Year: 2017

A method for in-depth comparative evaluation: How (dis)similar are outputs of POS taggers, dependency parsers and coreference resolvers really?

Tuggener, Don

Abstract: This paper proposes a generic method for the comparative evaluation of system outputs. The approach is able to quantify the pairwise differences between two outputs and to unravel in detail what the differences consist of. We apply our approach to three tasks in Computational Linguistics, i.e. POS tagging, dependency parsing, and coreference resolution. We find that system outputs are more distinct than the (often) small differences in evaluation scores seem to suggest.

Originally published at:
Tuggener, Don (2017). A method for in-depth comparative evaluation: How (dis)similar are outputs of POS taggers, dependency parsers and coreference resolvers really? In: 15th Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 3 April 2017 - 7 April 2017. Association for Computational Linguistics, 188-198.

A method for in-depth comparative evaluation: How (dis)similar are outputs of POS taggers, dependency parsers and coreference resolvers really?

Don Tuggener
University of Zurich
Institute of Computational Linguistics
tuggener@cl.uzh.ch

Abstract

This paper proposes a generic method for the comparative evaluation of system outputs. The approach is able to quantify the pairwise differences between two outputs and to unravel in detail what the differences consist of. We apply our approach to three tasks in Computational Linguistics, i.e. POS tagging, dependency parsing, and coreference resolution. We find that system outputs are more distinct than the (often) small differences in evaluation scores seem to suggest.

1 Introduction

While there exist well-defined procedures to evaluate system outputs against manually annotated gold data for many tasks in Computational Linguistics, generally little effort and exploration goes into identifying and analysing the differences between the outputs themselves. System outputs are usually compared in the following manner: The standard evaluation protocol for many tasks consists of comparing a system output (the *response*) to a manual annotation of the same data (the *key*). The difference between the response and key is quantified by a similarity metric such as accuracy, and different system outputs are compared to each other by ranking their scores with respect to the similarity metric.

However, comparing the scores of the similarity metric does not paint the full picture of the differences between the outputs, as we will demonstrate. There are hardly any principled or generic evaluation approaches that aim at comparing two or more system responses directly to investigate, highlight, and quantify their differences in detail. Closing this gap is desirable, because progress in many NLP tasks is often made in small steps, and it is often left unclear what the specific contribution of a novel approach is if the comparison to related work is solely based on a (sometimes marginally) small improvement in F1 score or accuracy. Furthermore, an overall improvement regarding accuracy achieved by a new approach might come at the cost of failing in some areas where a baseline system was correct. Vice versa, a new approach might not improve overall accuracy, but solve particular problems that no other system has been able to address.

We propose an evaluation approach which aims at shedding light on the particular differences between system responses and which is intended as a complement to evaluation metrics such as F1-score and accuracy. By doing so, we strive to provide researchers with a tool that is able to give insight into the particular strengths and weaknesses of their system in comparison to others.[1] Our method is also useful in iterative system development, as it tracks changes in the outputs of different system versions or feature sets. Furthermore, our approach is able to compare multiple system outputs at once, which enables it to identify hard (or easy) problem areas by assessing how many of the systems solve a problem correctly and give according upper bounds for system ensembles. The performance difference between the simulated ensemble and the individual systems serves as an additional indicator of the difference between the system outputs.

We exemplify the application of our approach by aiming to answer the question of how (dis)similar are the outputs of several state-of-the-art systems for different tasks in NLP. We first motivate why using evaluation metrics such as accuracy is not suited for comparing outputs (next section). We then propose a method to do so which

[1] Code available at: https://github.com/dtuggener/ComparEval

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 188–198,
Valencia, Spain, April 3-7, 2017.

introduces an inventory to systematically classify and quantify output differences (section 2). Next, we demonstrate how combining a set of outputs can be used to identify their divergence and to identify hard (and easy) problem areas by looking at upper bounds in performance achieved by an oracle output combination (section 3).

1.1 Motivation

First let us motivate why comparing accuracy or F1 scores is not a suited method for establishing the (dis)similarity of system outputs. Consider a simple synthetic problem set with four test cases $\{A, B, C, D\}$ (e.g. a sequence of POS tags). A system response S_1 solves correctly the cases A and B, while a system response S_2 returns the correct answers for the cases C and D. In terms of accuracy, both responses achieve identical scores, i.e. 50%. However, their output is maximally dissimilar. Extending the set of cases, assume five problems, $\{A, B, C, D, E\}$, and three responses S_1, S_2, and S_3 as shown in table 1. Although the three responses achieve the same accuracy (left table), their pairwise overlap in terms of identical correct responses (right table) varies considerably, i.e. S_1 is much more similar to S_2 (two shared answers) than to S_3 (one shared answer).

Key	A	B	C	D	E	Acc.
S$_1$	A	B	C			60%
S$_2$		B	C	D		60%
S$_3$			C	D	E	60%

Overlap	
S$_1$, S$_2$	67%
S$_2$, S$_3$	67%
S$_1$, S$_3$	33%

Table 1: Accuracy vs. Overlap on correct answers

In fact, the establishment of the similarity of the responses S_1, S_2, and S_3 is more complicated, because we have left out the overlap of the incorrect answers in the responses. Consider the full responses in table 2.

Key	A	B	C	D	E	Acc.
S$_1$	A	B	C	*X*	*Y*	60%
S$_2$	*Z*	B	C	D	*U*	60%
S$_3$	*Z*	*W*	C	D	E	60%

Overlap	
S$_1$, S$_2$	40%
S$_2$, S$_3$	60%
S$_1$, S$_3$	20%

Table 2: Accuracy vs. Overlap on all answers

The overlap metric (right table) now compares how many of the cells in two rows have identical answers, regardless of whether the answer is correct. The overlap-based similarities between the systems have become more diverse, i.e. S_1 and S_3 are more dissimilar than in the previous table, and the similarities of the pairs (S_1, S_2) and (S_2, S_3) are now distinct, because S_2 and S_3 share the error Z (beside the correct answers C and D), while S_1 and S_2 do not share an error.

Hence, evaluating systems based on performance metrics such as accuracy and F1 scores provides no insight into the differences between the systems and is not able to accurately quantify the similarities between them. That is, a small difference in accuracy does not necessarily imply a high similarity of the outputs, and, vice versa, a larger difference in accuracy does not necessarily signify vastly dissimilar outputs.

Moreover, evaluation based on scores in performance metrics such as F1 does not detail in what regard a system performs better than another. Two systems might implement very distinct approaches, but achieve very similar scores in evaluation. Based on e.g. F1, we cannot assert whether a response S_2 performs better than a response S_1 because a) it solves the same problems as S_1 and then some additional ones, or b) if S_2 and S_1 solve a quite diverse set of problems and S_2 happens to solve a few more in its area of expertise. Additionally, a system that performs better than a baseline is bound to make errors where the baseline was correct. The overall accuracies cannot tell us how often this is the case.

In summary, the comparison of systems based on overall performance scores only lets us glimpse the proverbial tip of the iceberg. Therefore, our approach to comparative evaluation features three main points of interest:

1. How are the differences between system responses quantifiable?
2. What is the nature of the difference between two responses?
3. How can we assess the divergence of a set of responses, how complementary are they?

We try to answer these questions regarding three main tasks, namely POS tagging, dependency parsing, and coreference resolution. We select these tasks because they are fairly widespread procedures in Computational Linguistics and their evaluation increases in complexity. While we limit ourselves to these, we believe our approach to be generic enough to be applied to other labeling problems, such as named entity recognition and semantic role labeling.

2 Quantifying differences in system responses

As argued above, system responses differ both regarding the correct answers they give and the errors they make. The underlying idea of our approach is to assess how many of the labeled linguistic units (i.e. tokens) in the key have different labels in the responses, regardless of whether the labels are correct.[2] In a second step, we use a class inventory to analyse and quantify these differences in more detail.

Formally, given a set of tokens T and two accompanying system responses S_1 and S_2, we quantify how many of the tokens $t_i \in T$ have a different label in S_1 and S_2:

$$diff(S_1, S_2 \mid T) = \frac{|\forall t_i \in T : label(t_i, S_1) \neq label(t_i, S_2)|}{|T|} \quad (1)$$

Note that switching the inequality condition ($\neq$) to equality ($=$) actually yields the accuracy metric. That is, taking S_1 as the key and the S_2 as the response and calculating accuracy produces the inverted results of our metric, i.e. $1 - diff(S_1, S_2 \mid T)$, since accuracy is the ratio of tokens that have identical labels. The question is then, why not simply use S_1 as the key and S_2 as the response and calculate accuracy? While this answers whether two systems solve a similar or diverse set of problems, it does not enable us to identify the sources of the differences that drive the better performance of one response over the other. That is, if a token has a different label in S_1 and S_2, we cannot tell which and if any of the responses is correct. Hence, we need to look at the gold labels of the tokens T in a key K. This enables us to categorise differences in the outputs into three distinct and informative classes[3]:

- **Correction**: S_1 labels a token incorrectly, S_2 corrects this error
- **New error**: S_1 is correct, S_2 introduces an error
- **Changed error**: Both S_1 and S_2 are incorrect but have different labels

The general algorithm to quantify differences in two responses S_1 and S_2 given a set of tokens $t_{i...n}$ in a key K is outlined in algorithm 1. This procedure lets us track and count how often S_2 has a different label than S_1, classify the difference, and calculate the percentage of each class of difference. The approach can be applied straightforwardly to comparing outputs of POS taggers and dependency parsers.

Algorithm 1 Track differences in two responses

Input: Key $K \ni$ tokens $t_{i...n}$, Responses S_1, S_2
Output: Difference D, Changes C

for $t_i \in K$ **do**
    $G = label(t_i, K)$
    $L_1 = label(t_i, S_1)$
    $L_2 = label(t_i, S_2)$
    $TokCnt{+}{+}$
    if $L_1 \neq L_2$ **then**
        if $L_2 = G$ **then**
            $C[correction][L_1, L_2]{+}{+}$
        else if $L_1 = G$ **then**
            $C[new\ error][L_1, L_2]{+}{+}$
        else
            $C[changed\ error][L_1, L_2]{+}{+}$
        $DiffLabel{+}{+}$
$D = \frac{DiffLabel}{TokCnt}$
return D, C

2.1 POS tagging

We compare three POS taggers than can be used off-the-shelf to tag German: the Stanford POS Tagger (Toutanova et al., 2003), the TreeTagger (Schmid, 1995), and the Clevertagger (Sennrich et al., 2013, state-of-the-art). Following Sennrich et al. (2013), we use 3000 sentences from the TübaD/Z (Telljohann et al., 2004), a corpus of articles from a German newspaper, as a test set.[4]

Table 3 shows the labeling accuracy of the POS taggers and the percentage of correctly tagged sentences. The accuracy improvement of Clevertagger over TreeTagger is +1.27 points, and the percentage of correctly tagged sentences increases substantially (+9.9 points). In comparison to the Stanford tagger[5], Clevertagger raises performance

[2] In other words, the complement of the overlap metric in tables 1 and 2.

[3] To motivate the nomenclature, we assume that S_1 is e.g. a baseline upon which S_2 tries to improve. However, the outputs can stem from any two systems.

[4] We change the POS tag for pronominal adverbs from PROP to PROAV in the test set, since all taggers feature only the latter tag.

[5] The most frequent error by the Stanford tagger is labeling some punctuation tokens (e.g. '-') as '$[' instead of '$('. Considering it a minor error, we replace all '$[' labels in the Stanford response with '$(', increasing accuracy from 86.60 to 90.41%.

	Accuracy	**Correct sents.**
Stanford	90.41	30.07
TreeTagger	94.89	46.87
Clevertagger	96.16	56.77

	diff	Δ **Acc.**
Stanford ↔ TreeTagger	**11.06**	4.48
Stanford ↔ Clevertagger	9.41	**5.75**
TreeTagger ↔ Clevertagger	4.96	1.27

Table 3: Accuracy and differences between POS taggers

by roughly 6 points in accuracy, but almost doubles the numbers of correctly tagged sentences.

In the lower table, we see that although the accuracy difference puts the Stanford tagger closer to the TreeTagger (4.48) than to the Clevertagger (5.75), the Stanford tagger's response is more different from the one of TreeTagger (11.06) than from the response of Clevertagger (9.41). Comparing the two best performing taggers, we see that despite their accuracy difference of only 1.27 points, they label 4.96% of the tokens differently.

To get a more detailed understanding of the differences, we apply algorithm 1 to the two outputs, whose results are shown in table 4[6], listing the five most frequent changes per difference class.[7] Of the 4.96% different labels in Clevertagger compared to TreeTagger, 58.71% are corrections, 33.13% are new errors, and 8.15% changed errors.[8] That is, one third of the changes that Clevertagger introduces are errors. This is a noteworthy observation which applies to all our system comparisons: Every improved response introduces a considerable amount of errors with respect to the baseline, i.e. it invalidates correct decisions of the baseline. While this observation is to some degree expected, our method is able to quantify and analyse such changes in detail.

Regarding the differences, we see that both the most frequent correction (NN→NE) and new error (NE→NN) evolve around the confusion of named entities and common nouns, which is especially difficult for German, since capitalization cannot be exploited to distinguish the two. Furthermore, Clevertagger frequently invalidates TreeTagger's correct labeling of finite verbs, tagging them as nonfinite (VVFIN → VVINF), although this change occurs under the most frequent corrections as well. While these are commonly known error sources for POS tagging German, our approach shows that they are in fact the main source of the differences between the output of the two top performing taggers.

Difference: 4.96% (2674/53928)	
Corrections: 58.71% (1570/2674)	
NN→NE	27.96
PIAT→PIDAT	10.83
NN→ADJA	5.03
VVFIN→VVINF	4.52
NE→NN	3.25
New errors: 33.13% (886/2674)	
NE→NN	19.53
VVFIN→VVINF	9.48
NN→NE	9.26
ADV→ADJD	6.09
KOUS→PWAV	4.06
Changed errors: 8.15% (218/2674)	
NE→NN→FM	13.30
FM→NN→NE	7.34
NE→NN→ADJD	6.88
KOUS→KOKOM→PWAV	4.13
XY→NN→NE	2.29

Table 4: Token-based label changes comparing TreeTagger → Clevertagger (and Key → TreeTagger → Clevertagger for changed errors)

2.2 Dependency parsing

The next task we investigate is dependency parsing. We choose English as the test language due to the lack of the availability of multiple parsers in other languages. We evaluate Google's recently released Parsey McParseface (Andor et al., 2016, state-of-the-art) and two versions of the Stanford parser, i.e. the PCFG (Klein and Manning, 2003) and the Neural Network version (Chen and Manning, 2014). We follow the standard evaluation protocol and use section 23 of the PennTreebank (Marcus et al., 1993) as a test set and exclude punctuation tokens. We evaluate on Stanford Dependency labels (de Marneffe and Man-

[6]For tag description see `http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html`

[7]Note that the change comparison can also be sorted by the biggest accuracy difference, cf. appendix table A.1.

[8]We here use the TreeTagger as S_1 and the Clevertagger as S_2. Inverting the roles does not change the percentage values, but simply switches corrections to new errors and vice versa.

ning, 2008), since Parsey support only them.[9] We apply the parsers and their models "as is", i.e. we do not change any configuration settings.

	UAS	LS	LAS	Sent
Stan. PCFG	87.96	92.26	85.36	24.17
Stan. NN	88.68	92.45	86.43	26.95
Parsey	92.70	92.86	88.94	28.89

	diff	Δ LAS
Stan. PCFG ↔ Stan. NN	14.01	1.07
Stan. PCFG ↔ Parsey	15.49	3.58
Stan. NN ↔ Parsey	13.62	2.51

Table 5: Parser performance and difference

In table 5, we report the unlabeled attachment score (UAS), the labeling score (LS), and the labeled attachment score (LAS) for the parsers. Furthermore, we evaluate how many of the sentences are fully parsed correctly given each criterion.

We see that Parsey outperforms the Stanford parsers mainly due to the performance in attachment (UAS). The performance differences in assigning grammatical labels (LS) are comparably marginal. Parsey also features almost identical performance in both attaching and labeling tokens. However, there is a gap compared to labeled attachment score, which indicates that although Parsey attaches more tokens correctly than the other parsers, it does not necessarily assign the correct grammatical label to these tokens. Looking at the difference chart, we see that despite the rather small differences in LAS (1-4 points), the parsers attach and label around 15% of the tokens differently. The Stanford parsers only differ in 1.07 points in LAS, but this difference is based on 14.01% (*diff*) of the tokens in the test set. Parsey outperforms the Stanford NN parser by 2.51 LAS based on 13.62% of the tokens. To gain a better understanding of the differences contained in these 13.62% of the tokens, we apply algorithm 1, whose output is shown in table 6.

The table shows that half (50.22%) of the 13.62% changed token annotations from Stanford NN to Parsey are corrections. All of these changes are attachment corrections, i.e. the label of the tokens are not changed, which correlates with the small difference we saw in labeling score. The

[9]We convert the PennTreebank to Stanford dependencies using the Penn Treebank converter included in the Stanford parser (`http://nlp.stanford.edu/software/stanford-dependencies.shtml`).

Difference: 13.62% (6776/49748)	
Corrections: 50.22% (3403/6776)	
nn → nn	10.93
prep → prep	9.49
cc → cc	5.32
conj → conj	4.17
advmod → advmod	2.59
New errors: 31.79% (2154/6776)	
vmod → partmod	9.38
amod → nn	8.08
prep → prep	7.38
vmod → infmod	5.43
npadvmod → dep	4.32
Changed errors: 17.99% (1219/6776)	
prep → prep → prep	5.00
vmod → vmod → partmod	2.95
advmod → advmod → advmod	1.97
cc → cc → cc	1.89
conj → conj → conj	1.23

Table 6: Token-based analysis of parser differences (Stanford NN → Parsey, LAS)

two most prominent corrections are the attachment of noun compound modifiers (nn) and prepositions (prep). Almost one third (31.79%) of the changes are new errors. Compared to the corrections, Parsey here changes the labels of the tokens. Contrarily, changed errors (17.99%) constitute roughly one fifth of the changes and mainly consist of changes in attachment.

2.3 Coreference resolution

The final task we investigate is coreference resolution. We choose three freely available systems for English, again due to the lack of available systems for other languages: the Stanford statistical coreference resolver (Clark and Manning, 2015, state-of-the-art), HOTCoref (Björkelund and Kuhn, 2014), and the Berkeley coreference system (Durrett and Klein, 2013). We use the CoNLL 2012 shared task test set (Pradhan et al., 2012).

The coreference task differs from the previous two, since not all tokens in a document partake in coreference relations (but all tokens are in syntactic relations and feature a POS tag). Furthermore, the linguistic units of coreference relations are not only single word tokens, but syntactic units called *mentions* (i.e. mostly noun phrases). Therefore, we have to adapt our similarity metric in equation 1. To quantify the difference of two coreference system outputs S_1 and S_2, given a key K, we count how many of the mentions m are classified differently using a mention classification function c:

$$diff(S_1, S_2 \mid K) = \frac{|\forall m \in S_1 \cap S_2 \cap K : c(m, S_1) \neq c(m, S_2)|}{|\forall m \in S_1 \cap S_2 \cap K|} \tag{2}$$

The mention classification function c requires a class inventory which is not featured by the common evaluation metrics for coreference resolution.[10] Therefore, we adapt the mention classification paradigm introduced in the ARCS framework for coreference resolution evaluation (Tuggener, 2014) which assigns one of the following four classes to a mention m given a key K and a system response S:

- True Positive (TP): m is correctly resolved to an antecedent.
- False Positive (FP): m has no antecedent in K but one in S.
- False Negative (FN): m has no antecedent in S but one in K.
- Wrong Linkage (WL): m has an antecedent in K but is assigned an incorrect antecedent in S.

However, one issue with ARCS is to determine a criterion for the TP class, i.e. under what circumstances is m regarded as resolved correctly. Tuggener (2014) proposed to determine correct antecedents based on the requirements of prospective downstream applications.[11] We implement one loose criterion and regard m as correctly resolved if any of its antecedents in S is also an antecedent of m in K. Conversely, if none of the antecedents of m overlap in S and K, we label m as WL. This yields the ARCS$_{any}$ metric. Alternatively, we require that the closest preceding nominal antecedent of m in S is also an antecedent of m in K, which yields the ARCS$_{nom}$ metric. This metric is more conservative in assigning the TP class, but implements a more realistic criterion for correct antecedents from the perspective of downstream applications.

The official CoNLL score MELA (average of MUC, CEAFE, and BCUB) and the recently proposed LEA metric (Moosavi and Strube, 2016), which addresses several issues of of the other metrics, as well as the ARCS scores, are given in table 7. Using the ARCS class inventory and equation 2, we quantify how many of the mentions are classified differently in the system responses.

	MELA	**LEA**	**ARCS$_{any}$**	**ARCS$_{nom}$**
Berkeley	62.06	54.80	71.25	59.13
HOTCoref	64.32	57.13	73.27	61.95
Stanford	**66.62**	**60.92**	**76.30**	**62.04**

ARCS$_{any}$	*diff*	**Δ F1**
Stanford ↔ Berkeley	27.13	5.05
Stanford ↔ HOTCoref	26.30	3.03
Berkeley ↔ HOTCoref	27.22	2.02

ARCS$_{nom}$	*diff*	**Δ F1**
Stanford ↔ Berkeley	35.39	2.91
Stanford ↔ HOTCoref	**37.45**	**0.09**
Berkeley ↔ HOTCoref	34.89	2.82

Table 7: Coreference resolution evaluation (F1) and differences (%)

The F1 scores are lowest for the LEA metric, because it gives more weight to errors regarding longer coreference chains. The ARCS$_{any}$ metric assigns the highest scores due to the loose criterion that any antecedent is correct as long as it is in the key chain of a given mention. Furthermore, all the metrics agree on the ranking of the systems.

The mention-based differences between the systems are considerably larger than the relatively small differences in F1 scores suggest. The Stanford systems outperforms HOTCoref by 2.3 MELA, 3.79 LEA F1, and 3.03 ARCS$_{any}$ F1, but the systems process one fourth (26.30%) of the mentions differently in the ARCS$_{any}$ setting. For the ARCS$_{nom}$ criterion, the differences are even larger. The Stanford system outperforms the Berkeley system by 2.91 ARCS$_{nom}$ F1, but the systems process 35.39% of the mentions differently. Furthermore, we observe that the differences in F1 (Δ F1) do not correlate with the differences of the outputs (*diff*) for both ARCS metrics. Given ARCS$_{nom}$, we see that the smallest difference in F1 (Stanford ↔ HOTCoref: 0.09) actually occurs between the two responses that the *diff* metric deems most dissimilar (37.45).

Finally, we apply algorithm 1, using the ARCS$_{nom}$ criterion and our mention classification

[10] The common metrics analyse either the links between mentions or calculate a percentage of overlapping mentions in coreference chains in the key and a response. They are not able to determine whether a given mention m is resolved correctly or assign a class to it.

[11] Machine translation requires pronouns to be linked to nominal antecedents, Sentiment analysis needs Named Entity antecedents (if available) etc.

scheme to the two best performing systems, i.e. HOTCoref and Stanford. Results are given in table 8.

Difference: 37.45% (5760 / 15382)	
Corrections: 44.62% (2570/5760)	
wl → tp	20.09
fn → tp	12.34
fp → tn	12.19
New errors: 41.65% (2399/5760)	
tp → wl	17.08
tp → fn	12.33
tn → fp	12.24
Changed errors: 13.73% (791/5760)	
fn → wl	3.87
wl → fn	9.86

Table 8: Comparison of two coreference responses (HOTCoref → Stanford)

We see that less than 50% of the changes that the Stanford system introduces are corrections (44.62%). But this percentage is still higher than the newly introduced errors (41.65%); hence the improvement in overall F1. Furthermore, the most frequent change is wrong linkages to true positives (wl → tp). The most frequent new error also involves true mentions, i.e. attaching correctly resolved mentions to incorrect antecedents (tp → wl). Recovering false negatives and rendering true positives to false negatives occurs equally frequent, roughly. Hence, the performance difference stems mainly from attaching anaphoric mention to (nominal) antecedents, rather than from deciding which mentions to resolve, which are two subproblems in coreference resolution.

3 System combination

Lastly, we combine the system outputs per task and calculate the upper bounds for perfect system combinations by deeming a token labeled correctly if at least one of the systems provides the correct label. The upper bounds are intended to be another measure of the (dis)similarity of the outputs: the higher the upper bound, the higher the divergence of the outputs. Furthermore, looking at per-label performance of all systems, we can identify labels with low scores but high upper bounds, which is an interesting starting point for future work.

3.1 POS tagging

We start with the POS tagging task and present the upper bound of the system combination if table 9. We also indicate the accuracy gains for the top ten most frequent POS tags relative to the best performing tagger (Clevertagger).

	Stan.	**Tree.**	**Clever.**	**Upper bound**	
Overall	90.41	94.38	96.16	98.52	+2.36
NN	96.01	98.47	98.06	99.55	+1.49
ART	99.62	99.32	99.43	99.85	+0.42
APPR	*86.10*	98.03	99.20	99.56	+0.36
NE	87.35	*77.46*	85.31	95.17	**+9.86**
ADJA	92.73	94.50	98.40	99.44	+1.04
ADV	89.25	91.71	90.93	95.48	**+4.55**
VVFIN	*79.73*	95.15	91.52	97.48	**+5.96**
VVAFIN	91.97	98.93	97.74	99.56	+1.82
KON	97.13	95.37	96.55	98.37	+1.82
ADJD	*72.37*	89.29	88.80	93.53	**+4.73**

Table 9: POS tagging upper bounds and accuracy (highest scores in green; lowest in red; middle in yellow)

The Stanford tagger, despite performing the lowest with respect to overall accuracy, achieves the highest accuracy on named entities (NE), while the TreeTagger struggles in this category particularly. The TreeTagger surpasses the other taggers on finite verbs (VVFIN) by a wide margin and auxiliary finite verbs (VAFIN). Clevertagger performs best overall, but interestingly, it only achieves the highest accuracy on three of the ten most frequent POS tags. Looking at the overall upper bound, we see that it more than halves the error rate of the best performing system and is near 99% accuracy. The POS tags that profit most from the combination are named entities. Interestingly, all the taggers have low accuracy with respect to this tag, but the upper bound of the combination drastically raises it. Hence, it seems that the taggers diverge mostly here, which correlates with our analysis of the difference between the two best performing systems in table 4.

3.2 Dependency parsing

Next, we analyse the upper bounds of the combination of the dependency parsers, given in table 10. in contrast to the POS tagging task, we find that the best performing system, Parsey (P-MP) achieves highest LAS for almost all considered labels. Still, its overall LAS is drastically increased by the upper bound (+5.99) of the perfect system combination. Two of the labels that benefit the most of the combination are amod (adjectival modifier), which is often confused with nn (noun compound modifier) as we saw in table 6, and advmod (adverb modifier). All parsers have below 90 LAS for these labels, but the combination raises performance to 95.26 and 91.40, respectively. Furthermore, prepositions (prep) gains considerably in LAS in the combination. We observed in table 6 that almost ten percent of the difference between Parsey and the Stanford NN parser stem from correcting attachments of prepositions. However, also more than seven percent of the difference stems from invalidating correctly attached prepositions in the Stanford NN output. The large performance jump in the combination of the systems is further evidence that the parsers are highly complementary with respect to prepositions.

	S-PCFG	S-NN	P-MP	Upper bound	
All	85.36	86.43	88.94	94.93	**+5.99**
prep	*78.76*	84.26	88.21	94.18	**+5.97**
pobj	94.26	95.30	96.35	98.62	+2.27
det	96.81	96.65	98.66	99.38	+0.72
nn	74.64	76.81	86.36	88.91	+2.55
nsubj	92.08	*89.78*	94.41	97.85	+3.44
amod	87.59	88.45	86.95	95.26	**+8.31**
root	93.79	*89.61*	95.74	98.63	+2.89
dobj	90.19	90.88	92.91	97.47	**+4.56**
aux	97.53	97.11	97.63	99.30	+1.67
advmod	74.48	78.56	82.97	91.40	**+8.43**

Table 10: Parsing accuracy (LAS) and upper bounds.

3.3 Coreference resolution

For the coreference task, it is not trivial to calculate the F1 upper bound of the response combination, as the systems do no feature the same mentions in their outputs[12], and disentangling the false positives is a cumbersome undertaking. Therefore, we limit our investigation to the gold mentions in the key and count for how many of them at least one of the responses produces a correct nominal antecedent, which yields the upper bound for ARCS$_{nom}$ recall. To gain a deeper insight into the benefits of the combination and the performance of the systems, we divide the mentions into nouns (named entities and common nouns), personal pronouns (PRP), and possessive pronouns (PRP$). Results are given in table 11.[13]

The system with overall best recall features the highest recall with respect to all mention types.

	Berk.	Stan.	HOT.	Upper bound	
Overall	55.72	58.13	59.34	73.39	+14.05
Nouns	59.67	59.76	60.99	73.48	+12.49
PRP	*50.66*	56.30	57.56	73.80	+16.24
PRP$	62.36	64.62	65.98	80.57	+14.59

Table 11: Mention-based coreference performance (ARCS$_{nom}$ recall) and upper bounds

However, there is a considerable difference in recall to the mention types for all systems: Possessive pronouns are more easily attached to correct nominal antecedents than nouns and personal pronouns. Furthermore, we see that upper bounds raise recall uniformly for all mention types by a considerable margin. This suggest that the outputs are indeed different in several regards, which correlates to the comparisons in tables 7 and 13.

4 Related work

One way to establish the difference of two system outputs is to apply statistical significance tests. However, there is generally little agreement on which test to use, and it is often not trivial to verify if all criteria are met for the application of a specific test to a given data set (Yeh, 2000). Furthermore, the significance tests provide no insight into the nature of the differences between two outputs.

Several survey papers analysed performance of state-of-the-art tools for POS tagging (Volk and Schneider, 1998; Giesbrecht and Evert, 2009; Horsmann et al., 2015) or dependency parsing (McDonald and Nivre, 2007). While these surveys provide performance results along different axes (accuracy, time, domain, frequent errors), they do not analyse the particular differences between the system responses on the token level and hence do not provide a (dis)similarity rating of the responses. Regarding dependency parsing, our work is most closely related to McDonald and Nivre (2007) and Seddah et al. (2013). Both papers analyse the performance of parsers with respect to several subproblems. McDonald and Nivre (2007) also performed output combination experiments to stress that the two parsers that they investigated are complementary to a significant degree.

Comparative system evaluation in shared tasks is usually performed by pitting scores in evalua-

[12]The systems have to decide which NPs they consider for coreference resolution (the anaphoricity detection problem). I.e. the mentions are not known beforehand, and the systems will hallucinate different incorrect ones.

[13]Note that the HOTCoref system has better recall than the Stanford system, but the Stanford system features better precision, which leads to a higher F1 score in table 7.

tion metrics against each other, e.g. the CoNLL shared tasks on coreference (Pradhan et al., 2011; Pradhan et al., 2012) or on dependency parsing (Buchholz and Marsi, 2006; Nilsson et al., 2007). While the post task evaluation of the CoNLL shared task 2007 included an experiment of system combination which showed performance improvements, it is generally left unclear how similar are the system outputs with (sometimes marginally) small differences with respect to the evaluation metrics.

Another branch of evaluation related to our work is error analysis. Gärtner et al. (2014) presented a tool to explore coreference errors visually, but does not aggregate and classify them. Kummerfeld and Klein (2013) devised a set of error classes for coreference and analysed quantitatively which systems make which errors. Martschat and Strube (2014) presented an analysis and grouping of recall errors for coreference and evaluated a set of system responses. However, these analyses focus on the errors of one system at a time and then compare the overall error statistics, i.e. there is no direct linking or combination of the responses. Hence, we believe our approach to be complementary to the work outlined above.

5 Conclusion

We have presented a generic dissimilarity metric for system outputs and applied it to several systems for POS tagging, dependency parsing, and coreference resolution. We found that systems with marginal differences in accuracy scores or F1 actually have considerably distinct outputs. We combined system outputs and calculated upper bounds in performance as an additional measure of the degree of difference between the outputs.

We discussed and applied a method for analysing the specific differences between two system outputs using a class inventory to label and quantify the differences. Our analysis revealed the (often considerable) quantity of new errors that improvements introduce compared to baselines. We believe that this kind of analysis is also useful during system and method design, as it allows one to track all changes in the output when adjusting a system or a feature set.

While we have explored our approach on three core tasks in Computational Linguistics, we believe it to be applicable to other areas in the field. Our hope is that our method of comparative evaluation will motivate other researchers to gain an in-depth understanding of the output of their systems and what distinguishes them from others, beyond differences in accuracy or F1 scores.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Google.

Anders Björkelund and Jonas Kuhn. 2014. Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47–57, Baltimore.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar.

Kevin Clark and Christopher D Manning. 2015. Entity-Centric Coreference Resolution with Model Stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405–1415, Beijing, China.

Marie-Catherine de Marneffe and Christopher D Manning. 2008. The Stanford Typed Dependencies Representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, Seattle, Washington, USA.

Markus Gärtner, Anders Björkelund, Gregor Thiele, Wolfgang Seeker, and Jonas Kuhn. 2014. Visualization, Search, and Error Analysis for Coreference Annotations. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 7–12, Baltimore, Maryland.

Eugenie Giesbrecht and Stefan Evert. 2009. Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus. In *Proceedings of the 5th Web as Corpus Workshop (WAC5)*, San Sebastian, Spain.

Tobias Horsmann, Nicolai Erbs, and Torsten Zesch. 2015. Fast or Accurate? - A Comparative Evaluation of PoS Tagging Models. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL-2015)*, Essen, Germany.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 423–430.

JK Jonathan K Kummerfeld and Dan Klein. 2013. Error-Driven Analysis of Challenges in Coreference Resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 265–277, Seattle.

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, jun.

Sebastian Martschat and Michael Strube. 2014. Recall error analysis for coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 2070–2081, Doha.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the Errors of Data-Driven Dependency Parsing Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131.

Nafise Sadat Moosavi and Michael Strube. 2016. Which Coreference Evaluation Metric Do You Trust? A Proposal for a Link-based Entity Aware Metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 1–27, Portland.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning*, Jeju.

Helmut Schmid. 1995. Improvements In Part-of-Speech Tagging With an Application To German. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Djame Seddah, Reut Tsarfaty, Sandra Kuebler, Marie Candito, Jinho Choi, Richard Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Wolin ski, Alina Wroblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Fourth Workshop on Statistical Parsing of Morphologically Rich Languages*.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 601–609, Hissar.

Heike Telljohann, Erhard Hinrichs, and Sandra Kübler. 2004. The Tüba-D/Z Treebank: Annotating German with a Context-Free Backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 2229–2232, Lisbon.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 173–180.

Don Tuggener. 2014. Coreference Resolution Evaluation for Higher Level Applications. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 231–235, Gothenburg.

Martin Volk and Gerold Schneider. 1998. Comparing a statistical and a rule-based tagger for German. In *Proceedings of KONVENS*.

Alexander Yeh. 2000. More Accurate Tests for the Statistical Significance of Result Differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, pages 947–953.

A Appendix

A.1 POS tagging

∆ Acc.	TT	CT	POS	#tok
-100.00	100.00	0.00	VAIMP	1
+97.34	0.00	97.34	PROAV	301
+96.65	0.00	96.65	PIDAT	179
+50.00	14.74	64.74	FM	156
+46.67	46.67	93.33	PWAT	15
+39.39	54.55	93.94	PTKA	33
+33.33	33.33	66.67	VVIMP	6
+28.57	71.43	100.00	APZR	7
+26.87	71.64	98.51	PWAV	67
-24.00	96.00	72.00	VMINF	25
+13.33	78.33	91.67	KOUI	60

Table 12: Largest accuracy differences between TreeTagger (TT) and Clevertagger(CT); number of token with POS tag in the test set (#tok)

A.2 Coreference resolution

Since the ARCS framework is relatively unknown and not widely used, we revisit the connection of our $diff$ metric to accuracy and F1 outlined in section 2 in order to use one of the coreference metrics to establish the differences between the outputs. We saw that our metric is inversely equivalent to accuracy when taking one system response as the key and the other as the response. That is, we can calculate the $diff$ ratio by $1 - \frac{|t_i \in T: label(t_i, S_1) == label(t_i, S_2)|}{|T|}$, which is equivalent to taking S_1 as the key and S_2 as the response (or vice versa). For the coreference task, we can thus use one response as the key and the other as the response. The resulting F1 score can then be used as an agreement value, which, however, does not provide any detailed analysis of the nature of the differences compared to the ARCS approach. Table 13 shows the F1 scores when using one response as the key and the second as response. Note that switching the key and the response role provides the same F1 scores for two responses; the only effect is that the recall and precision values are switched.

The table shows that using this approach, we obtain F1 scores that give quite high dissimilarities when turned into the $diff$ metric, i.e. $diff = 100 - F1$. The average of the $diff$ metric given MELA F1 is 28.90 ($100 - 71.10$); given LEA F1 it is 35.22 ($100 - 64.78$). Compared to the ARCS$_{any}$ average $diff$ (25.56) and the ARCS$_{nom}$ average $diff$, 34.09, the values are in a similar range.

Key	Response	LEA F1	MELA F1
Berkeley	HOTCoref	63.58	70.32
Berkeley	Stanford	66.03	71.91
Stanford	HOTCoref	64.73	71.08

Table 13: Coreference system comparison pairing responses