Robust cross-domain disfluency detection with pattern match networks

Vicky Zayats
Electrical Engineering Department
University of Washington
vzayats@uw.edu

Mari Ostendorf
Electrical Engineering Department
University of Washington
ostendor@uw.edu

Abstract

In this paper we introduce a novel pattern match neural network architecture that uses neighbor similarity scores as features, eliminating the need for feature engineering in a disfluency detection task. We evaluate the approach in disfluency detection for four different speech genres, showing that the approach is as effective as hand-engineered pattern match features when used on in-domain data and achieves superior performance in cross-domain scenarios.

1 Introduction

Disfluencies are self corrections in spontaneous speech, including filled pauses, repetitions, repairs and false starts. Below are some examples of disfluent sentences from the corpora used in this work:

[He is + our clients are] subject to
[it 's + {you know} it 's] one of the last
you want [it + just something] that is

The brackets indicate the beginning and of the disfluency and the end of the correction. The reparandum includes the words before the interruption point (+), which the speaker intends to replace or ignore. The words that come as a correction to the reparandum follow the interruption point. An optional interregnum (in brackets) follows the interruption point, including words such as filled pauses, discourse markers, etc. Systems are usually evaluated on the ability to correctly identify the reparandum.

Previous studies on disfluency detection observe that a repair is often a "rough copy" of a reparandum (Charniak and Johnson, 2001; Zwarts et al., 2010); thus, hand-crafted pattern match features play an important role in many disfluency detection approaches. They have been shown to be helpful to both sequential and parsing based approaches (Wang et al., 2017; Zayats et al., 2016; Ferguson et al., 2015; Wu et al., 2015; Qian and Liu, 2013). In the examples above, "he" resembles "clients", "is" resembles "are" and "it 's" is a repetition. However, in many cases, the pattern match is not simple, if present at all, as in the last example. In addition, disfluencies can have domain-dependent characteristics.

In this work. we present a novel architecture that allows automated discovery of the patterns instead. We first calculate a similarity score between neighboring words in a sentence. Then, we use those scores directly to identify multi-token patterns with convolutional neural networks (CNN). Experiments show that our proposed architecture has an in-domain performance comparable to using hand-crafted pattern match features, and it outperforms baselines in cross-domain setting. In this paper, our main contribution is a novel neural network architecture which allows automatic discovery of patterns using a mechanism similar to attention but where the similarity scores serve as input features to a CNN, rather than as weights for computing a context vector. In addition to eliminating the need for feature engineering, the model is shown to be robust in cross-domain testing.

2 Method

The main motivation behind our approach is to allow the model to automatically learn and find patterns in sentences without defining them via hand-crafted features. Our proposed model uses two levels to automatically find patterns in sentences. In the first level we calculate similarities for each word in a sentence with words in the surrounding window, which we refer to as neighbor similarity. After calculating the single-token similarity weights, in the second level, we use those weights

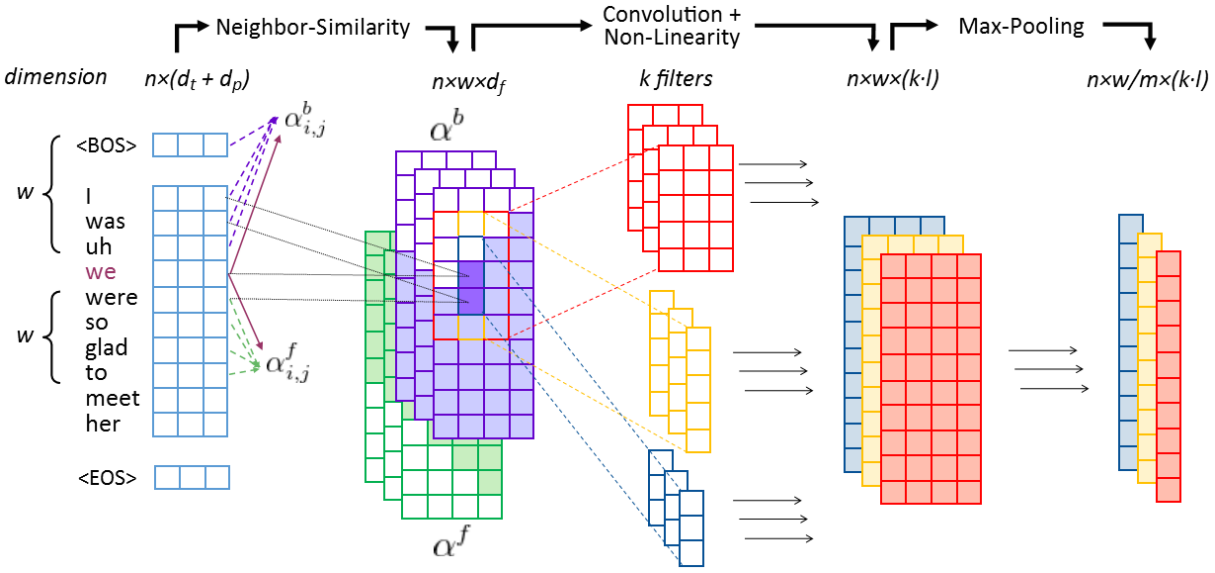

Figure 1: An illustration of the model. In this example, the backward neighbor-similarity layer α^b identifies that *"we"* has high similarity with *"I"* and *"were"* has high similarity with *"was"*. Both *"we"* and *"were"* are at a distance of 3 from the corresponding *"I"* and *"was"*. A convolutional filter can catch the horizontal pattern in row 3, thus indicating the presence of a bigram pattern match between *"we were"* and *"I was"*.

as features to extract local patterns using a convolutional neural network. The schematic diagram of the model is presented in Figure 1.

2.1 Neighbor Similarity

The hand-crafted pattern match features used in disfluency detection are usually in the form of "*does the exact word/POS/bigram appeared previously in a fixed length window?*" In our work, instead of manually defining similarity functions (e.g. exact match of the word/POS), we learn similarity functions between individual words in a sentence. For each word in a sentence of length n, we calculate a similarity between the given word x_i and each of the words x_j in the preceding/following window of size w, for $j \in [i \pm 1, \ldots, i \pm w]$. In our task we used cosine similarity sim to calculate the alignment score between a pair of words due to the straightforward resemblance of words in the reparandum and repair:

$$\alpha_{i,j}^{\{f,b\}} = sim(W^1 x_i, W^2 x_j) \qquad (1)$$

where $W^1, W^2 \in \mathbb{R}^{d_f \times d_g \times (d_t + d_p)}$ are learned.

We refer to similarity scores in the preceding/following windows as α^b (backward) and α^f (forward), respectively. For the cases when the x_j is outside of the sentence boundaries, we set the similarity score to be zero. In order to capture multiple types of similarities between two word representations, we concatenate token and part-of-speech (POS) embeddings and learn multi-dimensional similarity scores $\alpha_{i,j} \in \mathbb{R}^{d_f}$. In our experiments, we set $d_g = (d_t + d_p)$, where d_t and d_p are the dimensions of token and POS embeddings, respectively. The overall dimension of the similarity matrix is $\alpha \in \mathbb{R}^{w \times n \times d_f}$, where n is the sentence length and w is the size of the window.

2.2 Convolution over Similarity Features

While neighbor-similarity features can be useful, they do not exploit all the information about repeating patterns. A simple example can be a bigram pattern match feature: the model can find a similarity between closely related words on the unigram level, but it is unable to directly identify cases where the bigram would be repeated. To capture temporal patterns presented in neighbor-similarity scores, we apply convolutional filters on the output of the neighbor-similarity layer, followed by a non-linearity (tanh). For example, in Figure 1, the neighbor-similarity layer would identify similarity between individual tokens *"we"* and *"I"*, and *"were"* and *"was"*. A convolutional layer on top would capture the horizontal bigram pattern between *"we were"* and *"I was"*. The output of the convolutional layer is $f_{conv}(\alpha) \in \mathbb{R}^{w \times n \times kl}$, where k is number of different filter shapes and l is the number of output filters for each filter shape. We apply the max-pooling layer with a downsample rate m on top to summarize the convolutional layer output at each time i. The output of the max-pooling layer is $g(\alpha) \in \mathbb{R}^{w/m \times n \times kl}$. We flatten the outputs of the max-pooling layer

Model	pattern	SWBD test	CallHome	SCOTUS	FCIC
CRF	–	71.3	58.1	70.8	53.2
	✓	82.5	63.2	79.2	63.8
LSTM	–	82.9	54.1	57.9	36.9
	✓	**86.8**	58.7	66.6	48.9
LSTM + sim	–	85.9	64.8	78.8	65.0
LSTM + sim + conv	–	86.7	**65.2**	**79.9**	**66.1**

Table 1: F1 scores on cross-domain disfluency detection. "pattern" stands for hand-crafted pattern match features.

Model	CallHome	SCOTUS	FCIC
CRF	71.5	90.9	88.7
LSTM	54.7	71.8	58.7
LSTM pm net	68.4	88.0	87.2

Table 2: Precision across domains for CRF, LSTM with hand-engineered pattern match features and LSTM with pattern match networks.

and concatenate with the input feature embeddings, and input the resulting vector to an LSTM.

3 Experiments and Analysis

Our experiments target both in-domain and cross-domain scenarios. In addition, we analyze the differences in errors made by the models.

3.1 Data

Switchboard (SWBD) (Godfrey et al., 1992) is the standard and largest corpus used for disfluency detection. The current state-of-the-art in disfluency detection achieves F1 score of 88.1 on the SWBD test set (Wang et al., 2017). In addition to Switchboard, we test our models on three out-of-domain publicly available datasets annotated with disfluencies (Zayats et al., 2014):

Switchboard: phone conversations between strangers on predefined topics;

CallHome: phone conversations between family members and close friends;

SCOTUS: transcribed Supreme Court oral arguments between justices and advocates;

FCIC: two transcribed hearings from Financial Crisis Inquiry Commission.

3.2 Model Comparisons

We train the CRF [1] and bidirectional LSTM-CRF[2] models as baselines, both with and without pattern match features. For simplicity, we refer to bidirectional LSTM-CRF model as just LSTM.

[1] https://taku910.github.io/crfpp

[2] https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf

In all models we use **identity features**, including word, POS tag, whether the word is a filled pause, discourse marker, edit word or fragment. The **hand-crafted pattern match features** include: distance to the repeated {word, bigram, POS, word+next POS, POS bigram, POS trigram} in the {preceding, following} window; whether word bigram is repeated in the {preceding, following} window allowing some words in between the two words; and distance to the next conjunction word. Following (Zayats et al., 2016), we use 8 BIO states. For each experiment, we average the performance of 15 randomly initialized models.

For our proposed model we use the following parameters: window size $w = 10$, neighbor-similarity dimension $d_f = 100$, $k = 5$ different filter shapes: $[1, 1], [3, 1], [3, 3], [5, 1]$ and $[5, 3]$, with output filter dimension $l = 16$, and down-sample rate $m = 3$. In our initial experiments we have tuned the parameters d_f, k, l and m to the values mentioned above using the Switchboard development set.

3.3 Results

The cross-domain experiment results are presented in Table 1. In general, for the in-domain data (Switchboard), the pattern match networks achieve performance comparable to the LSTM model with hand-crafted pattern match features, and significantly outperforms the CRF model. In addition, our model is robust compared to the baselines when applied to out-of-the-domain data, with a consistent improvement over the CRF. Surprisingly, the LSTM performs poorly on out-of-the-domain data. To better understand the model differences, in the next section we conduct error analysis and discuss the findings.

3.4 Error analysis

The difficulty in applying the model on out-of-domain data lie in both difference in corpora and underlying model. There is substantial variation in

Corpus	Ex	Sentence
CallHome	1	Oh he [looks like + John Travolta but he has like] curly blond hair.
	2	[I do n't think + [I know her + but I 've]] heard of her
SCOTUS	3	What is your [authority + for that proposition, Mr. Guttentag, your case authority]?
	4	. . . as to permit review in [the court + of appeals, then the district court] habeas corpus procedure need . . .
FCIC	5	Thank you for the opportunity [to + contribute to the commission 's work to] understand the causes of . . .
	6	. . . counter parties were unaware [of + the full extent of] those vehicles and therefore could not . . .

Table 3: Example sentences wrongly predicted as disfluent by LSTM model with hand-crafted pattern match features. The brackets indicate predicted disfluency regions, where the respected gold annotation is "non-disfluent".

vocabulary, conversational style, disfluency types, and sentence segmentation criteria across corpora. CallHome is more casual than SWBD; SCOTUS and FCIC are formal high stakes discussions with vocabularies highly dissimilar to SWBD. SCOTUS, FCIC, and CallHome contain 2, 5 and 7 times more restarts token-wise, respectively, than Switchboard. Also, on average, disfluencies in all three out-of-the-domain corpora tend to be longer, especially in CallHome and FCIC.

To further study the effect of pattern match features, we trained models with identity features only. When comparing models with identity features only, CRF performs poorly compared to LSTM on in-domain data. On the other hand, LSTM with no pattern match features performs considerably well on in-domain Switchboard. By looking at cross-domain results, we see that CRF is more stable across the domains, compared to the LSTM. We hypothesize that LSTM is more powerful in learning specific data structure, compared to the CRF, and overfit the models to match Switchboard style. On the other hand, pattern match networks are better at capturing patterns that are more general across domains.

Table 2 shows that that there is a significant drop in precision for LSTMs with hand-derived features. In particular, the LSTM with pattern match features "hallucinates" a lot of disfluencies, longer ones in particular. This might be due to the long memory of the LSTM, as opposed to CRF, which tends to be more local. Table 3 presents some examples with false positives made by the LSTM with hand-engineered features, but were correctly identify by our model.

4 Related Work

Most work on disfluency detection fall into three main categories: sequence tagging, noisy-channel and parsing-based approaches. Sequence tagging approaches include conditional random fields (CRF) (Georgila, 2009; Ostendorf and Hahn, 2013; Zayats et al., 2014), Max-Margin Markov Networks (M^3N) (Qian and Liu, 2013), Semi-Markov CRF (Ferguson et al., 2015), and recurrent neural networks (Hough and Schlangen, 2015; Zayats et al., 2016; Wang et al., 2016). The main benefit of sequential models is the ability to capture long-term relationships between reparandum and repairs. Noisy channel models operate on a relationship between the reparandum and repair for identifying disfluencies (Charniak and Johnson, 2001; Zwarts et al., 2010). Lou and Johnson (2017) used a neural language model to rerank sentences using the noisy channel model. Approaches that combine parsing and disfluency removal tasks include (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Tran et al., 2017). The current state-of-the-art in disfluency detection uses a transition-based neural model architecture (Wang et al., 2017).

There exist a limited effort on cross-domain disfluency detection. Georgila et al. (2010) used CRF and integer linear programming in detecting disfluencies in human-agent interactions. Zayats et al. (2014) introduced pattern match features with a CRF and released the datasets that we use for testing in our experiments. Zayats et al. (2015) used semi-supervised learning in adapting the model to SCOTUS non-careful transcripts.

5 Conclusion

In this paper we introduce a novel neural network architecture which allows automatic discovery of patterns and directly uses similarity scores as input features to a CNN. We show that our approach can be as effective as using carefully designed, hand-engineered pattern match features in a disfluency detection task, eliminating the need for feature engineering, and show that it is robust in cross-domain testing. In the future, following Ganin and Lempitsky (2014), we are interested in exploring domain adaptation techniques in disfluency detection. Motivated by Tran et al. (2017), we are also interested in incorporating prosody information to further improve disfluency detection.

References

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proc. NAACL*. pages 118–126.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency detection with a semi-markov model and prosodic features. In *Proc. NAACL HLT*.

Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495* .

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proc. NAACL HLT*.

Kallirroi Georgila, Ning Wang, and Jonathan Gratch. 2010. Cross-domain speech disfluency detection. In *Proc. Annual SIGdial Meeting on Discourse and Dialogue*.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proc. ACL*. volume I, pages 517–520.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics* 2(1):131–142.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. In *Proc. Interspeech*.

Paria Jamshid Lou and Mark Johnson. 2017. Disfluency detection using a noisy channel model and a deep neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 547–553.

Mari Ostendorf and Sangyun Hahn. 2013. A sequential repetition model for improved disfluency detection. In *Proc. Interspeech*.

Xian Qian and Yang Liu. 2013. Disuency detection using multi-step stacked learning. In *Proc. NAACL HLT*.

Mohammad Sadegh Rasooli and Joel R Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *Proc. EMNLP*. pages 124–129.

Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf. 2017. Joint modeling of text and acoustic-prosodic cues for neural parsing. *arXiv preprint arXiv:1704.07287* .

Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A neural attention model for disfluency detection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 278–287.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based disfluency detection using lstms. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2785–2794.

Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2015. Efficient disfluency detection with transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 495–503.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional lstm. *arXiv preprint arXiv:1604.03209* .

Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2014. Multidomain disfluency and repair detection. In *Proc. Interspeech*.

Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2015. Unediting: Detecting disfluencies without careful transcripts. In *Proc. NAACL HLT*.

Simon Zwarts, Mark Johnson, and Robert Dale. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proc. Coling*. pages 1371–1378.