How (not) to train a dependency parser: The curious case of jackknifing part-of-speech taggers

Željko Agić and **Natalie Schluter**
Department of Computer Science
IT University of Copenhagen
Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark
zeag@itu.dk nael@itu.dk

Abstract

In dependency parsing, jackknifing taggers is indiscriminately used as a simple adaptation strategy. Here, we empirically evaluate when and how (not) to use jackknifing in parsing. On 26 languages, we reveal a preference that conflicts with, and surpasses the ubiquitous ten-folding. We show no clear benefits of tagging the training data in cross-lingual parsing.

1 Introduction

Dependency parsers are trained over manually annotated treebank data. By contrast, when applied in the real world, they parse over sequences of predicted parts of speech. As POS tagging accuracy drops due to domain change, the parsing quality declines proportionally. Bringing these two POS tag sources closer together thus makes for a reasonable adaptation strategy.

Arguably the simplest of such adaptations is n-fold **jackknifing**. In it, a treebank is divided into n equal parts, and the n-th part is POS-tagged with a tagger trained on the remainder. The procedure is repeated until all n parts are assigned with predicted POS tags. A parser is then trained over the thus altered treebank, under the assumption that its POS features will now more closely resemble those of the input data.

Jackknifing is simplistic as it i) has a very limited adaptation range for $n \in \mathbb{N}^+$, and it ii) does not in any way take the input data into account, other than through a vague assumption of an undefined amount of tagging noise in the input. As such, it exhibits very mixed results. Still, the method is now ubiquitous in the parsing literature.

In Figure 1, we survey the ACL Anthology[1] for POS jackknifing. We uncover that ~80% of the 70 parsing papers we retrieved make use of **ten-fold** jackknifing. This choice spans across the various languages and domains parsed in these papers, and is even motivated by simply "following the traditions in literature".[2]

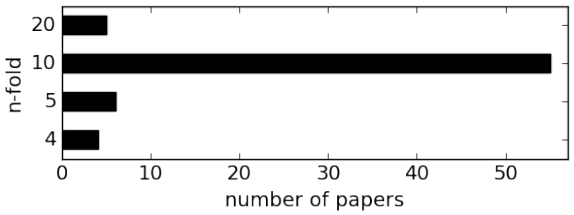

Figure 1: Jackknifing in the ACL Anthology. Distribution of n over 70 parsing papers that use tagger n-folding.

Our contributions. We evaluate jackknifing to establish whether its use is warranted in dependency parsing. Controlling for tagging quality in training and testing, we experiment with monolingual and delexicalized cross-lingual parsers over 26 languages, showing that:

i) Indiscriminate use of ten-fold jackknifing results in sub-optimal parsing.
ii) Tagging the training data does not yield clear benefits in realistic cross-lingual parsing.
iii) Our jackknifing extension improves parsing through finer-grained adaptation.

2 Method

Jackknifing generally refers to a leave-one-out procedure for reducing bias in parameter estimation from an unbiased sample (Quenouille, 1956; Tukey, 1958). More recently, in machine learning the term is used synonymously with "cross-

[1] http://aclweb.org/anthology/

[2] Incidentally, using ten-folds with the WSJ data yields roughly the same train- and test-set tagging accuracy, and seems to be where the choice originated.

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 679–684
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-2107

validation" for estimation of predictive model performance measures. In NLP, jackknifing has commonly been used to describe a procedure by which the training input is adjusted to correspond more closely to the expected test input, and it is in this latter sense that we use the term here.

In particular, in parsing research, the ***n*-fold jackknifing** proceeds as follows. The treebank is first partitioned into n non-overlapping subsets of equal size. Then, iteratively, each part acts as a *test subset* and is tagged using a model induced by the remaining $n-1$ parts, the *training subset*, until the entire treebank is tagged.

We want to control for POS tagging accuracy through the jackknifing method. To do this, we train a tagger on increasing sized subsets of the training set. In fold terminology, this corresponds to dividing the training set into equal parts of size $\frac{1}{n}$, training on $\frac{n-1}{n}$ths of the training set and testing on the remaining $\frac{1}{n}$th. However, this constrains the size of the training subset to be larger than half the original data, and thus concentrates our study on models that use almost all the data, since the non-linear curve $f(n) = \frac{n-1}{n}$ becomes very flat very fast. Thus, varying fold numbers reveals very little variation in terms of POS tagging accuracy in the lower accuracy range.

Linear extension. We now propose a simple extension of the jackknifing paradigm to study parser accuracy given a percentage p of the training set: **linear jackknifing**.

Let $p \in (0, 1)$ be the percentage of the randomly shuffled training set D used to induce a model to tag some remaining number of instances. A training subset of this size allows a test subset of size at most $\lceil |D| \cdot (1-p) \rceil$. Given a test subset to tag, we can induce a model from a random subset of the remaining examples of size approximately $p \cdot |D|$ to become our training subset. We randomize the choice of examples in the training subset to avoid introducing bias. In order to tag all of D, the minimum number of models we need to generate is $\lceil 1/(1-p) \rceil$. We thus separate D into test subsets $f_1, \ldots, f_{\lceil 1/(1-p) \rceil}$ each of size approximately $\lceil |D| \cdot (1-p) \rceil$. For each f_i, we randomly sample a training subset of size approximately $\lceil p \cdot |D| \rceil$ from the remainder of D, induce a model and then tag f_i. This results in the original full training set tagged with an accuracy corresponding to the performance of a randomly selected tagger trained on approximately $p \cdot |D|$ of the examples.

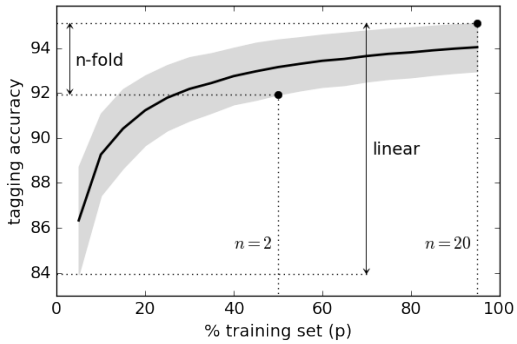

Figure 2: Tagger learning curve for 26 languages: mean tagging accuracy with 95% confidence intervals. Accuracy ranges for n-fold and linear jackknifing are indicated.

Intrinsic evaluation. For increasing values of p, at 5% increments, we carried out linear jackknifing on 26 languages. For each p, we averaged the performance of the induced taggers on the respective gold standards. Figure 2 illustrates the difference in informativeness of the two approaches, where each tagging accuracy score is averaged across the 26 languages. We see that with n-fold jackknifing, tagging accuracy is constrained to between approximately 92% and 95%, whereas linear jackknifing explores accuracies as low as approximately 86%. Moreover, the confidence intervals are consistent across the p, demonstrating unbiased tagging models generated on less data (lower p). We now show that these smaller levels of p are essential for good parser performance in some cases of jackknifing.

3 Experiments

Our experiment aims at judging the adequacy of jackknifing in dependency parsing. First, we outline the experiment setup, where we conduct two sets of experiments:

i) **monolingual**, where lexicalized parsers are trained on treebanks for their respective languages, and
ii) **cross-lingual**, that features **SINGLE**-best and **MULTI**-source delexicalized parsers.

Tagging sources. By jackknifing we explore how the mismatch between training and test POS affects parsing. Our setup thus critically relies on the sources of tags. We tag our test sets using:

i) **PRED**, the monolingual taggers, and
ii) **PROJ**, the low-resource taggers by Agić et al. (2016), based on annotation projection.

Monolingual parsing												
Train:	GOLD		linear jackknifing				n-fold jackknifing				PROJ	
Test:	PRED	PROJ	PRED	$p_{\max}$	PROJ	$p_{\max}$	PRED	$n_{\max}$	PROJ	$n_{\max}$	PRED	PROJ
Arabic (ar)	79.4	51.4	**79.5**	90	64.0	5	**79.5**	12	55.3	2	73.5	**74.4**
Bulgarian (bg)	86.8	56.4	**87.2**	80	66.7	5	**87.2**	12	60.2	2	82.2	**78.0**
Czech (cs)	81.0	58.2	**81.6**	85	62.5	5	**81.6**	8	60.1	2	77.3	**70.7**
Danish (da)	74.4	65.7	78.2	95	71.8	5	**78.3**	11	68.5	2	75.1	**76.0**
German (de)	79.0	51.8	**80.2**	20	56.6	5	80.1	2	54.4	7	75.6	**69.2**
Greek (el)	81.1	59.4	**81.4**	70	64.7	5	**81.4**	20	61.2	2	74.7	**75.6**
English (en)	80.9	71.6	**82.3**	80	77.6	5	**82.3**	17	76.0	2	80.2	**80.9**
Spanish (es)	80.7	75.4	**82.1**	20	78.3	5	81.8	9	77.3	4	80.2	**80.2**
Estonian (et)	74.3	61.6	76.0	55	67.1	5	**76.1**	13	65.0	2	73.3	**70.5**
Persian (fa)	82.3	25.8	**83.2**	35	46.3	5	83.1	4	35.1	2	66.3	**71.9**
Finnish (fi)	72.0	53.9	72.9	75	60.4	5	**73.0**	18	56.6	2	69.2	**64.7**
French (fr)	80.9	65.4	81.7	90	72.7	5	**81.8**	19	68.7	2	79.2	**77.1**
Hebrew (he)	80.6	54.2	**81.5**	75	68.1	5	**81.5**	7	61.7	2	77.9	**75.7**
Hindi (hi)	89.1	51.0	**91.0**	70	70.1	5	**91.0**	11	63.1	2	84.8	**85.5**
Croatian (hr)	78.1	54.6	78.2	45	62.9	5	**78.4**	19	56.3	20	73.6	**72.5**
Hungarian (hu)	74.3	55.4	74.9	95	63.2	5	**75.2**	17	58.3	2	71.0	**66.8**
Indonesian (id)	79.4	70.6	**79.6**	90	74.9	5	**79.6**	9	72.3	2	77.8	**77.7**
Italian (it)	86.2	76.2	**87.6**	55	82.9	5	**87.6**	2	80.6	2	86.2	**85.5**
Dutch (nl)	72.1	60.2	**73.8**	50	67.9	5	**73.8**	12	65.6	8	67.6	**72.6**
Norwegian (no)	83.7	74.8	**85.3**	95	79.9	5	**85.3**	8	78.0	2	83.7	**82.6**
Polish (pl)	83.7	69.8	84.7	30	75.8	5	**84.9**	20	73.0	2	81.4	**78.6**
Portuguese (pt)	82.2	75.7	**83.2**	30	78.8	5	83.1	2	77.7	19	80.2	**82.0**
Romanian (ro)	81.8	66.8	**82.5**	85	72.9	5	**82.5**	5	69.0	2	79.7	**79.1**
Slovene (sl)	82.0	62.5	**84.2**	55	71.4	5	**84.2**	16	68.9	7	79.3	**77.4**
Swedish (sv)	80.9	74.4	81.9	90	77.3	10	**82.0**	19	76.8	7	80.0	**79.1**
Tamil (ta)	65.1	33.4	**65.8**	95	49.5	5	65.6	5	42.5	2	51.5	**56.3**
Mean	79.7	60.6	80.8	67.5	68.6	5.2	80.8	11.4	64.7	4.2	76.2	75.4
Best for #/26	0	0	18	–	0	–	21	–	0	–	0	26

Table 1: Parsing accuracy (UAS) in relation to the underlying sources of POS tags in training and at runtime. **Bold:** best result for language, separately for PRED and PROJ test sets.

We do not experiment with gold POS tags in the test sets. Instead, we only focus on realistic parsing over predicted tags. The tags in our training sets can be **GOLD**, PROJ, or they can be predicted through n-fold or linear jackknifing.

In n-fold jackknifing, we experiment with $n \in \{2, 3, .., 20\}$, while for the linear extension we set $p \in \{5, 10, ..., 95\}$. We report the average parsing scores over 5 runs for each n and p so as to mitigate the effects of random shuffling in the two jackknifing procedures. In finding the optimal values of the parameters $n_{\max}$ and $p_{\max}$, we report the highest values in case of ties. For example, if $n = 5$ and $n = 10$ both yield the same maximum UAS, we set $n_{\max} = 10$.

We emphasize the importance of realistic settings especially in cross-lingual parsing. Thus, we commit to using PROJ taggers with an outlook on true low-resource languages.

Data. We use the Universal Dependencies (UD) treebanks version 1.2 (Nivre et al., 2016).[3] As the projection-based taggers are trained on the WTC dataset by Agić et al. (2016), we intersect the list of WTC languages with the UD list for a total of 26 languages.

Tagging and parsing. For POS tagging, we use the TNT tagger by Brants (2000). The PRED taggers score 94.1±1.1%, while the low-resource PROJ taggers are on average 71.7±5.7% accurate. We experiment with two parsers. Bohnet's (2010)

[3] http://universaldependencies.org/

Delexicalized transfer						
Train:	GOLD⇝PROJ			PROJ⇝PROJ		
Test:	MULTI	SINGLE		MULTI	SINGLE	
Arabic (ar)	**34.2**	he	**38.3**	28.3	id	37.0
Bulgarian (bg)	49.5	cs	50.1	**50.2**	cs	**51.1**
Czech (cs)	**50.4**	sl	50.7	48.4	sl	**50.8**
Danish (da)	58.0	no	58.5	**58.1**	no	**61.4**
German (de)	43.9	no	45.0	**45.8**	sv	**45.4**
Greek (el)	56.3	it	**55.4**	**57.3**	no	55.3
English (en)	55.8	no	56.7	**57.2**	sv	**58.2**
Spanish (es)	**67.9**	it	**68.5**	64.9	it	67.6
Estonian (et)	**45.8**	fi	**53.1**	43.9	fi	50.8
Persian (fa)	**21.5**	ar	**25.7**	18.9	pl	24.2
Finnish (fi)	38.8	et	45.1	**40.0**	et	**45.5**
French (fr)	52.8	it	54.4	**54.9**	it	**58.9**
Hebrew (he)	**44.6**	ro	**45.1**	41.7	ro	44.0
Hindi (hi)	16.9	ta	**38.1**	**18.0**	ta	31.0
Croatian (hr)	**50.9**	sl	**50.8**	46.8	cs	49.3
Hungarian (hu)	39.9	sv	46.4	**40.1**	et	**49.3**
Indonesian (id)	**54.5**	ro	**56.6**	48.7	ro	54.4
Italian (it)	67.0	es	67.8	**67.4**	es	**69.2**
Dutch (nl)	**55.1**	es	**53.9**	52.2	sv	52.5
Norwegian (no)	**63.5**	sv	64.3	62.4	sv	**64.4**
Polish (pl)	**62.9**	cs	**64.2**	57.3	cs	59.2
Portuguese (pt)	**65.7**	es	**67.7**	64.7	it	66.9
Romanian (ro)	**53.6**	it	53.7	50.5	es	**53.9**
Slovene (sl)	50.5	cs	53.4	**52.6**	cs	**56.3**
Swedish (sv)	**62.7**	no	66.8	61.9	no	**67.1**
Tamil (ta)	21.2	hu	28.9	**24.6**	hi	**33.8**
Mean	**49.4**	–	**52.3**	48.3	–	52.2
Best for #/26	14	–	12	12	–	14

Table 2: UAS scores for the delexicalized transfer parsers. TRAIN⇝TEST indicates the training and testing POS. **Bold:** best result for language, separate for MULTI and SINGLE transfer. For SINGLE, best source names are also reported.

second-order graph-based system MATE[4] is the primary. Further, we verify all parsing results by using a transition-based parser YARA[5] with dynamic oracles (Rasooli and Tetreault, 2015).

The following CoNLL 2009 features are used for training the parsers:[6] ID, FORM (in monolingual parsing only), POS, and HEAD. Since ours is not a benchmarking effort, we apply all systems with their default settings.

3.1 Results

In **monolingual parsing** over PRED tags (Table 1), we achieve an identical average UAS with linear and n-fold jackknifing. Our adaptations surpass training with GOLD data by +1.1 UAS. Linear jackknifing improves over GOLD training by +8.1 UAS when parsing over low-resource PROJ tags. There, we top GOLD training by n-fold jackknifing as well, but it trails the linear variant by -3.9 UAS. In the low-resource PROJ setup, PROJ-trained parsers are dominant. They score +6.8 UAS over linear, +10.7 UAS over n-folding, and +14.8 UAS over GOLD training.

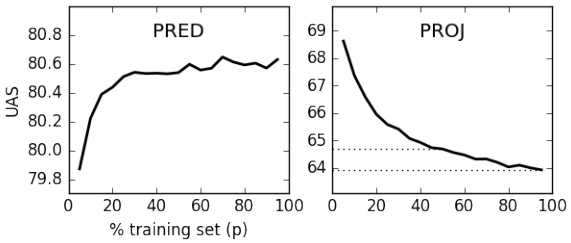

Figure 3: Parsing accuracy (UAS) in relation to linear jackknifing over 26 languages, with two sources of test set POS tags.

Figure 3 plots the relation between the sample size p in linear jackknifing and the resulting UAS in parsing, split for PRED and PROJ test-set taggings. Parsing over PRED tags, the UAS generally increases with p, but we note that this increase is rather small: over 26 languages, moving p from 5% to 95% yields only +0.7 UAS on average. By contrast, adapting to the lower-quality PROJ tags sees a larger +5 UAS benefit from decreasing p all the way to 5%, which is well outside the n-fold range, as indicated for $n \in \{2, ..., 20\}$ by the dotted lines in the figure.

Our **cross-lingual parsing** experiment (Table 2) contrasts two options: we either tag (PROJ⇝) or do not tag (GOLD⇝) the parser training data. To reflect realistic low-resource parsing, the test data is tagged with PROJ taggers only. On average, the unadapted parsers are slightly better (UAS: +1.1 MULTI, +0.1 SINGLE). However, they are almost evenly split with the adapted ones in terms of offering the best performance for 12-14 out of the 26 test languages each. These results suggest, at least for simplicity, a preference for not tagging the treebanks.

4 Discussion

Linear or n-fold? In resource-rich PRED parsing, the two jackknifing methods are evenly split, with identical average UAS score and an overlap

[4] https://code.google.com/archive/p/mate-tools/

[5] https://github.com/yahoo/YaraParser

[6] https://ufal.mff.cuni.cz/conll2009-st/

on 13 languages. In low-resource PROJ parsing, n-folding falls far behind as the constraint for $n \geq 2$ prevents it from adapting accordingly. The median $p_{\max}$ in PRED and PROJ are 75% and 5%. The first one roughly corresponds to 4-fold jackknifing, while the second one is far below the two-fold range. The median $n_{\max}$ are 11 and 2, and we note that $n_{\max}$ is rarely ~10 in Table 1.

If we simply use ten-fold jackknifing for PRED tags, we manage to match the $p_{\max}$ scores for only 9 of 26 languages, and we score -0.2 UAS on average. If using $n = 10$ with PROJ tags, the disconnect is much more substantial, and we are unable to reach $p_{\max}$ (-4.6 UAS).

The GOLD training data is *never* the best choice in our monolingual parsing experiment, regardless of whether the test tags are PRED or PROJ. This result in itself justifies the usage of jackknifing as adaptation for the monolingual setting, provided that it is not indiscriminate.

Finding $\hat{p}_{\max}$. For choosing the optimal linear jackknifing in real-world parsing, we note that $p_{\max}$ closely correlates with test set tagging accuracy (Spearman's $\rho = 0.76$), and negatively with treebank size ($\rho = -0.42$, for $|D| \leq 10$k sentences). Thus, to adapt via linear jackknifing, we must i) approximate the expected input data tagging accuracy, while at the same time ii) accounting for the fact that the accuracy associated with any p depends on treebank size as well. In other words, given two treebanks $|D_1| < |D_2|$, we would typically need $p_1 > p_2$ with the goal of matching the same test-set accuracy.

The other parser. Replacing the MATE parser with the transition-based YARA does not change the outcome of our monolingual parsing experiment, save for the average 0.58–1.65 drop in UAS. On the other hand, in cross-lingual parsing, YARA highlights the benefits of *not* tagging the training data, as the GOLD⇝PROJ parsers are there the best choice for parsing 17/26 languages. On average, we see +2.1 UAS for MULTI, and +0.7 for SINGLE over PROJ⇝PROJ. This is especially worth noting since large-scale parsing generally favors transition-based systems.

GOLD test tags. Thus far, we have shown the need for more careful jackknifing in parser training with respect to the expected tagging quality at parse time. Fixing $n = 10$ was suboptimal in parsing over the fully supervised PRED tags, while $n = 2, 10$ were way below the threshold in low-resource parsing over our cross-lingual PROJ tags. We have purposely excluded GOLD test tags from the discussion so far.

Still, while parsing over GOLD POS input does not hold much significance for real-world applications, it is worth noting how jackknifing performs in the upper limit: trying to reach the accuracy of parsers trained and tested on GOLD tags. In that particular setup, we observe the maximum UAS of 83.8 for median $n_{\max} = 12$ and $p_{\max} = 80\%$. The respective modal values are $n = 20$ and $p = 95\%$, meaning that for most languages, we come closest to GOLD⇝GOLD by maximizing the tagging accuracy. The overall score amounts to -0.7 UAS below the upper bound.

5 Related work

Jackknifing itself is for the most part incidental to the work that employs it. Here, we mention a few notable exceptions.

Che et al. (2012) compare jackknifing to using gold tags in parsing Chinese for constituents and dependencies, where they observe mixed results: improvement with one parser, and decrease with the other. Seeker and Kuhn (2013) briefly touch upon the importance of jackknifing in bridging the gap between training and test data, and experiment with 5- and 10-folds. Honnibal and Johnson (2015) passingly contrast jackknifing to joint learning, giving precedence to the latter for simplicity. Finally, Kong et al. (2015) follow Zhu et al. (2013) in ten-folding for Chinese and English, citing 2.0% and 0.4% improvements. Incidentally, jackknifing parsers then hurts their performance in tree conversions.

6 Conclusions

The parsing literature is riddled with indiscriminate use of n-fold part-of-speech tagger jackknifing as makeshift domain adaptation.

In this paper we have proposed a careful empirical treatment of jackknifing in dependency parsing, far surpassing ten-folding via fine-grained control over the data adjustment.

Acknowledgements. We thank Barbara Plank and Djamé Seddah for their valuable comments on an earlier version of this paper. We appreciate the incisive feedback by the anonymous reviewers. Finally, we acknowledge the NVIDIA Corporation for supporting our research.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association of Computational Linguistics* 4:301–312. http://aclweb.org/anthology/Q16-1022.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, pages 89–97. http://aclweb.org/anthology/C10-1011.

Thorsten Brants. 2000. Tnt - a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference*. http://aclweb.org/anthology/A00-1031.

Wanxiang Che, Valentin Spitkovsky, and Ting Liu. 2012. A comparison of chinese parsers for stanford dependencies. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 11–16. http://aclweb.org/anthology/P12-2003.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1373–1378. https://doi.org/10.18653/v1/D15-1162.

Lingpeng Kong, M. Alexander Rush, and A. Noah Smith. 2015. Transforming dependencies into phrase structures. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 788–798. https://doi.org/10.3115/v1/N15-1080.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France, pages 1659–1666.

Maurice H. Quenouille. 1956. Notes on Bias in Estimation. *Biometrika* 61:1–17. https://doi.org/10.2307/2332914.

Mohammad Sadegh Rasooli and Joel Tetreault. 2015. Yara Parser: A Fast and Accurate Dependency Parser. *arXiv preprint arXiv:1503.06733* .

Wolfgang Seeker and Jonas Kuhn. 2013. The effects of syntactic features in automatic prediction of morphology. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 333–344. http://aclweb.org/anthology/D13-1033.

John W. Tukey. 1958. Bias and Confidence in Not Quite Large Samples. *Annals of Mathematical Statistics* 29:614. https://doi.org/10.1214/aoms/1177706647.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 434–443. http://aclweb.org/anthology/P13-1043.