Non-Linear Text Regression with a Deep Convolutional Neural Network

Zsolt Bitvai
University of Sheffield, UK
z.bitvai@shef.ac.uk

Trevor Cohn
University of Melbourne, Australia
t.cohn@unimelb.edu.au

Abstract

Text regression has traditionally been tackled using linear models. Here we present a non-linear method based on a deep convolutional neural network. We show that despite having millions of parameters, this model can be trained on only a thousand documents, resulting in a 40% relative improvement over sparse linear models, the previous state of the art. Further, this method is flexible allowing for easy incorporation of side information such as document meta-data. Finally we present a novel technique for interpreting the effect of different text inputs on this complex non-linear model.

1 Introduction

Text regression involves predicting a real world phenomenon from textual inputs, and has been shown to be effective in domains including election results (Lampos et al., 2013), financial risk (Kogan et al., 2009) and public health (Lampos and Cristianini, 2010). Almost universally, the text regression problem has been framed as linear regression, with the modelling innovation focussed on effective regression, e.g., using Lasso penalties to promote feature sparsity (Tibshirani, 1996).[1] Despite their successes, linear models are limiting: text regression problems will often involve complex interactions between textual inputs, thus requiring a non-linear approach to properly capture such phenomena. For instance, in modelling movie revenue conjunctions of features are likely to be important, e.g., a movie described as 'scary' is likely to have different effects for children's versus adult movies. While these kinds of features can be captured using explicit feature engineering, this process is tedious, limited in scope (e.g., to conjunctions) and – as we show here – can be dramatically improved by representational learning as part of a non-linear model.

In this paper, we propose an artificial neural network (ANN) for modelling text regression. In language processing, ANNs were first proposed for probabilistic language modelling (Bengio et al., 2003), followed by models of sentences (Kalchbrenner et al., 2014) and parsing (Socher et al., 2013) *inter alia*. These approaches have shown strong results through automatic learning dense low-dimensional distributed representations for words and other linguistic units, which have been shown to encode important aspects of language syntax and semantics. In this paper we develop a convolutional neural network, inspired by their breakthrough results in image processing (Krizhevsky et al., 2012) and recent applications to language processing (Kalchbrenner et al., 2014; Kim, 2014). These works have mainly focused on 'big data' problems with plentiful training examples. Given their large numbers of parameters, often in the millions, one would expect that such models can only be effectively learned on very large datasets. However we show here that a complex deep convolution network can be trained on about a thousand training examples, although careful model design and regularisation is paramount.

We consider the problem of predicting the future box-office takings of movies based on reviews by movie critics and movie attributes. Our approach is based on the method and dataset of Joshi et al. (2010), who presented a linear regression model over uni-, bi-, and tri-gram term frequency counts extracted from reviews, as well as movie and reviewer metadata. This problem is especially interesting, as comparatively few instances are available for training (see Table 1) while each in-

[1] Some preliminary work has shown strong results for non-linear text regression using Gaussian Process models (Lampos et al., 2014), however this approach has not been shown to scale to high dimensional inputs.

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 180–185,
Beijing, China, July 26-31, 2015.

	train	dev	test	total
# movies	1147	317	254	1718
# reviews per movie	4.2	4.0	4.1	4.1
# sentences per movie	95	84	82	91
# words per movie	2879	2640	2605	2794

Table 1: Movie review dataset (Joshi et al., 2010).

stance (movie) includes a rich array of data including the text of several critic reviews from various review sites, as well as structured data (genre, rating, actors, etc.) Joshi et al. found that regression purely from movie meta-data gave strong predictive accuracy, while text had a weaker but complementary signal. Their best results were achieved by domain adaptation whereby text features were conjoined with a review site identifier. Inspired by Joshi et al. (2010) our model also operates over n-grams, $1 \leq n \leq 3$, and movie metadata, albeit using an ANN in place of their linear model. We use word embeddings to represent words in a low dimensional space, a convolutional network with max-pooling to represent documents in terms of n-grams, and several fully connected hidden layers to allow for learning of complex non-linear interactions. We show that including non-linearities in the model is crucial for accurate modelling, providing a relative error reduction of 40% (MAE) over their best linear model. Our final contribution is a novel means of model interpretation. Although it is notoriously difficult to interpret the parameters of an ANN, we show a simple method of quantifying the effect of text n-grams on the prediction output. This allows for identification of the most important textual inputs, and investigation of non-linear interactions between these words and phrases in different data instances.

2 Model

The outline of the convolutional network is shown in Figure 1. We have n training examples of the form $\{\mathbf{b}_i, \mathbf{r}_i, y_i\}_{i=1}^{n}$, where $\mathbf{b}_i$ is the meta data associated with movie i, y_i is the target gross weekend revenue, and $\mathbf{r}_i$ is a collection of u_i number of reviews, $\mathbf{r}_i = \{\mathbf{x}_j, t_j\}_{j=1}^{u_i}$ where each review has review text $\mathbf{x}_j$ and a site id t_j. We concatenate all the review texts $d_i = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_u)$ to form our text input (see part **I** of Figure 1).

To acquire a distributed representation of the text, we look up the input tokens in a pretrained word embedding matrix $\mathbf{E}$ with size $|V| \times e$, where

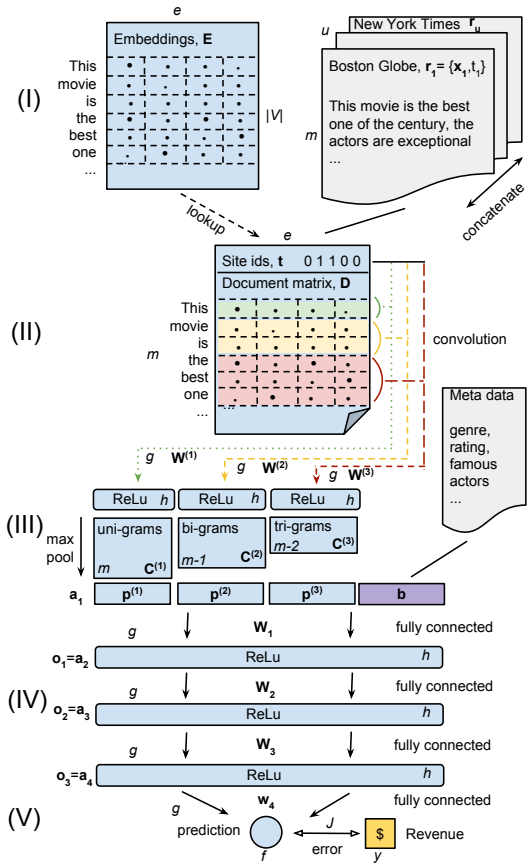

Figure 1: Outline of the network architecture.

$|V|$ is the size of our vocabulary and e is the embedding dimensionality. This gives us a dense document matrix $D_{i,j} = E_{d_i,j}$ with dimensions $m \times e$ where m is the number of tokens in the document.

Since the length of text documents vary, in part **II** we apply convolutions with width one, two and three over the document matrix to obtain a fixed length representation of the text. The n-gram convolutions help identify local context and map that to a new higher level feature space. For each feature map, the convolution takes adjacent word embeddings and performs a feed forward computation with shared weights over the convolution window. For a convolution with width $1 \leq q \leq m$ this is

$$\mathbf{S}_{i,\cdot}^{(q)} = (\mathbf{D}_{i,\cdot}, \mathbf{D}_{i+1,\cdot}, ..., \mathbf{D}_{i+q-1,\cdot})$$
$$\mathbf{C}_{i,\cdot}^{(q)} = \langle \mathbf{S}_{i,\cdot}^{(q)}, \mathbf{W}^{(q)} \rangle$$

where $\mathbf{S}_{i,\cdot}^{(q)}$ is q adjacent word embeddings concatenated, and $\mathbf{C}^{(q)}$ is the convolution output matrix with $(m-q+1)$ rows after a linear transformation with weights $\mathbf{W}^{(q)}$. To allow for a non-linear

transformation, we make use of rectified linear activation units, $\mathbf{H}^{(q)} = \max(\mathbf{C}^{(q)}, 0)$, which are universal function approximators. Finally, to compress the representation of text to a fixed dimensional vector while ensuring that important information is preserved and propagated throughout the network, we apply max pooling over time, i.e. the sequence of words, for each dimension, as shown in part **III**,

$$p_j^{(q)} = \max \mathbf{H}_{\cdot,j}^{(q)}$$

where $p_j^{(q)}$ is dimension j of the pooling layer for convolution q, and $\mathbf{p}$ is the concatenation of all pooling layers, $\mathbf{p} = (\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, ..., \mathbf{p}^{(q)})$.

Next, we perform a series of non-linear transformations to the document vector in order to progressively acquire higher level representations of the text and approximate a linear relationship in the final output prediction layer. Applying multiple hidden layers in succession can require exponentially less data than mapping through a single hidden layer (Bengio, 2009). Therefore, in part **IV**, we apply densely connected neural net layers of the form $\mathbf{o}_k = h(g(\mathbf{a}_k, \mathbf{W}_k))$ where $\mathbf{a}_k$ is the input and $\mathbf{o}_k = \mathbf{a}_{k+1}$ is the output vector for layer k, g is a linear transformation function $\langle \mathbf{a}_k, \mathbf{W}_k \rangle$, and h is the activation function, i.e. rectified linear seen above. $l = 3$ hidden layers are applied before the final regression layer to produce the output $f = g(\mathbf{o}_l, \mathbf{w}_{l+1})$ in part $\mathbf{V}$.

The mean absolute error is measured between the predictions $\mathbf{f}$ and the targets $\mathbf{y}$, which is more permissible to outliers than the squared error. The cost J is defined as

$$J = \frac{1}{n} \sum_{v=1}^{n} |f_v - y_v|.$$

The network is trained with stochastic gradient descent and the Ada Delta (Zeiler, 2012) update rule using random restarts. Stochastic gradient descent is noisier than batch training due to a local estimation of the gradient, but it can start converging much faster. Ada Delta keeps an exponentially decaying history of gradients and updates in order to adapt the learning rate for each parameter, which partially smooths out the training noise. Regularisation and hyperparmeter selection are performed by early stopping on the development set. The size of the vocabulary is 90K words. Note that 10% of our lexicon is not found in the embeddings pretrained on Google News. Those terms are initialised with random small weights. The model has around 4 million weights plus 27 million tunable word embedding parameters.

Model Description	**MAE**($M)
Baseline mean	11.7
Linear Text	8.0
Linear Text+Domain+POS	7.4
Linear Meta	6.0
Linear Text+Meta	5.9
Linear Text+Meta+Domain+Deps	5.7
ANN Text	6.3
ANN Text+Domain	6.0
ANN Meta	3.9
ANN Text+Meta	**3.4**
ANN Text+Meta+Domain	**3.4**

Table 2: Experiment results on test set. Linear models by (Joshi et al., 2010).

Structured data Besides text, injecting meta data and domain information into the model likely provides additional predictive power. Combining text with structured data early in the network fosters joint non-linear interaction in subsequent hidden layers. Hence, if meta data $\mathbf{b}$ is present, we concatenate that with the max pooling layer $\mathbf{a}_1 = (\mathbf{p}, \mathbf{b})$ in part **III**. Domain specific information $\mathbf{t}$ is appended to each n-gram convolution input $(\mathbf{S}_{i,\cdot}^{(q)}, \mathbf{t})$ in part **II**, where $\mathbf{t}_z = \mathbf{1}_z$ indicates whether domain z has reviewed the movie.[2] This helps the network bias the convolutions, and thus change which features get propagated in the pooling layer.

3 Results

The results in Table 2 show that the neural network performs very well, with around 40% improvement over the previous best results (Joshi et al., 2010). Our dataset splits are identical, and we have accurately reproduced the results of their linear model. Non-linearities are clearly helpful as evidenced by the ANN Text model beating the bag of words Linear Text model with a mean absolute test error of 6.0 vs 8.0. Moreover, simply using structured data in the ANN Meta beats all the Linear models by a sizeable margin. Further improvements are realised through the inclusion of text, giving the lowest error of 3.4. Note that Joshi et al. (2010) preprocessed the text by stemming, down-

[2] Alternatively, site information can be encoded with one-hot categorical variables.

Model Description	MAE($M)
fixed word2vec embeddings	**3.4***
tuned word2vec embeddings	3.6
fixed random embeddings	3.6
tuned random embeddings	3.8
uni-grams	3.6
uni+bi-grams	3.5
uni+bi+tri-grams	**3.4***
uni+bi+tri+four-grams	3.6
0 hidden layer	6.3
1 hidden layer	3.9
2 hidden layers	3.5
3 hidden layers	**3.4***
4 hidden layers	3.6

Table 3: Various alternative configurations, based on the ANN Text+Meta model. The asterisk ($*$) denotes the settings in the ANN Text+Meta model.

casing, and discarding feature instances that occurred in fewer than five reviews. In contrast, we did not perform any processing of the text or feature engineering, apart from tokenization, instead learning this automatically.[3]

We find that both text and meta data contain complementary signals with some information overlap between them. This confirms the finding of Bitvai and Cohn (2015) on another text regression problem. The meta features alone almost achieve the best results whereas text alone performs worse but still well above the baseline. For the combined model, the performance improves slightly. In Table 3 we can see that contrary to expectations, fine tuning the word embeddings does not help significantly compared to keeping them fixed. Moreover, randomly initialising the embeddings and fixing them performs quite well. Fine tuning may be a challenging optimisation due to the high dimensional embedding space and the loss of monolingual information. This is further exacerbated due to the limited supervision signal.

One of the main sources of improvement appears to come from the non-linearity applied to the neural network activations. To test this, we try using linear activation units in parts **II** and **IV** of the network. Composition of linear functions yields a linear function, and therefore we recover the linear model results. This is much worse than the model with non-linear activations. Changing the network depth, we find that the model performs much better with a single hidden layer than without any, while three hidden layers are optimal. For the weight dimensions we find square 1058 dimensional weights to perform the best. The ideal number of convolutions are three with uni, bi and trigrams, but unigrams alone perform only slightly worse, while taking a larger n-gram window $n > 3$ does not help. Average and sum pooling perform comparatively well, while max pooling achieves the best result. Note that sum pooling recovers a non-linear bag-of-words model. With respect to activation functions, both ReLU and sigmoid work well.

Model extensions Multi task learning with task identifiers, ANN Text+Domain, does improve the ANN Text model. This suggests that the tendency by certain sites to review specific movies is in itself indicative of the revenue. However this improvement is more difficult to discern with the ANN Text+Meta+Domain model, possibly due to redundancy with the meta data. An alternative approach for multi-task learning is to have a separate convolutional weight matrix for each review site, which can learn site specific characteristics of the text. This can also be achieved with site specific word embedding dimensions. However neither of these methods resulted in performance improvements. In addition, we experimented with applying a hierarchical convolution over reviews in two steps with k-max pooling (Kalchbrenner et al., 2014), as well as parsing sentences recursively (Socher et al., 2013), but did not observe any improvements.

For optimisation, both Ada Grad and Steepest Gradient Descent had occasional problems with local minima, which Ada Delta was able to escape more often. In contrast to earlier work (Kim, 2014), applying dropout on the final layer did not improve the validation error. The optimiser mostly found good parameters after around 40 epochs which took around 30 minutes on a NVidia Kepler Tesla K40m GPU.

Model interpretation Next we perform analysis to determine which words and phrases influenced the output the most in the ANN Text model. To do so, we set each phrase input to zeros in turn and measure the prediction difference for each movie across the test set. We report the min/max/average/count values in Table 4. We isolate the effect of each n-gram by making sure the uni, bi and trigrams are independent,

[3] Although we do make use of pretrained word embeddings in our text features.

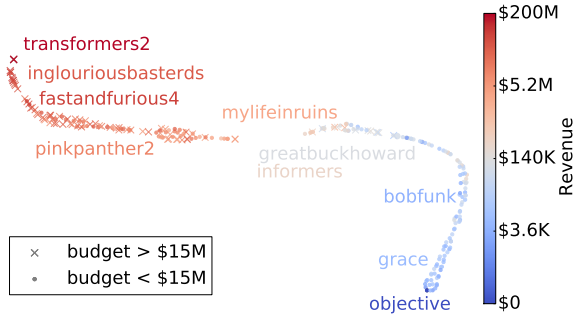

Figure 2: Projection of the last hidden layer of test movies using t-SNE. Red means high and blue means low revenue. The cross vs dot symbols indicate a production budget above or below $15M.

i.e. we process "Hong Kong" without zeroing "Hong" or "Kong". About 95% of phrases result in no output change, including common sentiment words, which shows that text regression is a different problem to sentiment analysis. We see that words related to series "# 2", effects, awards "praise", positive sentiment "intense", locations, references "Batman", body parts "chest", and others such as "plot twist", "evil", and "cameo" result in increased revenue by up to $5 million. On the other hand, words related to independent films "vérité", documentaries "the period", foreign film "English subtitles" and negative sentiment decrease revenue. Note that the model has identified structured data in the unstructured text, such as related to revenue of prequels "39 million", crew members, duration "15 minutes in", genre "[sci] fi", ratings, sexuality, profanity, release periods "late 2008 release", availability "In selected theaters" and themes. Phrases can be composed, such as "action unfolds" amplifies "action", and "cautioned" is amplified by "strongly cautioned". "functional" is neutral, but "functional at best" is strongly negative. Some words exhibit both positive and negative impacts depending on the context. This highlights the limitation of a linear model which is unable to discover these non-linear relationships. "13 - year [old]" is positive in New in Town, a romantic comedy and negative in Zombieland, a horror. The character strings "k /" (mannerism of reviewer), "they're" (unique apostrophe), "&#39" (encoding error) are high impact and unique to specific review sites, showing that the model indirectly uncovers domain information. This can explain the limited gain that can be achieved via multi task learning. Last, we have

Top 5 positive phrases	min	max	avg	#
sequel	20	4400	2300	28
flick	0	3700	1600	22
k /	1500	3600	2200	3
product	10	3400	1800	27
predecessor	22	3400	1400	13
Top 5 negative phrases	**min**	**max**	**avg**	**#**
Mildly raunchy lang.	-3100	-3100	-3100	1
(Under 17	-2500	1	-570	75
Lars von	-2400	-900	-1500	3
talk the language	-2200	-2200	-2200	1
. their English	-2200	-2200	-2200	1
Selected phrases	**min**	**max**	**avg**	**#**
CGI	145	3000	1700	28
action	-7	1500	700	105
summer	3	1200	560	42
they're	3	1300	530	68
1950s	10	1600	500	17
hit	8	950	440	72
fi	-15	340	160	26
Cage	7	95	45	28
Hong Kong	-440	40	-85	11
requires acc. parent	-780	1	-180	77
English	-850	6	-180	41
Sundance Film Festival	-790	3	-180	10
written and directed	-750	-3	-220	19
independent	-990	-2	-320	12
some strong language	-1600	6	-520	13

Table 4: Selected phrase impacts on the predictions in $ USD(K) in the test set, showing min, max and avg change in prediction value and number of occurrences (denoted #). Periods denote abbreviations (language, accompanying).

plotted the last hidden layer of each test set movie with t-SNE (Van der Maaten and Hinton, 2008). This gives a high level representation of a movie. In Figure 2 it is visible that the test set movies can be discriminated into high and low revenue groups and this also correlates closely with their production budget.

4 Conclusions

In this paper, we have shown that convolutional neural networks with deep architectures greatly outperform linear models even with very little supervision, and they can identify key textual and numerical characteristics of data with respect to predicting a real world phenomenon. In addition, we have demonstrated a way to intuitively interpret the model. In the future, we will investigate ways for automatically optimising the hyperparameters of the network (Snoek et al., 2012) and various extensions to recursive or hierarchical convolutions.

References

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

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Zsolt Bitvai and Trevor Cohn. 2015. Predicting peer-to-peer loan rates using Bayesian non-linear regression. In *Proceedings of the 29th AAAI conference on Artificial Intelligence*, pages 2203–2210.

Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A Smith. 2010. Movie reviews and revenues: An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 293–296.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Shimon Kogan, Dimitry Levin, Bryan R Routledge, Jacob S Sagi, and Noah A Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Vasileios Lampos and Nello Cristianini. 2010. Tracking the flu pandemic by monitoring the social web. In *Cognitive Information Processing*, pages 411–416.

Vasileios Lampos, Daniel Preotiuc-Pietro, and Trevor Cohn. 2013. A user-centric model of voting intention from social media. In *Proc 51st Annual Meeting of the Association for Computational Linguistics*, pages 993–1003.

Vasileios Lampos, Nikolaos Aletras, D Preoţiuc-Pietro, and Trevor Cohn. 2014. Predicting and characterising user impact on twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 405––413.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

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*, pages 1631–1642.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.