Quadratic Features and Deep Architectures for Chunking

Joseph Turian and **James Bergstra** and **Yoshua Bengio**
Dept. IRO, Université de Montréal

Abstract

We experiment with several chunking models. Deeper architectures achieve better generalization. Quadratic filters, a simplification of a theoretical model of V1 complex cells, reliably increase accuracy. In fact, logistic regression with quadratic filters outperforms a standard single hidden layer neural network. Adding quadratic filters to logistic regression is almost as effective as feature engineering. Despite predicting each output label independently, our model is competitive with ones that use previous decisions.

1 Introduction

There are three general approaches to improving chunking performance: engineer better features, improve inference, and improve the model.

Manual feature engineering is a common direction. One technique is to take primitive features and manually compound them. This technique is common, and most NLP systems use n-gram based features (Carreras and Màrquez, 2003; Ando and Zhang, 2005, for example). Another approach is linguistically motivated feature engineering, e.g. Charniak and Johnson (2005).

Other works have looked in the direction of improving inference. Rather than predicting each decision independently, previous decisions can be included in the inference process. In this work, we use the simplest approach of modeling each decision independently.

The third direction is by using a better model. If modeling capacity can be added without introducing too many extra degrees of freedom, generalization could be improved. One approach for compactly increasing capacity is to automatically induce intermediate features through the composition of non-linearities, for example SVMs with a non-linear kernel (Kudo and Matsumoto, 2001), inducing compound features in a CRF (McCallum, 2003), neural networks (Henderson, 2004; Bengio and LeCun, 2007), and boosting decision trees (Turian and Melamed, 2006). Recently, Bergstra et al. (2009) showed that capacity can be increased by adding quadratic filters, leading to improved generalization on vision tasks. This work examines how well quadratic filters work for an NLP task. Compared to manual feature engineering, improved models are appealing because they are less task-specific.

We experiment on the task of chunking (Sang and Buchholz, 2000), a syntactic sequence labeling task.

2 Sequence labeling

Besides Collobert and Weston (2008), previous work on sequence labeling usually use previous decisions in predicting output labels. Here we do not take advantage of the dependency between successive output labels. Our approach predicts each output label independently of the others. This allows us to ignore inference during training: The model maximizes the conditional likelihood of each output label independent of the output label of other tokens.

We use a sliding window approach. The output label for a particular focus token x_i is predicted based upon $\bar{k}$ tokens before and after x_i. The entire window is of size $k = 2 \cdot \bar{k} + 1$. Nearly all work on sequence labeling uses a sliding window approach (Kudo and Matsumoto, 2001; Zhang et al., 2002;

Proceedings of NAACL HLT 2009: Short Papers, pages 245–248,
Boulder, Colorado, June 2009.

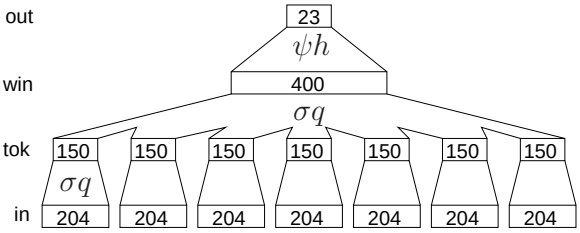

Figure 1: Illustration of our baseline I-T-W-O model (see Secs. 4 and 5.1). The input layer comprises seven tokens with 204 dimensions each. Each token is passed through a shared 150-dimensional token feature extractor. These $7 \cdot 150$ features are concatenated and 400 features are extracted from them in the window layer. These 400 features are the input to the final 23-class output prediction. Feature extractors σq and ψh are described in Section 3.

Carreras and Màrquez, 2003; Ando and Zhang, 2005, for example). We assume that each token x can be transformed into a real-valued feature vector $\phi(x)$ with l entries. The feature function will be described in Section 4.

A standard approach is as follows: We first concatenate the features of k tokens into one vector $[\phi(x_{i-\bar{k}}), \ldots, \phi(x_{i+\bar{k}})]$ of length $k \cdot l$ entries. We can then pass $[\phi(x_{i-\bar{k}}), \ldots, \phi(x_{i+\bar{k}})]$ to a feature extractor over the entire window followed by an output log-linear layer.

Convolutional architectures can help when there is a position-invariant aspect to the input. In machine vision, parameters related to one part of the image are sometimes restricted to be equal to parameters related to another part (LeCun et al., 1998). A convolutional approach to sequence labeling is as follows: At the lowest layer we extract features from individual tokens using a shared feature extractor. These higher-level individual token features are then concatenated, and are passed to a feature extractor over the entire window.

In our baseline approach, we apply one convolutional layer of feature extraction to each token (one *token layer*), followed by a concatenation, followed by one layer of feature extraction over the entire window (one *window layer*), followed by a 23-D output prediction using multiclass logistic regression. We abbreviate this architecture as I-T-W-O (input→token→window→output). See Figure 1 for an illustration of this architecture.

3 Quadratic feature extractors

The most common feature extractor in the literature is a linear filter h followed by a non-linear squashing (activation) function σ:

$$f(x) = \sigma(h(x)), \quad h(x) = \boldsymbol{b} + \boldsymbol{W}x. \tag{1}$$

In our experiments, we use the softsign squashing function $\sigma(z) = z/(1 + |z|)$. n-class logistic regression predicts $\psi(h(x))$, where softmax $\psi_i(z) = \exp(z_i)/\sum_k \exp(z_k)$. Rust et al. (2005) argues that complex cells in the V1 area of visual cortex are not well explained by Eq. 1, but are instead better explained by a model that includes quadratic interactions between regions of the receptive field. Bergstra et al. (2009) approximate the model of Rust et al. (2005) with a simpler model of the form given in Eq. 2.[†] In this model, the pre-squash transformation q includes J quadratic filters:

$$f(x) = \sigma(q(x)), \quad q(x) = \left(\boldsymbol{b} + \boldsymbol{W}x + \sqrt{\sum_{j=1}^{J} (\boldsymbol{V}_j x)^2} \right) \tag{2}$$

where $\boldsymbol{b}$, $\boldsymbol{W}$, and $\boldsymbol{V}_1 \ldots \boldsymbol{V}_J$ are tunable parameters.

In the vision experiments of Bergstra et al. (2009), using quadratic filters improved the generalization of the trained architecture. We were interested to see if the increased capacity would also be beneficial in language tasks. For our logistic regression (I-O) experiments, the architecture is specifically I–ψq→O, i.e. output O is the softmax ψ applied to the quadratic transform q of the input I. Like Bergstra et al. (2009), in architectures with hidden layers, we apply the quadratic transform q in all layers *except* the final layer, which uses linear transform h. For example, I-T-W-O is specifically I–σq→T–σq→W–ψh→O, as shown in Figure 1. Future work will explore if generalization is improved by using q in the final layer.

4 Features

Here is a detailed description of the types of features we use, with number of dimensions:

• **embeddings**. We map each word to a real-valued 50-dimensional embedding. These embeddings were obtained by Collobert and Weston (2008), and

[†] Bergstra et al. (2009) do not use a sqrt in Eq. 2. We found that sqrt improves optimization and gives better generalization.

were induced based upon a purely unsupervised training strategy over the 631 million words in the English Wikipedia.

• **POS-tag**. Part-of-speech tags were assigned automatically, and are part of the CoNLL data. 45 dim.

• **label frequencies**. Frequency of each label assigned to this word in the training and validation data. From Ando and Zhang (2005). 23 dim.

• **type(first character)**. The type of the first character of the word. type(x) = 'A' if x is a capital letter, 'a' if x is a lowercase letter, 'n' if x is a digit, and x otherwise. From Collins (2002). 20 dim.

• **word length**. The length of the word. 20 dim.

• **compressed word type**. We convert each character of the word into its type. We then remove any repeated consecutive type. For example, "Label-making" ⇒ "Aa-a". From Collins (2002). 46 dim.

The last three feature types are based upon orthographic information. There is a combined total of 204 features per token.

5 Experiments

We follow the conditions in the CoNLL-2000 shared task (Sang and Buchholz, 2000). Of the 8936 training sentences, we used 1000 randomly sampled sentences (23615 words) for validation.

5.1 Training details

The optimization criterion used during training is the maximization of the sum (over word positions) of the per-token log-likelihood of the correct decision. Stochastic gradient descent is performed using a fixed learning rate η and early stopping. Gradients are estimated using a minibatch of 8 examples. We found that a learning rate of 0.01, 0.0032, or 0.001 was most effective.

In all our experiments we use a window size of 7 tokens. In preliminary experiments, smaller windows yielded poorer results, and larger ones were no better. Layer sizes of extracted features were chosen to optimize validation F1.

5.2 Results

We report chunk F-measure (F1). In some tables we also report Acc, the per-token label accuracy, *post*-Viterbi decoding.

Figure 2 shows that using quadratic filters reliably improves generalization on all architectures. For the I-T-W-O architecture, quadratic filters increase validation F1 by an absolute 0.31. Most surprisingly, logistic regression with 16 filters achieves F1=93.94, which outperforms the 93.83 of a standard (0 filter) single hidden layer neural network.

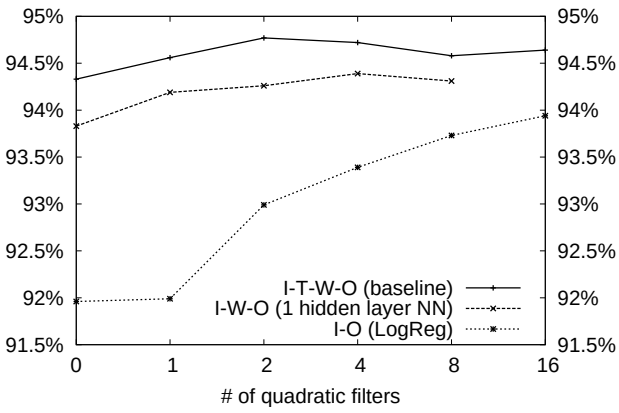

Figure 2: Validation F1 (y-axis) as we vary the number of quadratic filters (x-axis), over different model architectures. Both architecture depth and quadratic filters improve validation F1.

Architecture	#qf	Acc	F1
I-O	16	96.45	93.94
I-W(400)-O	4	96.66	94.39
I-T(150)-W(566)-O	2	96.85	94.77
I-T(150)-W(310)-W(310)-O	4	96.87	94.82

Table 1: Architecture experiments on validation data. The first column describes the layers in the architecture. (The architecture in Figure 1 is I-T(150)-W(400)-O.) The second column gives the number of quadratic filters. For each architecture, the layer sizes and number of quadratic filters are chosen to maximize validation F1. Deeper architectures achieve higher F1 scores.

With embeddings as the only features, logreg with 0 filters achieves F1=85.36. By adding all features, we can raise the F1 to 91.96. Alternately, by adding 16 filters, we can raise the F1 to 91.60. In other words, adding filters is nearly as effective as our manual feature engineering.

Table 1 shows the result of varying the overall architecture. Deeper architectures achieve higher F1 scores. Table 2 compares the model as we lesion off different features. POS tags and the embeddings were the most important features.

We applied our best model overall (I-T-W-W-O in Table 1) to the test data. Results are shown in

Feature set	Acc	F1
default	96.81	94.69
no orthographic features	96.84	94.62
no label frequencies	96.77	94.58
no POS tags	96.60	94.22
no embeddings	96.40	93.97
only embeddings	96.18	93.53

Table 2: Results on validation of varying the feature set, for the architecture in Figure 1 with 4 quadratic filters.

	NP F1	Prc	Rcl	F1
AZ05	94.70	94.57	94.20	94.39
KM01	94.39	93.89	93.92	93.91
I-T-W-W-O	94.44	93.72	93.91	93.81
CM03	94.41	94.19	93.29	93.74
SP03	94.38	-	-	-
Mc03	93.96	-	-	-
AZ05-	-	93.83	93.37	93.60
ZDJ02	93.89	93.54	93.60	93.57

Table 3: Test set results for Ando and Zhang (2005), Kudo and Matsumoto (2001), our I-T-W-W-O model, Carreras and Màrquez (2003), Sha and Pereira (2003), McCallum (2003), Zhang et al. (2002), and our best I-O model. AZ05- is Ando and Zhang (2005) using purely supervised training, not semi-supervised training. Scores are noun phrase F1, and overall chunk precision, recall, and F1.

Table 3. We are unable to compare to Collobert and Weston (2008) because they use a different training and test set. Our model predicts all labels in the sequence independently. All other works in Table 3 use previous decisions when making the current label decision. Our approach is nonetheless competitive with approaches that use this extra information.

6 Conclusions

Many NLP approaches underfit important linguistic phenomena. We experimented with new techniques for increasing chunker model capacity: adding depth (automatically inducing intermediate features through the composition of non-linearities), and including quadratic filters. Higher accuracy was achieved by deeper architectures, i.e. ones with more intermediate layers of automatically tuned feature extractors. Although they are a simplification of a theoretical model of V1 complex cells, quadratic filters reliably improved generalization in all architectures. Most surprisingly, logistic regression with quadratic filters outperformed a single hidden layer neural network without. Also, with logistic regression, adding quadratic filters was almost as effective as manual feature engineering. Despite predicting each output label independently, our model is competitive with ones that use previous decisions.

Acknowledgments

Thank you to Ronan Collobert, Léon Bottou, and NEC Labs for access to their word embeddings, and to NSERC and MITACS for financial support.

References

R. Ando and T. Zhang. A high-performance semi-supervised learning method for text chunking. In *ACL*, 2005.

Y. Bengio and Y. LeCun. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. 2007.

J. Bergstra, G. Desjardins, P. Lamblin, and Y. Bengio. Quadratic polynomials learn better image features. TR 1337, DIRO, Université de Montréal, 2009.

X. Carreras and L. Màrquez. Phrase recognition by filtering and ranking with perceptrons. In *RANLP*, 2003.

E. Charniak and M. Johnson. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *ACL*, 2005.

M. Collins. Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In *ACL*, 2002.

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

J. Henderson. Discriminative training of a neural network statistical parser. In *ACL*, 2004.

T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *NAACL*, 2001.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. *IEEE*, 86(11):2278–2324, November 1998.

A. McCallum. Efficiently inducing features of conditional random fields. In *UAI*, 2003.

N. Rust, O. Schwartz, J. A. Movshon, and E. Simoncelli. Spatiotemporal elements of macaque V1 receptive fields. *Neuron*, 46(6):945–956, 2005.

E. T. Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *CoNLL*, 2000.

F. Sha and F. C. N. Pereira. Shallow parsing with conditional random fields. In *HLT-NAACL*, 2003.

J. Turian and I. D. Melamed. Advances in discriminative parsing. In *ACL*, 2006.

T. Zhang, F. Damerau, and D. Johnson. Text chunking based on a generalization of Winnow. *JMLR*, 2, 2002.