The Use of Transfer in Natural Language Processing

Jason Rennie
jrennie@csail.mit.edu

December 2, 2003

Abstract

Many NLP researchers have noted that a parser trained on one domain (e.g. WSJ articles) may not perform as well on another domain (e.g. New Yorker articles). Experiments have shown that replacing in-domain (target) training data with out-of-domain (reference) training data degrades parsing accuracy [Rat99]. Also, adding reference data to a set of target training data improves performance minimally [Gil01]. But, these results do not eliminate the possiblity of gaining useful information from reference data. [Hwa99] shows that a main contribution of target data is high-level phrase structure information. Low-level sentence structure can be learned from reference data. [RB03] shows that reference data is more useful when (1) there is less target data, and (2) when each reference example is given $1/5^{\text{th}}$ the weight of target example. While a parser trained on one domain is unlikely to perform well on a different domain, the reference data contains much useful information. The challenge is finding the right way to synthesize reference and target training data. The work we discuss shows initial progress, and also shows that there is much left to be done.

1 Introduction and Problem Statement

Parsing is the identification of the internal structure of a sentence. Humans, especially those trained in linguistics, are very good at producing a parse, a marked up version of a sentence that identifies parts-of-speech and how words in the sentence relate to each other. Much work has been done on the problem of teaching computers to parse sentences. A computer is given a large number of parsed sentences, the training data. An algorithm tells the computer how to use the training data to predict parses for new sentences. A problem with this approach is that the computer becomes an expert at parsing sentences that are similar to the training data, but isn't as good at parsing sentences from other domains. This isn't so surprising; humans have two big advantages: (1) we see data from a wide range of domains, and (2) when we read a sentence, we understand its "meaning," making it easier to abstract away from the syntactical

features that computer parsing algorithms tend to use. Getting a computer to understand the "meaning" of natural language is hard, or at least it's difficult enough that 50 years of research on the topic hasn't yet led to an answer. Narrowing the gap for the first advantage isn't quite as hard. We'll focus on this—giving the computer sufficient breadth of experience.

While humans are well built to experience a wide variety of dialogue—we conduct verbal exchanges on a daily basis—computers aren't. A computer can't (yet) learn like a human can. It needs labeled training data in order to learn. Trouble is that labeling training data is time consuming and expensive. There are only a handful of labeled, sizable parsing corpuses, such as the Wall Street Journal (WSJ) article corpus and the Brown corpus, which itself includes examples from various domains. For a new domain, such as medical abstracts, there isn't likely to be labeled data available. Our first option is to simply use a parser trained on the corpus training data. Such a parser is certain to make glaring errors. Medical terms will be out-of-vocabulary. Also, the sentence structures of medical documents are somewhat unique. The result will be a sub-par parser. A second option is to only use labeled data from the target domain. But, computer parsing algorithms require thousands of example sentences to be effective and labeling data is expensive. A final option is to label a small amount of target data and learn a parser using both the corpus data and the target data. This option clearly provides the most information; what is not clear is how to fuse these two sources of data to produce the best possible parser. The next section describes results that show the difficulty of the transfer task we have described. Attepts to blindly fuse reference with target data have not fared well. Next we will discuss work that has unveiled some of the structure of the transfer problem. Finally, we will provide our own thoughts on what can be done to improve the state-of-the-art.

2 Some (Not So) Negative Results

Some results in the NLP literature make the transfer problem look difficult. More to the point, they show that transfer cannot be solved simply by pooling the two sets of training data.

[Rat99] considers the three options we have discussed: (1) train with reference data only, (2) train with target data only, and (3) train with both reference and target data. He takes the most straightfoward approach to #3, pooling the two sets of examples into a single bunch of data. He uses 40,000 WSJ sentences for the reference data and uses three sets of data from the Brown corpus as three different target domains. He uses 2,000 sentences from each Brown corpus for target training data. Using the target data only, he gets the worst results, an average F1 of 77.6%. Performance is poor because of the small amount of training data. Using the reference data only, he gets an average F1 of 79.7%, much lower than the 85.2% Ratnaparkhi estimates he would get by training with 40,000 target example sentences. Using both the reference and target data performs best, an average F1 of 81.0%. But, this is still far below the 85.2%

mark. Ratnaparkhi concludes that using reference data in place of target data yields a significant loss in parser accuracy. Seen another way, reference data is not a replacement for target data.

What Ratnaparkhi does not discuss is the importance of the fact that using reference data improves performance. Clearly Brown and WSJ data are sufficiently similar that parsed WSJ sentences provide useful examples for the task of parsing Brown sentences. What is not clear is that his method of using the reference examples—simply pooling them with the target examples—is the most effective technique.

[Gil01] performs experiments on the WSJ and Brown corpra. He uses approximately 40,000 WSJ sentences and 22,000 Brown sentences for training. Like Ratnaparkhi, he finds results using a pooled data set uninspiring. F1 improves by just 0.25% for both data sets compared to using just the target data. Gildea comments that "even a large amount of additional data seems to have relatively little impact if it is not matched to the test material." One reason he sees such a small improvement is that he uses sizable target training sets. Ratnaparkhi used only 2000 target training setences for Brown and sees an F1 improvement of 3.4%. Clearly the amount that can be gained from the reference data depends on the amount of target training data. If there is already sufficient target data (as seems to be the case in Gildea's experiments), there is little need for reference data.

Gildea also tries reducing the side of the parsing model. He eliminates "lexical bigrams" a class of features that consume a large portion of the model (43% of one model). He claims that the statistics for these features are corpus-specific, but his results paint an unclear picture. When lexical bigrams are included, models built on both reference and target data always improve performance compared to models built without the reference data. However, when lexical bigrams are excluded, including reference data improves performance on the WSJ corpus but degrades performance on the Brown corpus. This indicates that the reference data lexical bigram statistics are advantageous.

Both Ratnaparkhi and Gildea claim that reference data isn't as useful as target data. This is hard to argue against. But, these claims fail to consider the common case where one might want to use reference data: when it is difficult or impossible to obtain large quantities of labeled target data. Here, what is important is that any gain can be made using reference data. Also, neither Ratnaparkhi nor Gildea made attempts to take full advantage of the reference data. Both simply pooled the additional data with the target data. Further improvements are likely to be found by weighting the reference data's inclusion in various parts of the parsing model.

3 Learning How to Use Reference Data

[RB03] takes the logical next step—to weight the reference data differently than the target data. He uses a Maximum a Posteriori (MAP) model that allows him to incorporate the reference data as prior information. Roark and Bacchi-

ani try two different weighting styles, count merging, where each target example is given a weight of one and each reference example is given a weight of $\alpha < 1$. The second style is called model interpolation; it does not weight the individual examples. Instead, it calculates probabilities for reference and target data separately and then interpolates between them. The disadvantage with model interpolation is that there is no inherent knowledge of the relability of statistics. As such, Roark and Bacchiani find that count merging always outperforms model interpolation.

The main contribution of Roark and Bacchiani is to show that reference data can significantly improve performance compared to using only target data. For example, Gildea finds a 0.25% improvement in F1 by training a parser on WSJ reference data and Brown target data. Roark and Bacchiani get a 0.95% improvement in F1 by simply giving each reference example a weight of 0.25. Roark and Bacchiani also show that the reference data becomes more useful as the amount of target data decreases. They test parsing performance on WSJ sentences, use the Brown corpus as reference data and vary the amount of WSJ training data. Using 100% of the WSJ target data, the inclusion of reference data only improves the model by 0.35% F1. When only 10% of the WSJ target data is used, referece data boosts performance 0.70% F1; giving each reference example a weight of 0.2 further boosts performance 1.05% F1, for a total improvement of 1.75% F1. These results show that reference data is more advantageous when there is less target data. Also, the advantage to be gained from the reference data depends on how it is used. Roark and Bacchiani use a single weight for the reference data. For example, reference data may provide excellent statistics for head word POS tags, but poor statistics for the actual words. Further improvements might be possible by, say, giving a large weight to the POS statistics and a small weight to the word statistics.

Roark and Bacchiani also learn a parser using labeled reference data and unlabeled target data. They first learns a parser with the reference data, then find the top 20 parses for each sentence in the target data set, calculate posterior probabilities for them and normalize the probabilities (so that the sum of posterior probabilities over the top 20 parses is unity). The expected counts from this distribution are used to train a new parser. In other words, they use Expectation-Maximization (EM) to learn parameters of the parsing model, with initial parameters set by a labeled reference data set. The find that even a single iteration greatly improves performance compared to using a parser trained on the reference data. Using the Brown corpus as reference data and using all 200,000 sentences in the WSJ corpus, they achieved an F1 score of 79.9% after one iteration and 80.6% after two iterations. This compares favorably against an F1 score of 75.7% for a parser trained on only the Brown corpus. Even when there is no labeled data for the target domain, it is still possible to adapt a parser to the target domain. The reference data serves as a good starting for an iterative procedure that adapts to the particulars of the target domain.

[Hwa99] examines the idea that the usefulness of reference data may depend on the statistic. Hwa looks at five different categories of phrases:

- BaseP - any phrase that does not subsume a lexical word
- HighP - any phrase that subsumes only lexical words
- NotBaseP - any phrase not in BaseP
- BaseNP - any noun phrase that does not subsume another noun phrase
- AllNP - any noun phrase

She considers learning a parser in two stages. First, a parser is learned using reference data. Then, the parser is retrained using the target data. But, instead of using fully labeled target sentences, one of the phrase categories is selected and only phrases from that category are identified. She compares this against a baseline[1] where a random subset of the brackets are included. She performs one set of experiments using the WSJ corpus as the reference data and a smaller, simpler corpus, ATIS, as the target data. In this set, HighP is the only category to perform better than the baseline. All other categories perform no better than the baseline. BaseNP and BaseP perform particularly poorly—just slightly better than using no labeled target data. The second set of experiments reverses the roles: ATIS is used as the reference data and WSJ is now the target data. Here, only HighP achieves a score equal to the baseline score while all other categories perform significantly worse. Again, BaseNP and BaseP perform the worst, both about seven percentage points below the baseline.

It is worth discussing why labeling a category of phrases can do worse than randomly labeling an equal number of phrases. Hwa does not explain this, probably because it is clear if you know the model. A plausable explanation is that when given good labels for only a category of phrases, the model has very good statistics for that category, but poor statistics for other types of phrases—it must rely on the reference data for such statistics. A parser with a random set of the target data brackets has partial target statistics for all types of phrases. Thus, it will do fairly well on all types of phrases. Increases in accuracy per example diminish as more examples are added, so it is usually better to spread examples across different types of categories. Only if the reference data provides good statistics for the complement category would you expect a particular category to do better than the baseline. This is what happens when HighP is used. When WSJ is the target data, HighP does as well as the baseline, when ATIS is the target data, HighP does better than the baseline. We conclude that reference data provides good statistics on non-HighP phrases. Since performance for BaseNP and BaseP are so poor, we can conclude that reference data provides poor statistics for non-baseNP and non-BaseP phrases. In summary, reference data provides good statistics for low-level phrases and poor statistics for high-level phrases. One might conclude that low-level phrase structure is a property of the English language (which changes little between corpra), yet high-level phrase structure is a property of the type of corpora.

[1]It is somewhat strange to call this a baseline since only once does any phrase category perform better than it. Most of the phrase categories perform much worse than the baseline.

Hwa's experiments show that the value of reference data is not uniform. Some aspects are more useful than others. Statistics for low-level phrase structure is very useful; further experiments may show that low-level phrase statistics from reference data can be used as a drop-in replacement for some types of low-level phrase target statistics. However, target data is necessary for learning high-level phrase statistics. Reference data improves performance in the absence of high-level phrase target statistics, but pales in comparison to using target data.

4 Summary and Commentary

The majority of work on parsing to this day has focused on improving performance when the training and test sets are drawn from the same domain. But, since labeling data for parsing is time-consuming, few domains have sufficient labeled training data for accurate parsing. More realistic is a scenario where a small amount of data has been labeled for the target domain. Using only this data will generally yield a sub-par parser. There should be some way to combine the target training data with training data from other domains to yield a good parser for the target domain. Experiments by [Rat99] and [Gil01] show that pooling reference data and target data yields a better parser. But, both Ratnaparkhi and Gildea were unimpressed with the pooled parser. [RB03] found that he could take better advantage of the reference data by down-weighting each reference example. He also found the greatest improvements (compared to using only the training data) when he used a small amount of target training data. We think this is the most realistic scenario. The experiments of Roark and Bacchiani showed that parsing performance improvements depend on how the reference data is used. [Hwa99] investigated an aspect of this, she trained parsers using reference data and certain types of target data. She found that the parser performed best when the target data included high-level phrase structure information. The reference data provided sufficient information about low-level structure to fill in the gaps left by the target data. This shows that the benefits of reference data are not uniform. To take full advantage of reference data, it is necessary to understand how the target domain relates to the reference domain—what aspects of parsing are similar and what are different.

It seems that the next step beyond the current work on using reference data in parsing is to extend the MAP framework of Roark and Bacchiani to allow for weights on individual statistics. Roark and Bacchiani used a single weight for the reference data. But, the reliability of statistics in the reference data will vary. Some statistics vary by corpora whereas others are properties of the English language. One can gain an advantage by using a different weight for each statistic. But, there is a tradeoff that must be made between the number of parameters and generalization performance. By allowing for one weight per statistic of the model, we get an explosion of new parameters. One would likely need to cluster together the statistics (such as the grouping used by Hwa) and assign one weight per cluster; alternatively, one could regularize the weights,

using a Beta prior with mean equal to the one found to work best in the single-weight framework. Hierarchical clustering or a hierarchical regularizer might also prove to be valuable.

References

[Gil01] Daniel Gildea. Corpus variation and parser performance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2001.

[Hwa99] Rebecca Hwa. Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 73–79, 1999.

[Rat99] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 1999.

[RB03] Brian Roark and Michiel Bacchiani. Supervised and unsupervised pcfg adaptation to novel domains. In *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics*, 2003.