What I think when I think about treebanks

Anders Søgaard
Dpt. of Computer Science
University of Copenhagen
soegaard@di.ku.dk

Abstract

In this opinion piece, I present four somewhat controversial suggestions for the design of future treebanks: a) Treebanks should be based on *adversarial* samples, rather than pseudo-representative samples. b) Treebanks should include *multiple* splits of the data, rather than just a single split, as in most treebanks today. c) They should include multiple annotations of each sentence, whenever possible, instead of adjudicated annotations. d) There is no real motivation for adhering to a notion of well-formedness, since we now have parsers based on deep learning that generalize easily and perform well on any type of graphs, and treebanks therefore do not have to limit themselves to trees or directed acyclic graphs.

1 Introduction

Treebanks are some of the most ambitious and expensive resources the NLP community has produced, and over the last two decades, they have enabled us to push research horizons and develop more advanced technologies. While treebanks remain invaluable, and we owe thanks to all the people who have contributed to existing ones, I nevertheless think future treebanks will have significantly more value if they are designed slightly differently.

Treebanks are supposed to enable us to induce syntactic parsers and estimate their performance on held-out, unseen data; that is, their performance in the wild. We have treebanks for more than 50 languages, albeit some of them very small, but typically only one treebank for each language. Reported parsing results vary quite a bit from treebank to treebank, and such results are often taken as indicative of our ability to parse the relevant languages in the wild, given our current linguistic resources.

Differences in parsing results, from treebanks to treebanks, are often explained on linguistic grounds. Here is an example from the shared task description paper from the CoNLL 2007 dependency parsing shared task (Nivre et al., 2007):[1]

> *. . . the languages involved in the multilingual track this year can be more easily separated into three classes with respect to top scores:*
>
> - *Low (76.31–76.94): Arabic, Basque, Greek*
> - *Medium (79.19–80.21): Czech, Hungarian, Turkish*
> - *High (84.40–89.61): Catalan, Chinese, English, Italian*
>
> *It is interesting to see that the classes are more easily definable via language characteristics than via characteristics of the data sets. [. . .] The most difficult languages are those that combine a relatively free word order with a high degree of inflection.*

[1]This shared task is a predecessor to the CoNLL 2017 dependency parsing shared task which included many more languages, but the survey paper produced by the organizers did *not* provide a similar explanation for differences in performance.

Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories (TLT16), pages 161–166,
Prague, Czech Republic, January 23–24, 2018.

Keywords: treebanks, design principles, generalization

While explaining parsing performance by language characteristics such as freeness of word order and morphological complexity, is quite intuitive, and pleasing for someone with a background in linguistic typology, like me; this explanation was later disputed in Søgaard and Haulrich (2010), who claimed that what they called *derivational perplexity*, i.e., (a particular way of measuring) the average complexity of the tree structures in the tree structures, explained performance differences better. Linguistic properties may of course be confounds, but tree structures are influenced heavily by linguistic theory and annotation guidelines, which also determine the complexity of the structures we use to analyze sentences. Other factors that influence performance include how homogeneous the underlying corpus is. Is the text single-authored? If multi-authored, were the texts written with different intended audiences? Do the authors span multiple demographics? Do they speak different dialects? Etc.

Nevertheless, explanations based on linguistic grounds are far more common, and we often hear claims based on treebank results that some languages are harder to parse than others. This was also the motivation behind a recent shared task in parsing morphologically rich languages.[2] You may think that linguistic differences are much more important than other factors. This is not always true, however. Foster et al. (2010), for example, evaluate a dependency parser trained on newswire (the English Penn Treebank) on hand-annotated Twitter data. On held-out newswire data, the parser has an unlabeled attachment score of 90.6%, but on Twitter data, the score is 73.6%. This 19% relative drop (17% absolute) is bigger than a lot of the drops we observe transferring models across languages. About the same time, Foster et al. (2010) ran the Twitter experiments, McDonald et al. (2011) revisited the idea of transferring the non-lexical part of parsing models across languages. Their impoverished English model scored 82.5% in unlabeled attachment score on English data. When evaluating their model on Portuguese, for example, scores dropped to 68.4%, which is a 17% relative drop (14% absolute). Changing the domain hurts model performance more than changing the language in this case. It is easy to find similar examples in the literature. These factors, by the same token, also influence how well we can generalize from validation and test set performance to *performance in the wild* or practical usefulness. This paper discusses different dimensions of this problem and proposes design principles for building treebanks in the future. My main observation is that our treebanks are being too nice on us, i.e., leading us to overestimate our performance in the wild.

2 Sampling for Treebanks

No one has to the best of my knowledge ever claimed that the sentences in the English Penn Treebank[3] were representative of the English language. All the sentences were written by Wall Street Journal journalists in the 1980s, who were trained to write in a particular way and asked to write about particular topics, of interest to the readers of the newspaper. The first version of the Slovene Dependency Treebank[4], used in the CoNLL 2006 Shared Task, was the annotation of a single novel, written by a single author. Clearly, a single piece of prose is not representative in any way of a language. The sentences in the Croatian Dependency Treebank[5] come from the newspaper Croatia Weekly. Other treebanks claim to be based on more representative language samples. The Danish Dependency Treebank[6] and the Turku Dependency Treebank,[7] for example, contain sentences from newswire, magazines, blogs, and literature.

In all the above treebanks (English, Slovene, Croatian, Danish, and Finnish), however, the training and test sentences come from the same sources, and while treebanks will never be i.i.d., there is a clear intention to sample training and test sentences in near-identical ways; I return to this in §3. Approximating i.i.d. makes sense if you can sample representative sentences at random. I argue that we can't, and that we therefore need to abandon the ideal of nearly identically sampled train and test portions in our treebanks. The argument relies on the observation that we cannot sample randomly from language.

[2] http://www.spmrl.org/
[3] LDC95T7
[4] http://nl.ijs.si/sdt/
[5] http://hobs.ffzg.hr/en/
[6] https://github.com/UniversalDependencies/UD_Danish
[7] http://bionlp.utu.fi/fintreebank.html

2.1 How can we sample from language?

Imagine you were to design an English treebank from scratch. You had a team of trained annotators, ready to annotate, just waiting for you to send them raw text files they can decorate with linguistic analyses. Now the question is what texts you send them.

How would you sample your sentences? Where would you go? A knee jerk reaction may be to say that you would sample them from a representative corpus, but that is putting your eggs in someone else's basket, relying on their ability to sample from English. What would you do? Would you get your sentences from the Wall Street Journal? Probably not exclusively. What else, then? Sentences from literary works? From Harry Potter? How about comics, then? How about 19th century literary works? From Facebook? Google Search queries? Speech logs? Learner data? Dialect? Expat English? Or how about the language of neurodiverse speakers of English? You probably feel you get my point, but stop for a minute and think about it. What *would* you do?

You may arrive at the conclusion that treebanks should be domain-specific. *Okay, so English comes in a lot of different flavors. Let us just build treebanks for each one of them.* This approach has three problems: a) It is not possible to enumerate the number of domains. The concept of *domains* is usually ambiguous between *topics* and *platforms/media/registers*, but it should be clear that both the list of topics covered in human history, and the list of platforms available to us, are growing and unbounded. Also, b) authors have different linguistic traits, and c) language is constantly changing.

The next conclusion – that it is simply *not possible to sample from language* – means sample bias is inevitable, inescapable, and something you have to embrace. This is, in my view, extremely important, and many things follow from this observation:

- A single test set is simply not gonna cut it.
- Your multiple test sets must be very different, with varying degrees of bias.
- Even so, you are likely to still overestimate performance on unseen data.

In other words, my first advise to designers of future treebanks is to include *multiple* test datasets and to make them as different as possible. To the best of my knowledge, no treebanks are designed that way. A few treebanks, such as OntoNotes,[8] contain meta-data that allow you to easily set up experiments with multiple test data sets, though.

3 Cutting the Cake Unfairly Again and Again

One way to achieve better data points for estimating performance in the wild with our current treebanks is by introducing multiple, more or less adversarial training-test splits. Here is a couple of ideas:

Splits based on meta-data Some treebanks, including for example the English Penn Treebank and the Danish Dependency Treebank, contain meta-information about where each sentence is from, and when it was written or published. Such data enables us to estimate cross-domain robustness, or how performance drops over time.

Splits based on divergence In the domain adaptation literature, divergence measures such as Jensen-Shannon divergence or $\mathcal{A}$-distance are often used to quantify the similarity of domains (Ben-David et al., 2007). Such measures can be used to construct splits of varying difficulty. The more such data points, the better we can estimate performance on future samples.

Splits based on sentence length In the recurrent neural network literature , as well as in unsupervised dependency parsing, it is customary to evaluate the ability of a model to generalize from short to long strings (Chalup and Blair; Spitkovsky et al., 2009). This is another interesting set of splits of a treebank. What is our performance on sentences of length $> n$ when training on sentences of length $\leq n$?

[8] LDC2013T19

My second advise to designers of future treebanks is thus to devise *alternative splits*. Again, if a treebank contains rich meta-data, such as Ontonotes, we can easily set up such splits. However, a list of standard splits would ensure comparability across the work of different research groups.

4 Learning from Disagreements

One thing that makes parsing harder than necessary, is our insisting on perfect agreement with the human gold-standard annotation. There are often multiple possible analyses of a sentence, even when wholeheartedly adopting a particular linguistic theory, but parsers are only rewarded for picking the analysis accepted by the treebank annotators after adjudication (Plank et al., 2014b).[9]

In unsupervised dependency parsing, some researchers have proposed alternative, less conservative metrics that would not penalize linguistically acceptable deviation from the gold standard (Schwartz et al., 2011; Tsarfaty et al., 2012). An alternative, however, is to use multiple reference annotations for each sentence in the test data. This is the approach taken in machine translation, for example.

I am confident that performance across multiple reference annotations (multiply annotated test data) is more predictive for downstream performance than performance on adjudicated annotations. Moreover, we already know that dependency parsers benefit from observing disagreements between annotators at training time (Plank et al., 2014a); learning from such disagreements using cost-sensitive agreements can lead to better performance even within datasets, and to big improvements across samples and annotation projects. My third advise therefore is therefore to rather spend the adjudication time on annotating more data. In other words, future treebank designers should not adjudicate, but *include multiple, possibly inconsistent, annotations* of each sentence.

5 Crazy Trees

In addition to sampling data adversarially, introducing multiple splits, and collecting multiple, unadjudicated annotations, I also would like to question another straight-jacket in treebanking projects, namely the need for our annotations to be well-formed trees (or directed acyclic graphs for that matter). Some linguists have argued that some sentences are best described by cyclic structures, for example (Pollard and Sag, 1994). Such analyses never make it into treebanks,[10] and I think the main motivation is the idea that modern parsers require well-formed input trees.

Many parsers *are* designed to work only on trees, whether dependency trees or constituent trees, but recently several architectures have been introduced that do not hardwire this constraint into their models. Examples include sequence-to-sequence parsers (Luong et al., 2016) and so-called *tensor-LSTMs* (Schlichtkrull and Soegaard, 2017).

Sequence-to-sequence parsers encode input sentences using recurrent neural networks, recurrently appliying the transition parameters of the encoder. In their simplest version, they then generate a sequence of output symbols, one symbol at a time. After encoding the input sentence, the initial state is the vector sentence representation. From this vector, the parsers predict the most likely output symbol. The next state, frmo which the next output symbol is predicted, is obtained by applying the transition parameters of the decoder to the current state.

The encoder, responsible for learning a representation of the input sentence, and the decoder, which generates the parse, only interact through the vector representation. The sequence-to-sequence parser does not guarantee well-formed tree output. On the other hand, this also means the parser is not restricted to generating trees. Sequence-to-sequence models can learn to generate sets of edges from strings and thus associate input sentences with general graphs.

[9] One reviewer raised the fair concern, reading this, that *in practice, most inconsistencies in annotation involve silly stuff like the proper annotation of named entities, foreign language, titles, annotating collocations as fixed expressions or compositionally, etc. All of these are not 'real' ambiguities [...], but just a matter of detailed instructions.* I agree, but for the same reason we should a) either not insist on there being a correct annotation in these cases, or b) simply not annotate these cases at all (see §5 for why it is not necessary to insist on fully connected trees).

[10] One reviewer rightly points out that some treebanks actually contain cyclic structures, because of secondary edges, but these are ignored in parsing papers. Good news is that we do not need to ignore such edges anymore.

Tensor-LSTMs run recurrent neural networks over the rows of weight matrices of input sentences. They produce weights over potential heads of possible dependent. In Schlichtkrull and Soegaard (2017), minimum spanning tree search is used to find the best output tree, but this decoding step – which is only used at test time, not during training – is easily removed, and the output matrices can be scored directly against gold-standard general graphs.

6 Is the World Ready?

You may be thinking whether the parsing community is ready for this? Even if you agree that adversarial splits and multiple annotations are great for scientific reasons or engineering purposes, you may wonder whether researchers are not too conservative to adopt treebanks that depart radically from what has been standard methodology for ages.

The answer to this question is that yes, it is unlikely that everyone in the parsing community will adopt this over night, but that designing adversarial, multiply annotated treebanks could pave the way for the researchers who *are* ready.

One reason to think that more and more researchers are ready, is the steadily growing interest in topics such as transfer learning, multi-task learning and robust generalization. This interest is evidenced by the growing number of papers at our main conferences on these topics, recent workshops fully dedicated to one or more of these topics, as well as conference tutorials giving young researchers the necessary background to engage in these topics.

One example of this was the builders-and-breakers workshop at EMNLP 2017 in Copenhagen, Denmark.[11] Here, attendants were encouraged to come up with hard examples that would fool state-of-the-art NLP models. This is exactly why I am proposing adversarial splits in treebanks with multiple, difficult test sets. In order to understand how our models generalize, we need to prevent our evaluation set-ups from rewarding overfitting.

7 Summary

This is clearly an opinion piece. While I feel I have provided some justifications for my opinions, the paper clearly does not live up the standards of a technical track paper. I nevertheless the community will gradually, over time, adopt the following principles: a) Include several test sets in your treebanks that diverge more or less from the training data, but are generally as heterogeneous as possible. b) Devise multiple training-test splits, providing researchers with more data points for estimating the performance of their parsers in the wild. c) Choose several annotations per sentence over adjudication. d) Do not necessarily restrict the citizens of treebanks to be trees. Parsers can handle more complex structures, so include them if linguistically motivated.

Acknowledgments

Thanks to the anonymous reviewers for their comments that helped improve the paper. This research is funded by the ERC Starting Grant LOWLANDS No. 313695.

References

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *NIPS*.

Stephan Chalup and Alan Blair. ???? *Neural Networks pages = 955–972, title = Incremental training of first order recurrent neural networks to predict a context-sensitive language, volume = 16, year = 2003* .

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *EMNLP*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

[11] https://generalizablenlp.weebly.com/

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *EMNLP*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *EMNLP-CoNLL*.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014a. Learning POS taggers with inter-annotator agreement loss. In *EACL*.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014b. Linguistically debatable or just plain wrong? In *ACL*.

Carl Pollard and Ivan Sag. 1994. *Head-driven phrase structure grammar*. The University of Chicago Press, Chicago, Illinois.

Michael Schlichtkrull and Anders Soegaard. 2017. Cross-lingual parsing with late decoding for truly low-resource languages. In *EACL*.

Roy Schwartz, and Omri Abend, Roi Reichart, and Ari Rappoport. 2011. Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation. In *ACL*.

Anders Søgaard and Martin Haulrich. 2010. On the derivation perplexity of treebanks. In *TLT*.

Valentin Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2009. Baby steps: how "less is more" in unsupervised dependency parsing. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*.

Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2012. Cross-framework evaluation for statistical parsing. In *EACL*.