Parser Adaptation for Social Media by Integrating Normalization

van der Goot, Rob; van Noord, Gerardus

Parser Adaptation for Social Media by Integrating Normalization

Rob van der Goot
University of Groningen
r.van.der.goot@rug.nl

Gertjan van Noord
University of Groningen
g.j.m.van.noord@rug.nl

Abstract

This work explores normalization for parser adaptation. Traditionally, normalization is used as separate pre-processing step. We show that integrating the normalization model into the parsing algorithm is beneficial. This way, multiple normalization candidates can be leveraged, which improves parsing performance on social media. We test this hypothesis by modifying the Berkeley parser; out-of-the-box it achieves an F1 score of 66.52. Our integrated approach reaches a significant improvement with an F1 score of 67.36, while using the best normalization sequence results in an F1 score of only 66.94.

1 Introduction

The non-canonical language use on social media introduces many difficulties for existing NLP models. For some NLP tasks, there has already been an effort to annotate enough data to train models, e.g. named entity recognition (Baldwin et al., 2015), sentiment analysis (Nakov et al., 2016) and paraphrase detection (Xu et al., 2015). For parsing social media texts, such a resource is not available yet, although there are some small treebanks that can be used for development/testing purposes (Foster et al., 2011; Kong et al., 2014; Kaljahi et al., 2015; Daiber and van der Goot, 2016). To the best of our knowledge, the only treebank big enough to train a supervised parser for user generated content is the English Web Treebank (Petrov and McDonald, 2012). This treebank consists of constituency trees from five different web domains, not including the domain of social media.

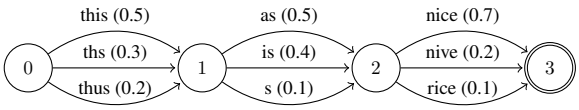

Figure 1: A possible output of the normalization model for the sentence ‘ths s nice’.

The magnitude of domain adaptation problems for the social media domain becomes clear when training the Berkeley parser on newswire text, and comparing its in-domain performance with performance on the Twitter domain. The Berkeley parser achieves an F1 score above 90 on newswire text (Petrov and Klein, 2007). An empirical experiment that we carried out on a Twitter treebank shows that the F1 score drops below 70 for this domain.

Annotating a new training treebank for this domain would not only be an expensive solution, the ever-changing nature of social media makes this approach less effective over time. We propose an approach in which we integrate normalization into the parsing model. The normalization model provides the parser with different normalization candidates for each word in the input sentence. Existing algorithms can then be used to find the optimal parse tree over this lattice (Bar-Hillel et al., 1961). A possible normalization lattice for the sentence ‘this is nice’ is shown in Figure 1. In this example output, the probability of ‘as’ is higher than the probability of ‘is’, whereas the most fluent word sequence would be ‘this is nice’. The parser can disambiguate this word graph because it has access to the syntactic context: ‘is’ is usually tagged as VBZ, while ‘as’ is mostly tagged as IN. This example shows the main motivation for using an integrated approach; the extra information from the normalization can be useful for parsing.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 491–497
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-2078

2 Related Work

SANCL 2012 hosted a shared task on parsing the English Web Treebank (EWT) (Petrov and McDonald, 2012). A wide variety of different approaches were used: ensemble parsers, product grammars, self/up-training, word clustering, genre classification and normalization. The teams that used normalization often used simple rule-based systems, and the actual effect of normalization on the final parser performance was not tested. Foster (2010) experiment with rule-based normalization on forum data in isolation and report a performance gain of 2% in F1 score.

A theoretical exploration of the effect of normalization on forum data is done by Kaljahi et al. (2015). They released the Foreebank, a treebank consisting of forum texts, annotated with normalization and constituency trees. They show that parsing manually normalized sentences results in a 2% increase of F1 score. Baldwin and Li (2015) evaluate the effect of different normalization actions on dependency parsing performance for the social media domain. They conclude that a variety of different normalization actions is useful for parsing.

A more practical exploration of the effect of normalization for the social media domain is done by Zhang et al. (2013). They test the effect of automatic normalization on dependency parsing by using automatically derived parse trees of the normalized sentences as reference. Other work that uses automatic normalization is Daiber and van der Goot (2016), which compare the effect of lexical normalization with machine translation on a manually annotated dependency treebank. All previous work uses only the best normalization sequence; errors in this pre-processing step are directly propagated to the parser.

For POS tagging, however, a joint approach is proposed by Li and Liu (2015). They use the n-best output of different normalization systems to generate a Viterbi encoding, based on all possible pairs of normalization candidates and their possible POS tags. Using this joint approach, they improve on both POS tagging and normalization.

3 Method

We first describe how an existing normalization model is modified for this specific use. Then we discuss how we integrate this normalization into the parsing model.

3.1 Normalization

We use an existing normalization model (van der Goot, 2016). This model generates candidates using the Aspell spell checker[1] and a word embeddings model trained on Twitter data (Godin et al., 2015). Features from this generation are complemented with n-gram probability features of canonical text (Brants and Franz, 2006) and the Twitter domain. A random forest classifier (Breiman, 2001) is exploited for the ranking of the generated candidates.

Van der Goot (2016) focused on finding the correct normalization candidate for erroneous tokens, gold error detection was assumed. Therefore, the model was trained only on the words that were normalized in the training data. Since we do not know in advance which words should be normalized, we can not use this model. Instead, we train the model on all words in the training data, including words that do not need normalization. Accordingly, we add the original token as a normalization candidate and add a binary feature to indicate this. These adaptations enable the model to learn which words should be normalized.

We compare the traditional approach of only using the best normalization sequence with an integrated approach, in which the parsing model has access to multiple normalization candidates for each word. Within the integrated approach, we compare normalizing only the words unknown to the parser against normalizing all words. We refer to these approaches as 'UNK' and 'ALL', respectively. Figure 1 shows a possible output when using ALL. When using UNK, the word 'nice' would not have any normalization candidates.

3.2 Parsing

We adapt the state-of-the-art PCFG Berkeley Parser (Petrov and Klein, 2007) to fit our needs. The main strength of this PCFG-LA parser is that it automatically learns to split constituents into finer categories during training, and thus learns a more refined grammar than a raw treebank grammar. It maintains efficiency by using a coarse-to-fine parsing setup. Unknown words are clustered by prefixes, suffixes, the presence of special characters or capitals and their position in the sentence.

Parsing word lattices is not a new problem. The parsing as intersection algorithm (Bar-Hillel et al., 1961) laid the theoretical background for ef-

[1] www.aspell.net

ficiently deriving the best parse tree of a word lattice given a context-free grammar. Previous work on parsing a word lattice in a PCFG-LA setup includes Constant et al. (2013), and Goldberg and Elhadad (2011) for the Berkeley Parser. However, these models do not support probabilities, which are naturally provided by the normalization in our setup. Another problem is the handling of word ambiguities, which is crucial in our model.

Our adaptations to the Berkeley Parser resemble the adaptations done by Goldberg and Elhadad (2011). In addition, we allow multiple words on the same position. For every POS tag in every position we only keep the highest scoring word. This suffices, since there is no syntactic ambiguity possible with only unary rules from POS tags to words, and therefore it is impossible for the lower scoring words to end up in the final parse tree.

To incorporate the probability from the normalization model (P_{norm}) into the chart, we combine it with the probability from the POS tag assigned by the built-in tagger of the Berkeley parser (P_{pos}) using the weighted harmonic mean (Rijsbergen, 1979):

$$P_{chart} = (1 + \beta^2) * \frac{P_{norm} * P_{pos}}{(\beta^2 * P_{norm}) + P_{pos}} \quad (1)$$

Here, β is the relative weight we give to the normalization and P_{chart} is the probability used in the parsing chart. We use this formula because it allows us to have a weighted average, in which we reward the model if both probabilities are more balanced.

4 Data

The normalization model we use is supervised, i.e. it needs annotated training data from the target domain. This is readily available for Twitter; we use 2,000 manually normalized Tweets from Li and Liu (2014) as training data.

We use the treebank from Foster et al. (2011) as develop and test data for our parser. It comprises 269 trees for developing and 250 trees for testing, all annotated using the annotation guidelines for the Penn Treebank (Bies et al., 1995) with some small adaptations for the Twitter domain (usernames, hashtags and urls are annotated as an NNP under an NP). For training, we use the English Web Treebank (EWT) concatenated with the standard training sections (2-21) of the Wall Street Journal (WSJ) part of the Penn treebank (Marcus et al., 1993).

Corpus	Sents	Words/ sent	Unk%
WSJ (2-21)	39,832	23.9	4.4
EWT	16,520	15.3	3.7
Foster et al. (2011)*	269	11.1	9.3
Li and Liu (2014)	2,577	15.7	14.1

Table 1: Some basic statistics for our training and development corpora. % of unknown words (Unk) calculated against the Aspell dictionary ignoring capitalization. *Only the development part.

Some basic statistics of our training and development data can be found in Table 1. Perhaps surprisingly, the percentage of unknown words in the EWT is lower than in the WSJ. This can be explained by the fact that the WSJ texts contains lots of jargon and named entities which are not present in the Aspell dictionary. The difference in percentage of unknown words between the normalization training data and the development treebank data might be an obstacle at first sight, but this can be overcome by tuning the weight (β) when combining the normalization and parse probabilities (Equation 1). Nevertheless, the effect of normalization will be smaller when there is less noise in the data.

5 Results

The parser is evaluated using the F1 score as implemented by EVALB[2]. All results in this section are averaged over 10 runs, using different seeds for the normalization model, unless mentioned otherwise.

The performance of our model depends on two parameters: the number of normalization candidates per word α and the weight given to the normalization β. We tuned these parameters on the development data using $\alpha \in$ [1-10] and $\beta \in [0.125, 0.25, 0.5, 1, 2, 4, 8, 16]$ to find the optimal values. The best performance is achieved using $\alpha = 6$ and $\beta = 2$. From this optimal setting, we will compare the effects of these variables for both the UNK and the ALL normalization strategies.

Figure 2 shows the effect of using different numbers of candidates and our baseline: the vanilla Berkeley parser. Using only the single best normalization sequence ($\alpha = 1$) we can obtain an improvement of 1.7% when normalizing all tokens. If we only normalize the unknown tokens

[2] nlp.cs.nyu.edu/evalb

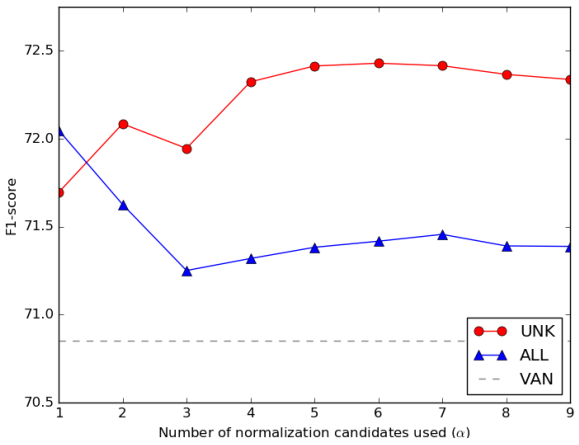

Figure 2: F1 scores on the development data when using multiple candidates while normalizing ALL words or only the UNKnown words ($beta = 2$), compared to a VANilla Berkeley parser.

the performance is slightly worse, but it still outperforms the baseline.

If we use more normalization candidates, performance increases; it converges around $\alpha = 6$. At this optimal setting, the baseline is outperformed by 2.2%. However, if more than only the first candidate is used, it is not beneficial to normalize all words anymore. This is probably an effect of creating too much distance between the original sentence and the normalization. The F1 score converges for higher number of candidates, because lower ranked candidates have very low normalization probabilities and are thus unlikely to affect the final parse.

The normalization model seldomly finds a correct candidate beyond $\alpha > 2$, at $\alpha = 2$ the recall for unknown words is 89.4% on the LexNorm corpus (Han and Baldwin, 2011), whereas the accuracy at $\alpha = 6$ is 91.7%. Perhaps surprisingly, the parser performance still improves when increasing α. Manual evaluation reveals that these improvements are obtained by using incorrect normalization candidates. Because these normalization candidates share some syntactic properties with the original word, they can still help in deriving a better syntactic parse. Figure 3 shows an example of this phenomenon; "Bono" is normalized to Bono's, and is therefore tagged as an NNS, even though this tag is still not correct, the head gets tagged correctly as NP. Combined with the normalization of "NOT", this results in a much better parse tree.

Table 2 shows the results using different

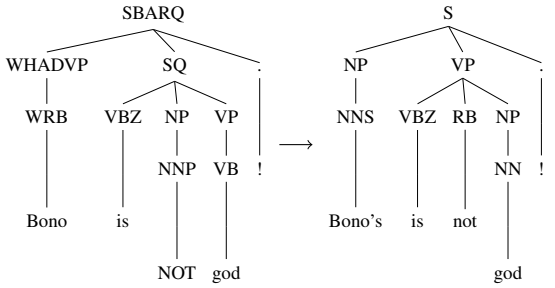

Figure 3: An example parse from the development corpus; left is the output of the vanilla Berkeley parser, right is the output with the integrated normalization.

weights. We compare the non-integrated approach ($\alpha = 1$) with the optimal number of candidates ($\alpha = 6$). The best results are achieved when β is 2, meaning that the normalization should get a higher weight than the POS tagger. The integrated model scores higher with almost all weights, the difference between ALL and UNK is similar as in Figure 2.

For the test data, we use the parameter settings that performed best on the development treebank (UNK, $\alpha = 6$, $\beta = 2$), and the best performing seed for the normalization model. The results on the test data are compared to the traditional approach of only using the best normalization sequence, the vanilla Berkeley parser, and the Stanford PCFG parser (Petrov and Klein, 2007) in Table 3. The integrated approach significantly outperforms the Berkeley parser as well as the traditional approach. It becomes apparent that the test part of the treebank is more difficult than the development part. Although the increase is smaller,

Cands(α)	1		6	
Weight(β)	UNK	ALL	UNK	ALL
0.125	70.79	71.12	70.88	69.45
0.25	70.79	71.12	70.99	62.97
0.5	70.86	71.18	71.21	70.51
1.0	71.19	71.52	71.78	71.08
2.0	**71.77**	**72.10**	**72.21**	**71.46**
4.0	71.73	72.01	72.02	71.43
8.0	71.12	71.33	71.69	71.26
16.0	70.29	70.32	70.50	70.09

Table 2: F1 scores on the development data using different weights, comparing only using the best candidate versus using 6 candidates.

Parser	dev	test
Stanford parser	66.05	61.95
Berkeley parser	70.85	66.52
Best norm. seq.	72.04	66.94
Integrated norm.	**72.77**	**67.36***
Gold POS tags	74.98	71.80

Table 3: F1 scores of our proposed models and previous work on the test set, trained on the EWT and WSJ. *Statistical significant against Berkeley-parser at $P < 0.01$ and at $P < 0.05$ against the best normalization sequence using a paired t-test.

normalization still improves parser performance. On the development set, 46% of the errors which can be accounted to mistakes made by the POS tagger are solved, whereas on the test set, we only solve 16% of this theoretical upper bound.

6 Discussion

The addition of multiple words on one position in the chart will probably lead to less pruning in the Berkeley parser, because more constituents in the tree will have a relatively high probability. To test if performance improvements are simply an effect of less pruning, we perform two additional experiments. Firstly, we use the vanilla Berkeley parser with lower pruning thresholds[3] on the Twitter development treebank. This results in a decrease in F1 score from 70.85 to 70.64, showing that our normalization model has a different effect. Secondly, we run our proposed parsing model on the standard development part of the more canonical WSJ data (section 24). The vanilla Berkeley parser achieves an F1 score of 89.15, whereas our best performing model scores 89.12 due to over-normalization. This shows that our model does not improve performance across all domains.

To test the effect of the normalization on the search space, we simply count the number of surviving constituents in the chart in the middle and final parse level. Results can be found in Table 4. There is a slight increase in the number of constituents when using normalization. A similar effect can be found for the parsing time; averaged over 10 runs, the vanilla Berkeley parser took 24.3 seconds on the development set, whereas our model took 24.5 seconds on the same machine.

Parse level	Berkeley	Integrated Norm.
3	756	765
6	3,086	3,115

Table 4: The average number of constituents in the chart per sentence for the middle parsing level (3) and the final level (6) on our development set.

7 Conclusion

We have shown that we can significantly improve the parsing of out-of domain data by using normalization. If we use normalization as a simple pre-processing step, we observe a small improvement in performance, while higher improvements can be achieved by using an integrated approach. Improvements in parsing performance are not only an effect of using correct normalization candidates, but are also due to wrong normalization candidates which share syntactic properties with the original word. Additionally, we show that when using only the best normalization sequence, it is better to normalize all words instead of only the unknown words. However, when using an integrated approach it is better to only consider unknown words for normalization.

Potential directions for future work include: allowing multiword replacements, normalization driven by the parsing model, and using lexicalized parsing so that the normalization candidates are used for more decisions in the parsing process than just assigning POS tags. To further improve the F1-score for the parsing of Tweets, complementary methods can be used: reranking, uptraining or ensembling parsers and grammars are some obvious next steps.

The source code of our experiments has been made publicly available [4].

Acknowledgements

We would like to thank our colleagues, especially Barbara Plank and Antonio Toral , and the anonymous reviewers for their valuable feedback. Furthermore we would like to thank Jennifer Foster for sharing the Twitter treebank. This work is part of the Parsing Algorithms for Uncertain Input project, funded by the Nuance Foundation.

[3] Tested by running the parser with `--accurate`. We also tried to tune the thresholds even further manually, but this had similar effects.

[4] https://bitbucket.org/robvanderg/berkeleygraph

References

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Beijing, China, pages 126–135. http://www.aclweb.org/anthology/W15-4319.

Tyler Baldwin and Yunyao Li. 2015. An in-depth analysis of the effect of text normalization in social media. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 420–429. http://www.aclweb.org/anthology/N15-1045.

Yehoshua Bar-Hillel, Micha Perles, and Eliahu Shamir. 1961. On formal properties of simple phrase structure grammars. *Sprachtypologie und Universalienforschung* 14:143–172.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Technical report, University of Pennsylvania.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1. Technical report, Google.

Leo Breiman. 2001. Random forests. *Machine learning* 45(1):5–32.

Matthieu Constant, Joseph Le Roux, and Anthony Sigogne. 2013. Combining compound recognition and pcfg-la parsing with word lattices and conditional random fields. *ACM Transactions on Speech and Language Processing (TSLP)* 10(3):8.

Joachim Daiber and Rob van der Goot. 2016. The denoised web treebank: Evaluating dependency parsing under noisy input conditions. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France. http://www.lrec-conf.org/proceedings/lrec2016/pdf/86_Paper.pdf.

Jennifer Foster. 2010. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 381–384. http://www.aclweb.org/anthology/N10-1060.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef Van Genabith. 2011. #hardtoparse: POS Tagging and parsing the Twitterverse. In *AAAI 2011 Workshop On Analyzing Microtext*. United States, pages 20–25.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia Lab @ ACL WNUT NER shared task: Named entity recognition for Twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Beijing, China, pages 146–153. http://www.aclweb.org/anthology/W15-4322.

Yoav Goldberg and Michael Elhadad. 2011. Joint hebrew segmentation and parsing using a pcfgla lattice parser. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 704–709. http://www.aclweb.org/anthology/P11-2124.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 368–378. http://www.aclweb.org/anthology/P11-1038.

Rasoul Kaljahi, Jennifer Foster, Johann Roturier, Corentin Ribeyre, Teresa Lynn, and Joseph Le Roux. 2015. Foreebank: Syntactic analysis of customer support forums. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1341–1347. http://aclweb.org/anthology/D15-1157.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for Tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1001–1012. http://www.aclweb.org/anthology/D14-1108.

Chen Li and Yang Liu. 2014. Improving text normalization via unsupervised model and discriminative reranking. In *Proceedings of the ACL 2014 Student Research Workshop*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 86–93. http://www.aclweb.org/anthology/P14-3012.

Chen Li and Yang Liu. 2015. Joint POS tagging and text normalization for informal text. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. pages 1263–1269. http://ijcai.org/Proceedings/15/Papers/182.pdf.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated

corpus of English: The Penn Treebank. *Computational linguistics* 19(2):313–330.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1–18. http://www.aclweb.org/anthology/S16-1001.

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*. Association for Computational Linguistics, Rochester, New York, pages 404–411. http://aclweb.org/anthology/N07-1051.

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)*. volume 59.

CJ Rijsbergen. 1979. *Information Retrieval*, volume 2. University of Glasgow.

Rob van der Goot. 2016. Normalizing social media texts by combining word embeddings and edit distances in a random forest regressor. In *Normalisation and Analysis of Social Media Texts (NormSoMe)*. http://www.let.rug.nl/rob/doc/normsome2016.pdf.

Wei Xu, Chris Callison-Burch, and Bill Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 1–11. http://www.aclweb.org/anthology/S15-2001.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. 2013. Adaptive parser-centric text normalization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1159–1168. http://www.aclweb.org/anthology/P13-1114.