# DCU-Paris13 Systems for the SANCL 2012 Shared Task

**Joseph Le Roux**†, **Jennifer Foster**‡, **Joachim Wagner**‡,
**Rasul Samad Zadeh Kaljahi**‡, **Anton Bryl**§
†Université Paris Nord – CNRS UMR 7030, France
‡, NCLT/CNGL, School of Computing, Dublin City University, Ireland
§Systransis AG, Zug, Switzerland
†leroux@univ-paris13.fr
‡firstname.lastname@computing.dcu.ie
§a.bryl@systransis.ch

## Abstract

The *DCU-Paris13* team submitted three systems to the SANCL 2012 shared task on parsing English web text. The first submission, the highest ranked constituency parsing system, uses a combination of PCFG-LA product grammar parsing and self-training. In the second submission, also a constituency parsing system, the $n$-best lists of various parsing models are combined using an approximate sentence-level product model. The third system, the highest ranked system in the dependency parsing track, uses voting over dependency arcs to combine the output of three constituency parsing systems which have been converted to dependency trees. All systems make use of a data-normalisation component, a parser accuracy predictor and a genre classifier.

## 1 Introduction

This paper describes the three systems we submitted to the shared task on parsing English web data organised by Petrov and McDonald (2012) and hosted by the 2012 NAACL-HLT Workshop on Syntactic Analysis of Non-Canonical Language (SANCL). The aim of the shared task was to encourage research groups around the world to build robust systems capable of parsing English documents belonging to the following five web genres: answers, emails, newsgroups, reviews and weblogs. To train their systems, participants were supplied with labelled Wall Street Journal (WSJ) data (annotated in the Ontonotes style) and unlabelled data from the five web genres. They were also provided with WSJ, email and weblog development data. One week before the deadline, four blind tests were released.

Our first system, *DCU-Paris13-1*, the top ranked constituency parsing system, employs self-training with products of PCFG-LA grammars (Huang et al., 2010). We use the Lorg parser (Attia et al., 2010), our in-house PCFG-LA parser (Matsuzaki et al., 2005; Petrov et al., 2006). Our second system, *DCU-Paris-2*, the fourth ranked constituency parsing system, uses a sentence-level product model to rerank the outputs of first-stage Brown models (Charniak, 2000; Charniak and Johnson, 2005) trained using increasingly large training sets. Our third system, *DCU-Paris13-Dep*, the top ranked dependency parsing system, converts constituency parser output created using various versions of the *DCU-Paris13-2* system into dependency trees and then combines these trees using a voting algorithm (Surdeanu and Manning, 2010).

The general architecture common to all three systems is displayed in Figure 1. The training and test web sentences are normalised before parsing. The baseline parser that is used to parse the unlabelled web sentences is a Lorg product model trained on the Ontonotes-WSJ training material. The product model combines eight 5th-order PCFG-LA grammars trained using different random seeds. Five split-merge-smooth cycles are employed rather than six since there is some evidence that a 6th-order model overfits to WSJ data (Petrov and Klein, 2007; Foster, 2010). The parser uses the English signature list described in Attia et al (2010) to assign part-of-speech tags to unknown words. Parser accuracy prediction (Ravi et al., 2008) is carried out on the

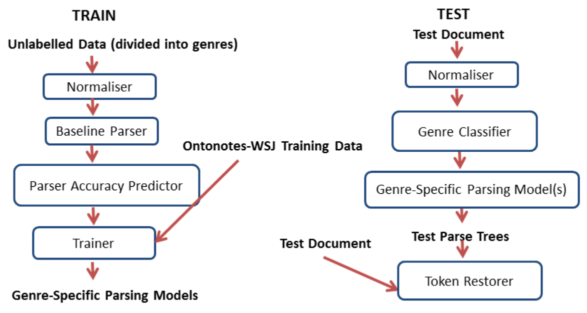
TRAIN
Unlabelled Data (divided into genres)
Normaliser
Baseline Parser
Parser Accuracy Predictor
Trainer
Genre-Specific Parsing Models
Ontonotes-WSJ Training Data
TEST
Test Document
Normaliser
Genre Classifier
Genre-Specific Parsing Model(s)
Test Parse Trees
Test Document
Token Restorer

Figure 1: General Architecture

| Mix | A | E | N | R | W |
|---|---|---|---|---|---|
| A SP | **40.0** | **40.0** | 0.0 | 0.0 | 20.0 |
| E SP | 0.0 | **100.0** | 0.0 | 0.0 | 0.0 |
| N SP | 0.0 | 0.0 | **100.0** | 0.0 | 0.0 |
| R SP | 0.0 | 0.0 | 0.0 | **100.0** | 0.0 |
| W SP | 0.0 | 0.0 | 0.0 | 0.0 | **100.0** |
| A BK | 26.6 | **27.6** | 10.5 | 10.9 | 24.4 |
| E BK | 5.2 | **50.0** | 19.6 | 0.9 | 24.3 |
| N BK | 4.5 | 22.4 | **50.0** | 3.2 | 19.9 |
| R BK | 19.9 | 12.1 | 2.2 | **50.0** | 15.8 |
| W BK | 15.0 | 13.5 | 12.9 | 8.6 | **50.0** |

Table 1: Genre-mixture of answers (A), emails (E), newsgroups (N), reviews (R) and weblogs (W) for genre-specific (SP) grammars and backoff (BK) grammars: All grammars are trained on 2 copies of the Ontonotes-WSJ training data and the above mixture of genre material (measured in number of tokens and rounded to the next tree). We train the genre-specific grammars with 100% of the respective genre and 2 Ontonotes copies (except for the answers due to insufficient unlabelled data). The genre-mixtures for the backoff grammars are derived from the genre classifier confusion matrices for test document lengths 500, 1k, 2k, 5k and 10k. The backoff grammars are only used in our *DCU-Paris13-Dep* system.

automatically parsed data in order to select instances for use in re-training. Preliminary experiments with the email and blog data suggested that better performance could be obtained by training on sentences from the same genre and so we train genre-specific grammars (see the upper half of Table 1) and rely on a genre classifier to choose the appropriate grammar during test time.

We describe the components common to all three systems in Section 2. In Sections 3, 4 and 5, we describe in more detail our three systems. Finally, we briefly outline our plans in Section 6.

## 2 Common Components

### 2.1 Data Normalisation

Following our previous work on parsing the British National Corpus (Wagner et al., 2007) and discussion forum comments (Foster, 2010), we transform the training and test data so that the sentences more closely resemble WSJ sentences. We use the following heuristics:

- Depending on its position within the input sentence, an emoticon is replaced by either a comma or full stop.
- Neutral quotes are transformed to opening or closing quotes.
- Email addresses and URLs are replaced by the generic strings *EmailAddress* and *LinkAddress* respectively.
- Sequences of uppercased words or individual uppercased words of length $>4$ letters are lowercased.
- Common abbreviations and spelling variants (*ppl*, *plz*) are replaced by their standard form.[1]
- The tokens *nt* and *s* are replaced by *n't* and *'s* respectively.
- Repeated punctuations symbols, e.g. *!!!* are collapsed into one.
- List items, e.g. *# 2*, are removed from the start of a sentence.
- Certain "sentences" are not parsed and instead are assigned a trival parse tree rooted by the symbol `X` with each token tagged as `NFP`. These include sequences of punctuation symbols separating the content of an email from its signature and sentences containing only a URL.

### 2.2 Parser Accuracy Prediction

Inspired by previous work in parser accuracy prediction (Ravi et al., 2008), we parse all the unlabelled data using the baseline Ontonotes-WSJ-trained parser and then sort the parsed output ac-

[1]The list of abbreviations is rather limited having been obtained in an ad-hoc manner by reading the annotation guidelines and manually inspecting approximately fifty sentences from each of the unlabelled training sets.

cording to its predicted Parseval f-score. We do this in an attempt to maximize the number of high-quality web trees in our training material. The f-score predictor is trained on the Ontonotes-WSJ development data using support vector regression with an RBF kernel and the following features:

- Sentence length
- Number of words in the sentence that are not in the Ontonotes-WSJ training set
- The f-score of the tree measured against a reference parse tree, produced using the first-stage Brown parser (Charniak, 2000)
- The presence and count of particular discriminative words, where discriminativeness is measured using information gain
- The category of the root of the tree
- For each Ontonotes-WSJ category, the number of nodes in the tree labelled with that category
- The depth of the parse tree

When we first applied this predictor to the automatically parsed trees, we observed that the highest ranked trees contained very few tokens and did not appear to represent good training material. For example, many of the highly ranked trees described sentences which were simply the addressee section of an email, e.g. *John*. We tried to improve the situation by filtering from the training material all those trees with a yield of less than six tokens and by introducing some randomness into the tree sorting process so that we were not relying completely on the parser accuracy predictor.

### 2.3 Genre Classifier

We use a 6-way classifier to choose between the 5 genre-specific grammars and the Ontonotes-WSJ baseline grammar. For simplicity, we represent documents with feature vectors listing the relative word frequency for any word that has a relative frequency higher than 0.0001 in at least one of data sets. Inspired by the cosine vector similarity used in information retrieval, we project all feature vectors to the unit sphere and then measure vector similarity with the Euclidian distance. As a classifier, we then use $k$-nearest neighbour ($k$-NN) with $k = 20$. To increase the number of training or reference items available to the $k$-NN model, we slice the training data into (overlapping) sub-documents of fixed size and number. Splitting the unlabelled genre data into training and development sets, we experiment with different values for size and number of training documents and size of test documents and find highest accuracy with 10,000 sub-documents per genre with 4,000 tokens each for training. In final application of the method to the data to be parsed, we use all unlabelled data to train the genre classifier. As this doubles the amount of training data, we cautiously increase the number of training documents to 15,000 and their size to 5,000 tokens.

Aiming for a whole document classification for the final blind test sets but also interested in classifier confidence, we also slice the test sets into overlapping sub-documents of 5,000 tokens each. The classifier achieves 100% confidence for blind A, C and D belonging to the answers, reviews and Ontonotes-WSJ genres respectively. For blind B, we have a near draw between newsgroups and weblogs. Given that the *a priori* probability of weblogs in the blind test sets is low as weblogs are already present in the development sets, we finally decide to use our newsgroup-specific grammars for blind B.[2]

## 3 *DCU-Paris13-1*

Our first system in the constituency track employs the product-of-PCFG-LA-grammars approach (Petrov, 2010) in conjunction with self-training (Huang et al., 2010) to obtain an accurate parsing system over out-of-domain genres.

The method works as follows: each edge in the chart is scored using the product of the scores of several grammars using the max-rule algorithm of Petrov and Klein (2007). The grammars are trained on the same dataset but, as PCFG-LA training uses the EM algorithm which does not guarantee a global likelihood optimum over the training set, different initialization setups will lead to different grammars. More precisely, the grammars have the same PCFG backbone but the weights associated with the annotated rules are different depending on the random deviation used when splitting annotations in two.

The baseline parsing system which is used to parse the unlabelled data is a product of eight grammars generated using five rounds of

[2]This could be formalised with the Bayesian decision rule.

split/merge/smooth (5th-order grammars). For each web genre, eight 6th-order grammars are trained on two copies of the Ontonotes-WSJ training material and 2.6 million tokens of the automatically parsed genre-specific treebanks. These self-trained grammars are then combined in a product model.[3]

The time it takes to train a PCFG-LA grammar[4] imposed a limit on the size of the training sets. Nonetheless, this is the best performing constituency parsing system. The Parseval f-scores for the five web genre development sets are shown in Table 2.

## 4 *DCU-Paris13-2*

Our previous work on parsing forum comments and tweets (Foster et al., 2011) suggests that the *DCU-Paris-1* system does not utilise enough of the unlabelled web data. Therefore, we decided to submit a second constituency parsing system using a parser that is quicker to train.

We train the first-stage Brown parser (Charniak, 2000; Charniak and Johnson, 2005) on web data parsed using the baseline parser (Ontonotes-WSJ-trained Lorg product model) with 10 target sizes from 1.3 to 13.1 million tokens and two copies of the Ontonotes-WSJ training data.

We parse each test set with the 10 trained grammars (of varying size) matching the genre predicted by our genre classifier. We implement an approximated product model based on the $n$-best output. For each parse tree present in at least one of the $n$-best lists, we multiply all observed parse probabilities, substituting a constant very low probability if a parse tree is absent in an $n$-best list. We use $n = 50$. This system performs well. However, despite seeing more data, it is systematically outperformed by our first system (see the second row of Table 2).

## 5 *DCU-Paris13-Dep*

Our third system combines the insights of individual constituency parsing models at the sub-sentential level using the following algorithm which is applied to dependency trees: each word in the sentence is attached to a parent word and assigned a dependency label according to a majority vote amongst the individual systems (Surdeanu and Manning, 2010) . Three constituency parsing systems are converted to dependencies using the Stanford converter (de Marneffe et al., 2006) and then combined using this voting algorithm. These systems are the following:

1. Our second constituency parsing system, *DCU-Paris13-2* (see Section 4)
2. A system which is the same as *DCU-Paris13-2* except that instead of using the genre-specific grammars, we use the backoff grammars listed in the lower half of Table 1
3. A system which is the same as *DCU-Paris13-2* except that it combines both genre-specific and backoff grammars

Thus, we can view this system as a combination of combinations since the individual systems that we combine are themselves combination systems. The voting algorithm is not guaranteed to produce a well-formed tree. In the rare case that an ill-formed tree is produced , we backoff to the *DCU-Paris13-2* system. The labelled attachment accuracy for this system over the five web genres are shown in the third row of Table 2.

| System | A | E | N | R | W |
|---|---|---|---|---|---|
| *1 (F1)* | 82.19 | 81.04 | 84.33 | 84.03 | 86.17 |
| *2 (F1)* | 79.62 | 80.79 | 82.65 | 82.52 | 85.38 |
| *Dep (LAS)* | 81.15 | 80.40 | 85.38 | 83.86 | 87.60 |

Table 2: Results for our three systems on the web genre development sets

## 6 Future Work

Our immediate next step is to perform an error analysis in order to better understand the strengths and weaknesses of the three systems, and based on this analysis, to try to improve the systems. Future work will also include the following: an analysis of the role of data normalisation, the use of the genre classifier to predict parsing models at the sub-document level, product model experiments involving larger training set sizes, sentence-level product model reranking experiments involving $n$-best lists $> 50$, and dependency voting experiments that combine both the Lorg and Brown systems.

[3]There is no hard limit on the number of grammars that our parser can use. We restricted ourselves to eight grammars mostly because of hardware resource limitations.

[4]Training $5 \times 8 = 40$ genre-specific grammars with six rounds and 2.6 million words of training material took 4.5 days on a cluster of six mixed machines with between 24 and 48 GB of RAM and 8 to 12 CPU cores.

## References

Mohammed Attia, Jennifer Foster, Deirdre Hogan, Joseph Le Roux, Lamia Tounsi, and Josef van Genabith. 2010. Handling unknown words in statistical latent-variable parsing models for arabic, english and french. In *Proceedings of SPMRL 2010*, pages 67–75. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Course-to-fine n-best-parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the ACL (ACL-05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the Annual Meeting of the North American Association for Computational Linguistics (NAACL-00)*, pages 132–139, Seattle, Washington.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. From News to Comment: Resources and benchmarks for parsing the language of Web 2.0. In *Proceedings of IJCNLP*.

Jennifer Foster. 2010. “cba to check the spelling” investigating parser performance on discussion forum posts. In *Proceedings of HLT NAACL*.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 12–22, Cambridge, MA, October. Association for Computational Linguistics.

Takuya Matsuzaki, Yusuke Miyao, and Jun’ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL ’05, pages 75–82, Stroudsburg, PA, USA. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, Sydney, Australia, July.

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Los Angeles, California, June. Association for Computational Linguistics.

Sujith Ravi, Kevin Knight, and Radu Soricut. 2008. Automatic prediction of parser accuracy. In *Proceedings of EMNLP*.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of HLT NAACL*.

Joachim Wagner, Djame Seddah, Jennifer Foster, and Josef van Genabith. 2007. C-structures and f-structures for the British National Corpus. In *Proceedings of LFG-07*, Stanford.