv preprint arXiv:1508.03721*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

[Socher et al.2013] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Wang et al.2015] Peng Wang, Jiaming Xu, Bo Xu, Chenglin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. 2015. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 352–357, Beijing, China, July. Association for Computational Linguistics.

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

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Appendix

Performance of logistic regression In Table 18, we additionally report results achieved using logistic regression (regularized via an $l2$ norm penalty on the coefficients, weighted proportionally to a hyper-parameter that we again tuned) using the same feature sets.

Dataset	bow-LG	wv-LG	bow+wv-LG
MR	78.24	77.65	79.68
SST-1	40.91	43.60	43.09
SST-2	81.06	81.30	83.23
Subj	89.00	90.88	91.84
TREC	87.93	77.42	89.23
CR	77.59	80.79	80.39
MPQA	83.60	88.30	89.14

Table 18: Performance of logistic regression; feature sets are as described above.

Effect of single filter region size The result of single filter region size using static CNN is in Table 19.

Effect of number of feature maps The effect of number of feature maps for each filter region size using static CNN is shown in Table 20.

In Table 21, we give the average running time of one 10-fold CV in sequential on TREC dataset as we increase the feature maps. [9].

Number of feature maps	Time (min)
10	8.25
50	9.92
100	13.73
200	19.49
400	29.46
600	37.02
1000	56.97
2000	105.88

Table 21: Running time of one 10-fold CV on TREC dataset

Effect of activation function. The effect of activation using static CNN is shown in Table 22.

Effect of pooling The result of pooling strategy using static CNN is shown in Table 23 and Table 24.

Effect of dropout rate The effect of dropout rate using static CNN is shown in Table 25.

Effect of $l2$ norm constraint on weight vector The effect of $l2$ norm constraint using static CNN is shown in Table 26.

[9] If people can parallelize the 10-fold CV, the running time will be greatly decreased.

	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
1	79.22 (79.02,79.57)	45.46 (44.88,45.96)	83.24 (82.93,83.67)	91.97 (91.64,92.17)	85.86 (85.54,86.13)	80.24 (79.64,80.62)	88.25 (88.04,88.63)
3	80.27 (79.94,80.51)	46.18 (45.74,46.52)	84.37 (83.96,94.70)	92.83 (92.58,93.06)	90.33 (90.05,90.62)	80.71 (79.72,81.37)	89.37 (89.25,89.67)
5	80.35 (80.05,80.65)	46.18 (45.69,46.63)	84.38 (84.04,84.61)	92.54 (92.44,92.68)	90.06 (89.84,90.26)	81.11 (80.54,81.55)	89.50 (89.33,89.65)
7	80.25 (79.89,80.60)	45.96 (45.44,46.55)	84.24 (83.40,84.59)	92.50 (92.33,92.68)	89.44 (89.07,89.84)	81.53 (81.09,82.05)	89.44 (89.26,89.68)
10	80.02 (79.68,80.17)	45.65 (45.08,46.09)	83.90 (83.40,84.37)	92.31 (92.19,92.50)	88.81(88.53,89.03)	81.19 (80.89,81.61)	89.26 (88.96,89.60)
15	79.59 (79.36,79.75)	45.19 (44.67,45.62)	83.64 (83.32,83.95)	92.02 (91.86,92.23)	88.41 (87.96,88.71)	81.36 (80.72,82.04)	89.27 (89.04,89.49)
20	79.33 (78.76,79.75)	45.02 (44.15,45.77)	83.30 (83.03,83.60)	91.87 (91.70,91.99)	88.46 (88.21,88.85)	81.42 (81.03,81.90)	89.28 (88.90, 89.42)
25	79.05 (78.91,79.21)	44.61 (44.05,45.53)	83.24 (82.82,83.70)	91.95 (91.59,92.16)	88.23 (87.57,88.56)	81.16 (80.69,81.57)	89.24 (88.87,89.42)
30	79.04 (78.86,79.30)	44.66 (44.42,44.91)	83.09 (82.61,83.42)	91.85 (91.74,92.00)	88.41 (87.98,88.67)	81.28 (80.96,81.55)	89.13 (88.91,89.33)

Table 19: Effect of single filter region size using static CNN.

	10	50	100	200	400	600	1000	2000
MR	79.38 (78.88, 79.82)	80.49 (80.16, 80.87)	80.60 (80.27,80.85)	80.76 (80.48,81.00)	80.80 (80.56,81.11)	80.79 (80.68,80.86)	80.90 (80.67,81.16)	80.84 (80.38,81.27)
SST-1	45.62 (45.28,46.01)	46.33 (46.00,46.69)	46.21 (45.68,46.85)	46.23 (45.70, 46.99)	46.10 (45.71,46.59)	46.20 (45.85,46.55)	46.56 (46.26,46.92)	45.93 (45.57,46.27)
SST-2	83.38 (82.65,83.68)	84.71 (84.46,85.27)	84.89 (84.56,85.16)	84.92 (84.81,85.18)	84.98 (84.66,85.18)	84.99 (84.29,85.44)	84.90 (84.66,85.05)	84.97 (84.79,85.14)
Subj	91.84 (91.30,92.02)	92.75 (92.61,92.88)	92.89 (92.66,93.06)	92.88 (92.75,92.97)	92.91 (92.75,93.01)	92.88 (92.75,93.03)	92.89 (92.74,93.05)	92.89 (92.64,93.11)
TREC	89.02 (88.62,89.31)	90.51 (90.26, 90.82)	90.62 (90.09,90.82)	90.73 (90.48,90.99)	90.72 (90.43,90.89)	90.70 (90.51,91.03)	90.71 (90.46,90.94)	90.70 (90.53,90.87)
CR	79.40 (78.76,80.03)	82.57 (82.05,83.31)	83.48 (82.99,84.06)	83.83 (83.51,84.26)	83.95 (83.36,84.60)	83.96 (83.49, 84.47)	83.95 (83.40,84.44)	83.81 (83.30,84.28)
MPQA	89.28 (89.04,89.45)	89.53 (89.31,89.72)	89.55 (89.18,89.81)	89.73 (89.62,89.85)	89.80 (89.65,89.96)	89.84 (89.74,90.02)	89.72 (89.57,89.88)	89.82 (89.52,89.97)

Table 20: Effect of number of feature maps for each filter using static word2vec-CNN

	Sigmoid	tanh	Softplus	Iden	ReLU
MR	79.23 (79.11, 79.36)	80.73 (80.29, 81.04)	80.05 (79.76, 80.37)	80.63 (80.26, 81.04)	80.65 (80.44, 81.00)
TREC	85.81 (85.65, 85.99)	90.25 (89.92, 90.44)	89.50 (89.36, 89.97)	90.36 (90.23, 90.45)	90.23 (89.85, 90.63)
CR	81.14 (80.57, 82.01)	83.51 (82.91,83.95)	83.28 (82.67, 83.88)	83.82 (83.50, 84.15)	83.51 (82.54, 83.85)
SST-1	45.25 (44.65, 45.86)	45.98 (45.68, 46.44)	46.76 (46.41, 47.45)	46.01 (45.60, 46.32)	46.25 (45.70, 46.98)
SST-2	83.07 (82.48, 83.54)	84.65 (84.36, 85.00)	84.01 (83.57, 84.40)	84.71 (84.40, 85.07)	84.70 (84.31, 85.20)
Subj	91.56 (91.39, 91.71)	92.75 (92.60, 92.95)	92.20 (92.08, 92.32)	92.71 (92.51, 92.89)	92.83 (92.67, 92.95)
MPQA	89.43 (89.27, 89.56)	89.75 (89.64, 89.86)	89.45 (89.30, 89.56)	89.75 (89.56, 89.87)	89.66 (89.44, 90.00)

Table 22: Performance of different activation function using static word2vec-CNN

	max,3	max,10	max,20	max,30	max,all (1-max)
MR	78.03 (77.69 78.54)	79.50 (79.20 79.83)	79.89 (79.46 80.10)	80.44 (80.06 80.85)	80.62 (80.24 81.17)
SST-1	43.84 (43.26 44.35)	45.04 (44.60 45.93)	45.74 (45.31 46.16)	45.87 (45.15 46.53)	46.06 (45.62,46.51)
SST-2	82.39 (82.00 82.78)	83.69 (83.42 83.95)	84.37 (84.17 84.67)	84.65 (84.47 84.95)	84.83 (84.56,85.16)
Subj	92.22 (92.07 92.40)	92.62 (92.37 92.76)	92.74 (92.55 93.03)	92.81 (92.60 93.10)	92.85 (92.56,93.06)
TREC	89.47 (89.27 89.83)	90.59 (90.43 90.82)	90.60 (90.27 90.93)	90.70 (90.47 91.01)	90.73 (90.41,91.11)
CR	80.77 (80.40 81.16)	82.17 (81.25 82.67)	83.08 (82.55 83.62)	83.46 (83.04 83.82)	83.53 (83.06,83.82)
MPQA	89.09 (88.80 89.28)	89.58 (89.40 89.77)	89.52 (89.31 89.71)	89.58 (89.27 89.76)	89.60 (89.46,89.76)

Table 23: Performance of local max pooling using static word2vec-CNN

	1 (1-max)	5	10	15	20
MR	80.66 (80.27,81.13)	80.44 (80.23 80.73)	79.89 (79.71 80.16)	79.63 (79.41 79.83)	79.60 (78.90 80.25)
SST-1	46.47 (45.96,46.81)	45.78 (45.53 46.15)	45.33 (44.85 45.72)	44.96 (44.60 45.49)	44.23 (43.72 44.83)
SST-2	84.85 (84.56,85.18)	84.13 (83.87 84.37)	83.53 (82.98 83.97)	83.11 (82.83 83.27)	82.71 (82.59 83.03)
TREC	90.68 (90.34,91.18)	89.67 (89.52 89.92)	89.37 (89.17 89.56)	89.04 (88.83 89.18)	87.85 (87.69 88.12)
CR	83.44 (82.78,84.00)	83.35 (82.65 83.71)	82.65 (82.32 83.01)	82.37 (82.03 82.64)	82.26 (81.79 82.79)
MPQA	89.54 (89.36,89.81)	89.58 (89.36 89.82)	89.57 (89.41 89.77)	89.38 (89.19 89.60)	88.68 (88.36 88.93)

Table 24: Performance of global k-max pooling using static word2vec-CNN

	None	0.0	0.1	0.3	0.5	0.7	0.9
MR	80.19(79.95,80.39)	80.37 (80.03, 80.66)	80.54 (80.13, 80.90)	80.46 (80.20, 80.63)	80.66 (80.34, 81.10)	80.70 (80.31, 80.95)	79.88 (79.57, 80.06)
SST-1	45.11 (44.57,45.64)	45.40 (45.00 ,45.72)	45.08 (44.45, 45.70)	45.94 (45.55, 46.45)	46.41 (45.89, 46.92)	46.87 (46.60 ,47.24)	45.37 (45.18, 45.65)
SST-2	84.58 (84.24,84.87)	84.70 (84.34, 84.96)	84.63 (84.41 ,84.95)	84.80 (84.54, 84.99)	84.95 (84.52, 85.29)	84.82 (84.61 ,85.15)	83.66 (83.45, 83.89)
Subj	92.88 (92.58,93.03)	92.82 (92.57 ,93.14)	92.81 (92.71, 92.90)	92.89 (92.64, 93.05)	92.86 (92.77, 93.04)	92.71 (92.51 ,92.93)	91.60 (91.50, 91.79)
TREC	90.55 (90.26,90.94)	90.69 (90.36 ,90.93)	90.84 (90.67, 91.06)	90.75 (90.56, 90.95)	90.71 (90.46, 91.10)	89.99 (89.67,90.16)	85.32 (85.01, 85.57)
CR	83.53 (82.96,84.15)	83.46 (83.03 ,84.04)	83.60 (83.22 ,83.87)	83.63 (83.03, 84.08)	83.38 (82.70, 83.67)	83.32 (82.72 ,84.07)	80.67 (80.12, 81.01)
MPQA	89.51 (89.42,89.67)	89.36 (89.12 89.63)	89.52 (89.32 89.68)	89.55 (89.28 89.77)	89.53 (89.37 89.79)	89.52 (89.29 89.70)	88.91 (88.76 89.12)

Table 25: Effect of dropout rate using static word2vec-CNN

	MR	**SST-1**	**SST-2**	**Subj**	**TREC**	**CR**	**MPQA**
1	81.02 (80.75 ,81.29)	46.93 (46.57, 47.33)	85.02 (84.76,85.22)	92.49 (92.35 92.63)	90.90 (90.62 91.20)	83.06 (82.50 83.42)	89.17 (88.97 89.36)
2	81.33 (81.04 ,81.71)	47.11 (46.77, 47.43)	85.40 (84.98,85.67)	92.93 (92.82 93.15)	91.44 (91.20 91.60)	84.00 (83.57 84.34)	89.31 (89.17 89.54)
3	81.29 (80.96, 81.59)	47.29 (46.90 ,47.82)	85.47 (85.17,85.77)	93.21 (93.03 93.37)	91.44 (91.18 91.68)	83.89 (83.24 84.47)	89.18 (88.84 89.40)
4	81.38 (81.21, 81.68)	46.91 (46.22 ,47.38)	85.33 (85.25,85.72)	93.08 (92.96 93.22)	91.56 (91.26 91.90)	84.00 (83.21 84.60)	89.27 (89.11 89.41)
5	81.22 (81.03, 81.49)	46.93 (46.44 ,47.38)	85.46 (84.98,85.73)	93.14 (92.90 93.33)	91.58 (91.39 91.87)	83.99 (83.73 84.31)	89.33 (89.02 89.55)
10	81.19 (80.94 ,81.42)	46.74 (46.19, 47.12)	85.41 (85.04,85.83)	93.11 (92.99 93.32)	91.58 (91.29 91.81)	83.94 (83.04 84.61)	89.22 (89.01 89.40)
15	81.12 (80.87, 81.29)	46.91 (46.58 ,47.48)	85.47 (85.23,85.74)	93.15 (92.99 93.29)	91.58 (91.37 91.84)	83.92 (83.40 84.54)	89.30 (88.93 89.66)
20	81.13 (80.64, 81.33)	46.96 (46.62 ,47.31)	85.46 (85.17,85.64)	93.10 (92.98 93.19)	91.54 (91.28 91.73)	84.09 (83.59 84.53)	89.28 (88.92 89.43)
25	81.22 (80.82, 81.66)	47.02 (46.73, 47.67)	85.42 (85.16,85.78)	93.09 (92.95 93.25)	91.45 (91.22 91.62)	83.91 (83.24 84.40)	89.33 (89.05 89.61)
30	81.19 (80.79 ,81.43)	46.98 (46.63 ,47.59)	85.48 (85.27,85.79)	93.06 (92.84 93.43)	91.55 (91.26 91.84)	83.94 (83.02 84.35)	89.26 (89.10 89.54)
None	80.19(79.95,80.39)	45.11 (44.57,45.64)	84.58 (84.24,84.87)	92.88 (92.58,93.03)	90.55 (90.26,90.94)	83.53 (82.96,84.15)	89.51 (89.42,89.67)

Table 26: Effect of constraint on $l2$-norms using static word2vec-CNN