Conversions for heterogeneous treebank parsing

L715: Seminar on: Data manipulation for parser improvement

Dept. of Linguistics, Indiana University

Fall 2011

Introduction
Niu et al. (2009)
Smith and Eisner (2009)
References

Introduction

Introduction
Niu et al. (2009)
Smith and Eisner (2009)
References

We are going to focus now on conversions for the purposes of creating more parsing data

- Fully automatic methods are preferable to rule-based ones
  - Allow for new schemes (i.e., be even more robust than last time)
- We will start with DS $\leftrightarrow$ PS issues, but the issue is more general
  - Convert a source annotation into a target annotation
    - different representation types, different conventions, different languages
  - i.e., find a common annotation scheme to parse with

Exploiting Heterogeneous Treebanks for Parsing

Niu et al. (2009)

Heterogeneous treebank contains multiple treebanks in different annotation schemes (grammar formalisms)

- To parse in target formalism, we have to solve: source treebank $\mapsto$ target treebank
- This is desirable, as it provides more labeled data

Two-step solution

1. Convert grammar formalism of source to target
2. Refine converted trees & use them as additional training data, for a target grammar parser
    - This can be iterative, retraining on converted data

Approach taken here:

- DS-to-PS conversion, to better train a PS parser
- Use existing *n*-best parser to generate conversion candidates
    - select the parse most consistent with source tree as the converted tree

Other avenues which are pursued:

- pruning low-quality trees
- interpolating scores from source & target grammars
- corpus weighting

Limitations of previous approaches

- “For each head-dependent pair, only one locally optimal conversion was kept during tree-building process”
  - Potentially ignores globally optimal conversions
- Heuristic rules are used to do the conversion, when multiple possible conversions exist
  - Usually have to be hand-crafted

Grammar formalism conversion

Notation:

- C_{DS} = source treebank annotated with dependency structure (DS)
- C_{PS} = target treebank annotated with phrase structure (PS)
- Goal: convert C_{DS} to C_{PS}

Steps:

1. Train a constituency parser on C_{PS} (target)
2. Generate *n*-best parsers for C_{DS} (source)
3. Convert *n* parses ($x_{i,t}$) to dependency trees ($x_{i,t}^{DS}$) (more on this in a moment)
4. Compare converted dependency trees ($x_{i,t}^{DS}$) to gold standard tree (y_i), obtaining $Score(x_{i,t})$
   - measured by parseval F-score
5. Determine the PS tree by taking the one which corresponds to the maximum $Score(x_{i,t})$

Grammar formalism conversion (2)

The method as outlined above can be repeated

- Converted trees can be used as additional data to retrain the *n*-best parser
- Development data ($C_{PS,dev}$) is used to determine when iterations are no longer helping

In general, once the conversion is done, heterogeneous parsing now is the same as homogeneous parsing

- i.e., treebanks are in the same format

Grammar formalism conversion (3)

The conversion from DS to PS involves a step of conversion between PS to DS, in order to make the *n*-best (PS) trees comparable to the gold (DS) tree

- The method relies upon there being some way to objectively compare the set of parsed trees with the gold ones in the treebank
- If it were a PS-to-PS conversion, this would have to be done differently

Their method is relatively simple:

1. Find the head of each constituent, using a head table
2. Make the head of each non-head child depend on the head

Target grammar parsing
Instance pruning

n-best parser may fail on some cases, i.e., giver poor-quality converted trees

- **Instance pruning**: remove converted trees with low unlabeled f-scores
- Then, do parser training

Target grammar parsing

Score interpolation

Unlabeled dependency F-score measures quality from the perspective of the *source* (DS) grammar

- What about from the perspective of the target grammar?
- After all, there can be different ways of viewing grammar that need to be reconciled towards the target
  - "conflicts of syntactic structure definition"
  - e.g., preposition or noun as the head? (see figure 1)

The score is thus modified to take parser probability/confidence into account:

$$(1)\quad \widehat{Score}(x_{i,t}) = \lambda Prob(x_{i,t}) + (1-\lambda)Score(x_{i,t})$$

Corpus weighting

One other issue to be determined: if corpora are of different sizes, how are they balanced as parser training data?

- **Corpus weighting:** reduce the weight of the larger corpus (in this case C_{DS}) when training
- This may also reduce the influence of potentially corrupt trees

Evaluation on WSJ

Their results in tables 2 & 3 show improvement

- The measurements correspond to accuracy of recovering the original PS trees (not parsing accuracy)

Parsing experiments on Chinese

Introduction
Niu et al. (2009)
Smith and Eisner (2009)
References

Used CDT and CTB, in order to parse in CTB phrase-structure style

- Corpus weighting: tried increasing the weight of CTB in merging: optimal value = 10
- Both generative and reranking parser show improvements over baseline (table 5)
  - e.g., 83.3% → 83.8%

Instance pruning

Instance pruning was done on the development set

- Result: it hurt to remove any converted trees
- Perhaps: even imperfect parses provide some useful syntactic information

Score interpolation

Used $\widehat{Score}(x_{i,t})$ to replace $Score(x_{i,t})$

(2) $\widehat{Score}(x_{i,t}) = \lambda Prob(x_{i,t}) + (1 - \lambda) Score(x_{i,t})$

- λ was tuned on the development set to be 0.4
- average index of 200-best trees increased to 2, i.e., higher up the list / more like target grammar

Results go up even further, e.g., 83.3% → 83.8% → 84.2%

Using unlabeled data as part of self-training helps even more (section 4.3)

Summary

Benefits of this approach:

- A parser generates globally-optimal syntactic structures
- No heurstic rules are needed
- Converted trees can retrain the parser and improve the conversion

Quasi-synchronous grammar features

Smith and Eisner (2009)

The framing of the problem for Smith and Eisner (2009) is a bit more general

- Any source corpus annotation needs to be converted to a target annotation, in order to train a parser
  - Without such conversion, adding source training data will result in ill-formed analyses
- Multiple constructions need alteration → must learn a statistical model, not just write a few rules

The general task

Additionally, these are *different* sentences which are annotated, so we cannot directly learn transformations

- But we can automatically obtain pairs of trees
- Train parser on source corpus, parse target, and learn from those pairings
  - Note that this is the opposite direction from Niu et al. (2009)
- Learn tree transformation model from those pairings to obtain the source corpus in the target style

Parser projection

Parser projection is a case of taking source annotation from one language and projecting it into a target language

Assume these variables:

- w = target language; t = target annotation
- w' = source language; t' = source annotation
- a = alignment between languages

Goal of projection is to model $p(t|w, w', t', a)$ (or, generatively, $p(w, t, a|w', t')$)

Parser adaptation is a subset of this problem, where the alignment is trivial: a word maps to itself

Form of the Model

Arbitrary graphs

Syncrhonous grammar modeling assumes that source & language trees have a direct correspondence

- e.g., "two nodes can be aligned only if their respective parents are also aligned"

Quasi-synchronous grammars: model the alignments as an *arbitrary graph*

- arbitrary links between the words of the two sentences
- permits non-synchronous & many-to-many alignments
  - "Local syntactic configurations tend to occur in each language"
  - "we might learn that parses are 'mostly synchronous,' but that there are some systematic cross-linguistic divergences"

General point: allow there to be divergences between trees, but learn the systematicity

Form of the Model

Scores & features

Introduction
Niu et al. (2009)
Smith and Eisner (2009)
References

Score of a given tuple:

(3) $s(t,t',a,w,w') = \sum_i w_i f_i(t,w) + \sum_j w_j g_j(t,t',a,w,w')$

- target features **f**: based only on target words and dependencies
  - features of an edge-factored dependency parser (e.g., POS of potential relation)
- alignment features **g**
  - features for $x \rightarrow y$ (target) consider relationship between x' and y'
  - e.g., features for monotonic projection, head-swapping, various configurations (e.g., sibling)

Adaptation

Training done with both gold and noisy trees, to gauge the effect of parser noise

- Use MSTParser to train on source & parse a (small) amount of target data
- Train edge-factored parser with QG features on target data

Source & Target are in different conditions (preposition-as-head, coordination differences):

- Results in table 1 show that even with a small amount of trees, substatntial gain can be made

Results for cross-lingual projection & adaptation also show improvement (section 6)

References

Niu, Zheng-Yu, Haifeng Wang and Hua Wu (2009). Exploiting Heterogeneous Treebanks for Parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 46–54.

Smith, David A. and Jason Eisner (2009). Parser Adaptation and Projection with Quasi-Synchronous Grammar Features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, pp. 822–831.