Combining decision trees and transformation-based learning to correct transferred linguistic representations

Simon Corston-Oliver and Michael Gamon
Microsoft Research
One Microsoft Way
Redmond WA 98052, USA
{simonco, mgamon}@microsoft.com

Abstract

We present a hybrid machine learning approach to correcting features in transferred linguistic representations in machine translation. The hybrid approach combines decision trees and transformation-based learning. Decision trees serve as a filter on the intractably large search space of possible interrelations among features. Transformation-based learning results in a simple set of ordered rules that can be compiled and executed after transfer and before sentence realization in the target language. We measure the reduction in noise in the linguistic representations and the results of human evaluations of end-to-end English-German machine translation.

1. Introduction

In an ideal world, the transferred linguistic representations produced by a machine translation (MT) system would contain all and only the features needed to ensure perfect fluent and grammatical text realization in the target language. In the world that we inhabit, transferred linguistic representations deviate from this ideal in three ways: the representations may be overspecified or underspecified or may contain noise.

Overspecification describes the situation in which more precision is given in the description of a linguistic constituent than is required by the target language. For example, a distinction in gender between masculine and feminine is semantically relevant to pronouns in English, but is only explicitly encoded for third person animate pronouns (e.g. “he” vs. “her”). Spanish, on the other hand, encodes a masculine vs. feminine distinction for inanimate third person pronouns. When translating from Spanish to English, we might see an overspecification of gender for third person pronouns, influenced by the lexical or semantic gender of the antecedent in Spanish. If the English sentence realization module is confronted with a node marked [-Animate +Pers3 +Masc] it must decide which features are overly specific. Overspecification of pronouns is a common problem given the fact that the resolution of antecedents in the source language is usually incomplete or uncertain using current anaphora resolution algorithms.

Underspecification describes the situation in which less precision is given in the description of a linguistic constituent than is required by the target language. For example, Japanese noun phrases are usually not marked for definiteness or number, whereas English frequently requires that these features be explicitly indicated by analytic or synthetic means. In the majority of cases, the machine translation system must make reasonable inferences concerning the source language to satisfy the requirements of the target language. An alternative approach would be to require the analysis of the source language to specify all the features required for a given target language, an approach that would become impractical as the number of target languages increases.

Among the residual noise, i.e. errors in the transferred linguistic representations that are neither over- nor under-specification, are incorrectly specified lexical items, gross errors in the structure of the logical form, and errors resulting from the transfer process or from mis-analyses of the source language. For typologically similar languages, in particular for those languages

with similar feature makeup, many residual errors are simply minor differences in logical form representation.

As analysis and transfer algorithms continue to be improved, we would hope to see less and less noise in machine translation. There remain other scenarios, such as dialogue (with noise inherent in speech recognition) and generation from non-linguistic inputs in which a sentence realization module must gracefully handle erroneous input.

We present a hybrid machine-learning approach to resolving errors in transferred linguistic representations, using a combination of decision tree learning and transformation-based learning. The resulting rule set changes binary feature values, thus correcting feature based deficiencies. We evaluate the effectiveness of this hybrid approach at reducing noise in transferred linguistic representations known as logical forms (LFs), and then evaluate the impact of those changes on translations generated in an English-to-German machine translation system (Dolan et al. 2001).

2. The Microsoft Research Machine Translation system (MSR-MT)

The experiments presented here were performed in the context of the MSR MT system (Dolan et al. 2001). The MSR-MT system is a data-driven translation system with knowledge-engineered analysis components. At training time, an aligned bitext is analyzed into Logical Form (LF) graphs. An alignment algorithm finds mappings between the LF of the source and target language (Menezes et al. 2001). The learned mappings are then stored in a bilingual transfer memory. At translation time, a sentence in the source language is analyzed to LF. The resulting LF is then mapped onto matching LF sub-graphs from the transfer memory, and a target language LF is constituted from the matching sub-graphs. Finally, the target language LF is used to generate the target sentence string.

The logical form that we employ is a graph data structure that expresses the propositional content of a sentence. Nodes in the graph contain citation forms of content words, annotated with binary features such as tense, aspect, definiteness, number, and person. Relations between nodes are indicated by labeled arcs. The LF normalizes surface syntactic alternations such as active/passive. The LF is described in more detail in (Heidorn 2000).

In our experiments, we employ our approach to clean up features on LFs transferred from English to German, attempting to make the combination of features on each node in the transferred LFs more closely resemble the LFs that result from analysis of native German sentences.

3. Transformation-based error-driven learning

Transformation-based error-driven learning (Brill 1993a, 1995), commonly referred to as "transformation-based learning" or TBL, is an automatic machine learning technique. The output of TBL is an ordered list of rules whose application to data results in a reduction in error. The best-known application of TBL has been to the task of part-of-speech tagging (Brill 1992, 1994). TBL has also been applied to a number of diverse linguistic tasks such as resolving syntactic attachment ambiguities (Brill and Resnik 1994), syntactic parsing (Brill 1993b), and word sense disambiguation (Dini et al 1998).

Figure 1 gives pseudo-code that describes the learning phase of transformation-based learning. As is customary, we explain the learning phase with respect to the task of part-of-speech tagging. An initial part-of-speech tag is assigned to each word, typically by choosing randomly among the parts-of-speech observed for each word, or by choosing the most commonly observed part-of-speech for each word. Transformations consist of what Brill (1995:545) calls a "rewrite rule" such as "Change the tag from modal to noun" and a "triggering environment" such as "The preceding word is a determiner."

```
Assign an initial value to each data point to create data set, D
Repeat
    Find the transformation T_i that gives the best reduction in errors in D
    If (ErrorReduction(T_i) ≥ Minimum)
        Add T_i to the ordered list of rules, R
        Apply T_i to all relevant cases in D
    End if
Until (ErrorReduction(T_i) < Minimum)
Emit R
```

Figure 1: Pseudo-code for TBL learning

The learning phase is a greedy search. During each iteration the transformation that results in the greatest reduction in errors in the data set D compared to a reference data set is selected. If more than one transformation yields the greatest reduction, one candidate is arbitrarily selected. The same transformation might be selected multiple times.

Learning ceases when the reduction in errors is less than a predetermined minimum. When investigating the performance of the algorithm in the limit, a minimum value of one or two is typically used. For practical purposes, higher minima might be used to reduce learning time and to avoid overfitting to the training data. The learned list of rules can simply be applied in strict sequence to new situations.

Transformation-based learning has several attractive properties. Most notably, the rules that are learned are interpretable by humans. Furthermore, the lists of rules tend to be more parsimonious than the output of a stochastic tagger. Brill (1995:557) for example, notes that 200 TBL rules trained on 64,000 words yielded comparable tagging accuracy to a set of 10,000 contextual probabilities emitted by a stochastic tagger.

On the downside, the run-time performance of TBL can be prohibitive. Performance during the learning phase can be improved by indexing schemes, by sampling from the set of possible transformations, or by assuming independence among the transformations (Samuel 1998, Ngai and Florian 2001, Hepple 2000). Similarly, the application of sequences of learned rules can be improved by indexing schemes that eliminate the vacuous application of rules to new data (Satta and Brill 1996) and by compiling the list of rules into a finite state transducer (Roche and Schabes 1995).

The most glaring deficiency of transformation-based learning, and the motivation for the technique described in this paper, is the lack of a mechanism for navigating the space of possible transformations. In practice, researchers have managed the search for transformations by specifying templates that describe a set of transformations to be tried. For example, in part-of-speech tagging, the pretheoretical intuition is that resolving part-of-speech ambiguities can be achieved with reference to very local contexts only. Templates can constrain the search space by considering the part-of-speech and/or lexeme of each token within a fairly small window of tokens on either side of the position under consideration. For example, one template might describe triggering environments that consider the part-of-speech of a word to the left of the current token and the lexeme of the word to the right. Ramshaw and Marcus (1994) show that with the right set of templates, TBL appears to be immune to overtraining. If irrelevant templates are added, however, overtraining is likely, especially near the end of the list of transformations. An example of an irrelevant template in part-of-speech tagging is a triggering environment five tokens removed from the current token, i.e. a token that is unlikely to be in a dependency relation to the current token. It is also conceivable that omitting a relevant rule template could lead to a degradation in tagging accuracy, since some relevant phenomena will not be captured.

Thus, current approaches to TBL crucially depend on preselecting all and only the relevant templates for transformations. Failure to satisfy this precondition will result in overtraining or under-performance. Satisfying this requirement has not been problematic in the tasks to which TBL has been applied to date, because a pretheoretical understanding of those tasks has enabled the formulation of appropriate sets of templates. In the experiments that we describe below, we did not have such pretheoretical intuitions to guide us. We have therefore formulated an approach to generating possible transformations by first building decision trees.

4. Combining decision trees and transformation-based learning

The main problem for the task of learning a set of rules to correct a noisy vector of features describing a node in the LF is the large number of features to be manipulated and the large number of features in the conditioning of the rules. On each LF node we have thirty-eight binary features that we want to manipulate. Each of those features is potentially conditioned on each of the remaining thirty-seven features on that node, as well as additional multi-valued features (such as the governing preposition) on the node and the parent node. Each node is described by a feature vector containing 370 elements—38 features to be manipulated, plus 332 features that are also

considered in the triggering environment. Each of the 38 features to be manipulated is potentially dependent on a combination of one through 369 of the remaining features. If all features were binary, this would yield a search space of

$$38 \cdot \sum_{n=1}^{369} \frac{369!}{n!(369-n)!}$$

i.e. 1.2×10^{111} possible transformations. Clearly it is not practical for TBL to consider such a large search space.

In order to narrow the rule space for the TBL learner to consider, we begin by learning a set of decision trees to predict the value of each of the thirty-eight target features given all other features in the vector. We use the WinMine toolkit (Chickering 2002) as our decision tree learner. Since the leaf nodes of the decision trees produced by WinMine describe probability distributions over possible values for the feature, the features in the vector could in principle be binary or multi-valued. For our experiments, however, all target features were binary-valued, although some of the input features are multi-valued.

For decision tree learning we use the complete set of feature vectors automatically extracted from all LF nodes in the analysis of 100k German sentences drawn from technical manuals.

A first reduction of the potential rule space is achieved through feature selection by the decision tree learner: of the 370 input features, only 221 were selected by WinMine as being predictive of the target features. Figure 2 illustrates a highly simplified fragment of the decision tree for the target feature [Def] (definite). On all nodes of the decision tree the probabilities for the 0-value and for the 1-value of the definiteness feature are given. The highlighted path through the tree is translated into rule format at the bottom of Figure 2. The feature [Proximal] is found on certain demonstratives. Note that in the feature notation used here, [-Def] does not entail [+Indefinite], e.g. generic NPs may occur with no explicit indication of definiteness.

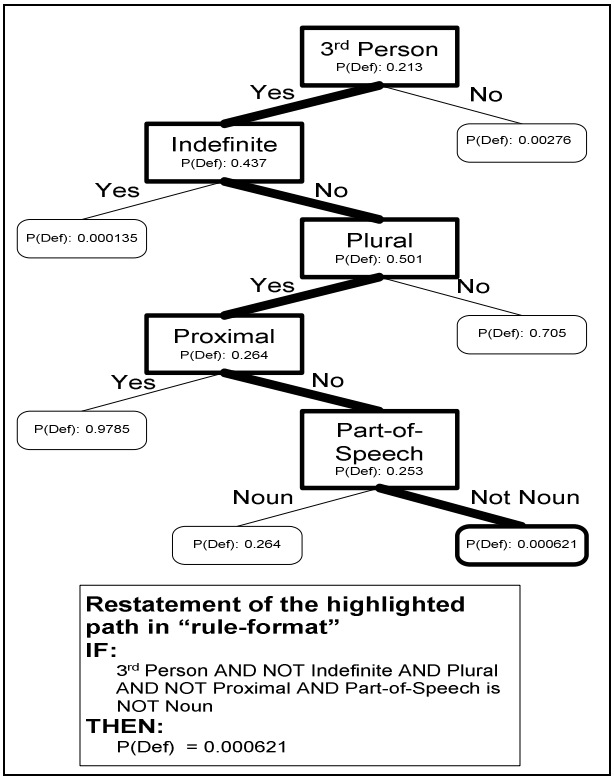

Figure 2: A decision tree fragment for the target feature [Def] (definite)

Next, we generate a set of transformations from the decision trees. We extract each path from the root to a leaf node, as well as each sub-path from the root to a branching vertex for each of the decision trees. These paths and sub-paths constitute the triggering environment for a transformation. For each path or sub-path, we note the most likely value for the target feature. This corresponds to the rewrite part of a transformation. As an illustration, given the decision tree fragment in Figure 2, we would extract the transformations in Figure 3. In our experiment, 18,420 transformations were generated from the decision trees. The triggering environments are formulated as C++ code, and then compiled.[1]

[1] It would be possible to manipulate the tree data structures directly at run-time, but compiling these conditions into executable code has yielded tremendous benefits in execution speed.

IF NOT(3rd Person): **-Def**
IF 3rd Person: **-Def**
IF 3rd Person AND Indefinite: **-Def**
IF 3rd Person AND NOT Indefinite: **+Def**
IF 3rd Person AND NOT Indefinite AND NOT Plural: **+Def**
IF 3rd Person AND NOT Indefinite AND Plural: **-Def**
IF 3rd Person AND NOT Indefinite AND Plural AND Proximal: **+Def**
IF 3rd Person AND NOT Indefinite AND Plural and NOT Proximal: **-Def**
IF 3rd Person AND NOT Indefinite AND Plural and NOT Proximal AND Part-of-Speech is Noun: **-Def**
IF 3rd Person AND NOT Indefinite AND Plural AND NOT Proximal AND Part-of-Speech is NOT Noun: **-Def**

Figure 3: All complete and partial paths of the decision tree fragment in Figure 2

The initial state of the data set used during TBL is set by taking the reference data (250k feature vectors extracted from German LF nodes) and randomly adding noise. Points in the vectors for the data set are randomly selected and changed. Because all target features are binary, values are either flipped from a one to a zero, or vice versa. More principled methods of adding noise might be appropriate in other contexts. For our experiments, however, we have no way of guessing what the pattern of errors might be in the transferred logical forms, since the pattern will vary according to the source language. Another possibility, training on actual transferred LFs, was not practical: The LFs are frequently so different from the LFs of the human reference translations as to make it impossible to align the two.

The TBL learning proceeds as described in Figure 1 with some optimization added to improve the execution speed of the learning phase. During the first iteration, each transformation is considered, and its error reduction noted. This error-reduction is cached for subsequent runs. The best transformation is added to the list of rules, and applied to the data. During subsequent iterations, we again consider all transformations, but only need to recalculate the error reduction for a subset of the transformations. If the best transformation during the previous iteration changed the value of $feature_j$ then for the next iteration we only need to recalculate the error reduction for those transformations that make reference to $feature_j$.

An additional optimization also substantially reduces the time taken during the learning phase. During learning each transformation is tested on each case in the data set. We can skip the remaining cases in the data set if the number of errors added by the current transformation is greater than or equal to the number of remaining cases, i.e. even if every remaining case were to result in an improvement the net effect of applying this transformation would be zero.

5. Results

Figure 4 presents the results for a data set of 250,000 LF nodes automatically extracted by the German NLPWin system from German technical manuals. The training set consisted of 200,000 randomly selected cases. The blind test set consisted of the remaining 50,000 cases. Varying amounts of noise were added to the data, ranging from 250,000 errors to five million errors. 221 features were used. Three multi-valued features, [POS] "part-of-speech", labeled relation to the parent node and the citation form of the governing preposition for both the current node and its parent were part of the conditioning features. The remaining 215 features, including the thirty-eight target features, were binary-valued linguistic features.

The results in Figure 4 show the reduction in the error rate measured against the blind test set. TBL learning ceased when no transformation yielded a net improvement of two or more. As the graph shows, the TBL learning is robust in the face of increasing amounts of noise. Furthermore, the TBL learning does not appear to overfit to the training data. As Ramshaw and Marcus (1994) observe, the addition of irrelevant rule templates does lead to overfitting. We avoid this situation by the automatic preselection of relevant rules.

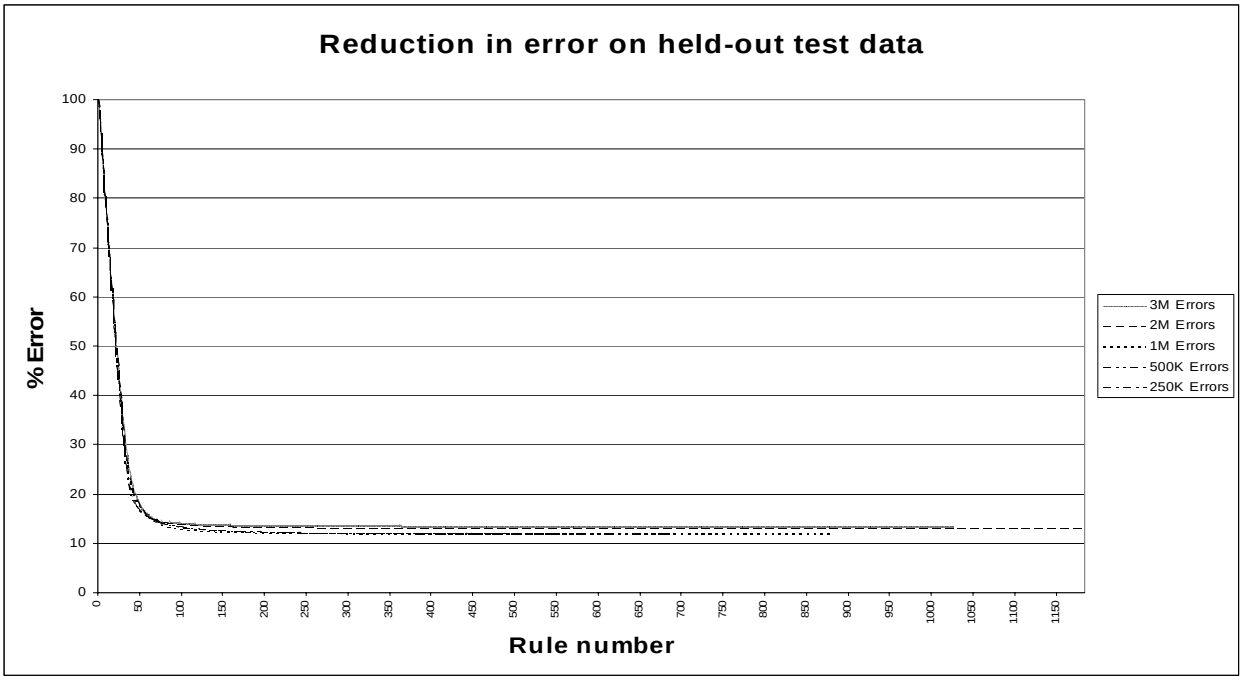

Figure 4: Reduction in error in held-out test data

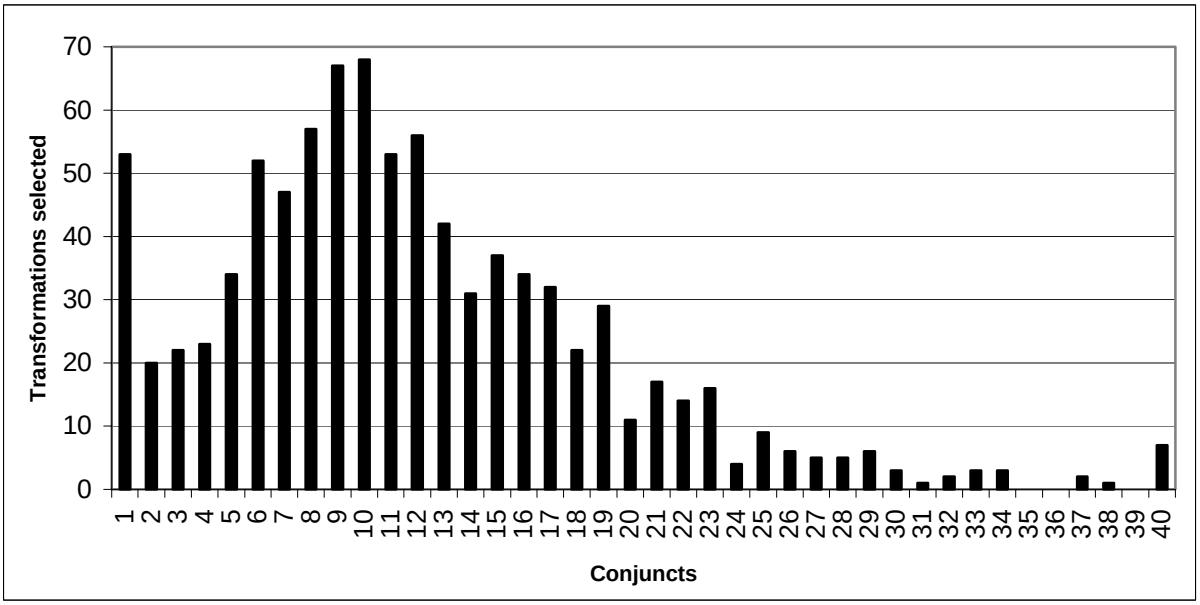

Figure 5: Transformations selected by TBL, introduced errors=1M

We experimented with two additional constraints that followed naturally from the fact that the transformations had been extracted from decision trees. Other things being equal:

1. Prefer transformations with fewer conjuncts in the conditioning environment, OR
2. Prefer the transformation with the sharpest separation between the two values of the binary feature.

The addition of these constraints did not materially affect the results. Constraint (1) merely affected the order in which the transformations were selected. Constraint (2) also produced only minor changes in the order in which transformations were added. This is not surprising: less sharp separations in the probabilities of the two binary values correlate with a greater amount of noise introduced by that transformation during learning. More noisy transformations are less likely to be selected during the greedy search.

The decision tree phase of the learning process proposed 18,420 possible transformations. With one million introduced errors, 894 of these transformations were selected. As Figure 5 shows, many rather complex transformations were selected. Seven transformations had forty or more conjuncts in the triggering environment. The most complex transformation had sixty conjuncts. The search for this transformation would have been prohibitive if the TBL stage were required to search more than 2^{60} possibilities. The initial decision tree stage however had reduced the search space to a mere two transformations worthy of consideration.

6. Evaluation

To evaluate the set of learned transformations, we compared two translations of a blind set of data. One set, labeled "No TBL", consisted of sentences that had been realized directly from the transferred LFs. For the second set, labeled "With TBL", we applied the learned set of transformations to the transferred LFs before performing sentence realization. We applied the set of transformations learned from 250,000 data points with one million introduced errors.

We took a sufficiently large sample to ensure that there were 250 differences in the output of the two systems. This sample consisted of 250 differences, and 716 sentences that showed no differences for the two scenarios.

Six independent human evaluators compared the output of the two systems in a randomized, anonymous presentation and indicated whether they preferred the output of one system over the other, or had no preference. Sentences that did not differ between the two systems were not evaluated. The results of the six human evaluators were averaged. An average score greater than zero indicates a preference for the "With TBL" scenario. Statistical significance was determined by Monte Carlo simulation. All results presented below are significant at $p < 0.01$.

For all 966 sentences, including those with no differences between the two systems, the average score was 0.093, i.e. there was a slight preference for the "With TBL" scenario. Considering only the 250 sentences with differences, the average score was 0.361, i.e. there was a marked preference for the "With TBL" scenario.

As a simple illustrative example of an improvement in translation quality through a TBL rule, consider the following English sentence and its translation:

Beachten	Sie	folgendes
Note	you	following

"Note the following."

In the German translation of the expression "the following", no definite determiner is used with the present participle "folgendes". TBL selected the following rule, extracted from the decision tree fragment in Figure 2.

IF (3rd person AND NOT Indefinite AND Plural AND NOT Proximal AND Part-of-Speech is NOT Noun): **-Def**

In the "No TBL" scenario, the transferred LF node corresponding to "folgendes" was marked [Def], i.e. it was over-specified, influenced by the English source. This resulted in the ungrammatical German output "Beachten Sie das folgendes". In the "With TBL" scenario, the application of this transformation removed the erroneous [Def] feature, resulting in the correct output, "Beachten Sie folgendes".

7. Conclusions

We have shown that it is possible to automatically select a manageable set of candidate rules for transformation-based learning in a scenario where feature vectors and the possible conditioning environments for transformations would otherwise be prohibitively complex. To select relevant rules from a massive space of logically possible transformations, we employ decision tree learning for those features that need to be manipulated by transformations. From the resulting set of decision trees we generate transformations by reading off partial and complete

paths to the leaf nodes. The set of transformations obtained from the decision trees is then used as the set of candidate rules for transformation-based learning.

We have implemented this technique in the domain of machine translation, in order to filter errors in transferred linguistic representations which are complex and contain large numbers of interdependent features. We believe that the technique we describe is generally applicable in scenarios where the candidate rule-space for TBL is prohibitive, and where it is impossible to constrain the space of possible transformation by using pretheoretical intuition to specify rule templates.

References

Brill, E. 1992. "A simple rule-based part-of-speech tagger." In *Proceedings of the Third Conference on Applied Natural Language Processing.* Trento, Italy. 152-155

Brill, E. 1993a. *A Corpus-Based Approach to Language Learning.* PhD thesis, University of Pennsylvania.

Brill, E. 1993b. "Automatic grammar induction and parsing free text: A transformation-based approach." In *Proceedings of the 31st Meeting of the Association for Computational Linguistics*. Columbus, Ohio, USA. 259-265.

Brill, E. 1994. "Some advances in transformation-based part-of-speech tagging." In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94).* 722-727.

Brill, E. 1995. "Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging." *Computational Linguistics* 21(4):543-565.

Brill, E. and P. Resnik. 1994. "A transformational-based approach to prepositional phrase attachment disambiguation." In *Proceedings of the Fifteenth International Conference on Computational Linguistics.* Kyoto, Japan. 1198-1204.

Chickering D. M. 2002. *The WinMine Toolkit.* Microsoft Technical Report MSR-TR-2002-103.

Dini, L., Di Tomaso, V., and Segond, F. 1998. Error Driven Word Sense Disambiguation. In *Proceedings of 36th ACL and 17th COLING*. 320-324.

Dolan, B., Pinkham, J., Richardson, S., and Menezes, A. 2001. "Achieving commercial quality translation with example-based methods." In *Proceedings of MT Summit VIII*, Santiago De Compostela, Spain. 293-298.

Heidorn, G. E.. 2000. Intelligent Writing Assistance. In *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*, R. Dale, H. Moisl, and H. Somers (ed.), Marcel Dekker, New York.

Hepple, M. 2000. "Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers." In *Proceedings of ACL 2000*. 278-285.

Menezes, A. and S. Richardson. 2001. "A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora." In *Proceedings of the Workshop on Data-driven Machine Translation at 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France. 39-46

Ngai, G. and R. Florian. 2001. "Transformation-based learning in the fast lane." In *Proceeding of NAACL* 2001. 40-47.

Ramshaw, L. and M. Marcus. 1994. "Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging." In *The Balancing Act: Proceedings of the ACL Workshop on Combining Symbolic and Statistical Approaches to Language*. New Mexico State University.pp. 135-156.

Roche, E. and Schabes, Y. 1995. "Deterministic part-of-speech tagging with finite state transducers." *Computational Linguistics* 21(2):227-253.

Satta, G. and E. Brill. 1996. "Efficient transformation-based learning". In Proceedings *of 35th ACL*. 255-262.

Samuel, K. 1998. Lazy transformation-based learning. In *Proceedings of the 11th International Florida AI Research Symposium*. Florida, USA. 235-239.