Grammar Engineering for CCG using Ant and XSLT

Scott Martin, Rajakrishnan Rajkumar, and Michael White
Department of Linguistics, The Ohio State University

Motivation

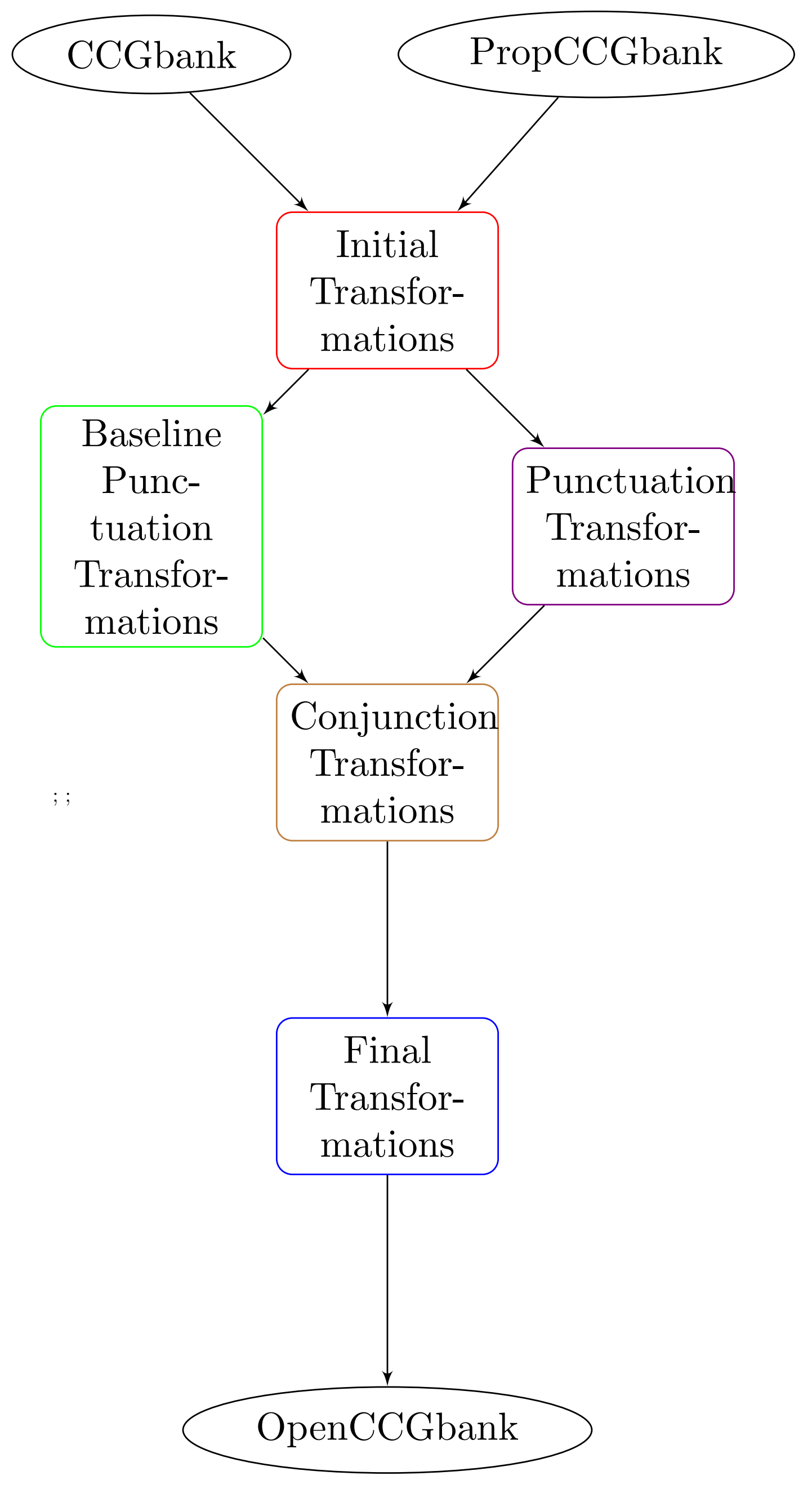
- Transform grammar engineering from a one-shot task to an evolving, iterable process.
- Augment the CCGbank (Hockenmaier and Steedman (2007)) with deeper linguistic insights:
  - ◦ Propbank roles (Boxwell and White (2008))
  - ◦ Derivational restructuring for punctuation analysis (White and Rajkumar (2008))
  - ◦ Head lexicalization for case-marking prepositions, named entity annotation, lemmatization

Design

- System organized as a pipline, with corpus conversion and grammar extraction splitinto separate steps to facilitate machine learning over the converted corpus.
- Each step controlled by a separate custom Ant (`http://ant.apache.org/`) task:
  1. Generate an XML version of CCGbank using a JavaCC parser.
  2. Apply a series of XSLT transforms to create a converted corpus (in the same XML format).
  3. Extract a grammar in OpenCCG (`http://openccg.sourceforge.net/`) format.

Corpus Conversion

Example Conversion Paths


CCGbank
PropCCGbank
Initial Transformations
Baseline Punctuation Transformations
Punctuation Transformations
Conjunction Transformations
Final Transformations
OpenCCGbank

Ant Target and File List

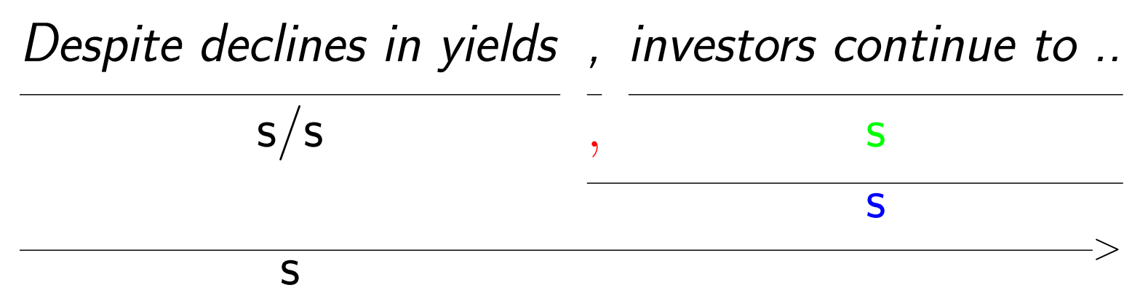
```
<target name="convert-puncts-baseline" depends="init-tasks">
  <convert target="${convert.dir}"
    wordsFile="${words}" stemsFile="${stems}">
    ...
    <templates>
      <filelist refid="convert-initial">
      <filelist refid="convert-orig-puncts">
      <filelist refid="convert-conj">
      <filelist refid="convert-final">
    </templates>
  </convert>
</target>
```

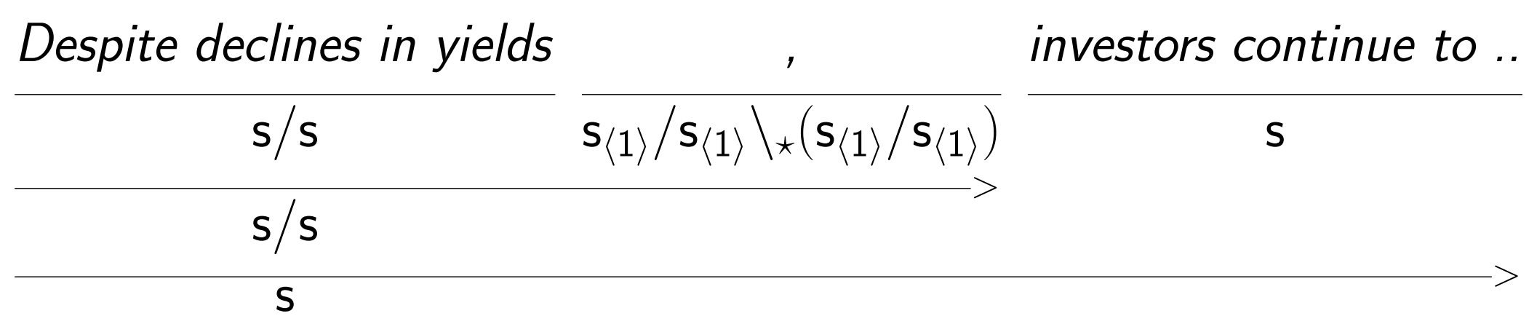
Example of a Filelist

```
<filelist id="convert-final" dir="${templates.dir}">
  ...
  <file name="adjustRoles.xsl"/>
  <file name="addStems.xsl"/>
  ...
</filelist>
```

Example XSLT Transformation

(1) Despite recent declines in yields, investors continue to pour cash into money funds. (wsj_0004.10)




Despite declines in yields
,
investors continue to ..
s/s
$s_{\langle 1\rangle}/s_{\langle 1\rangle}\backslash_{\star}(s_{\langle 1\rangle}/s_{\langle 1\rangle})$
s
s/s
s

XSLT Transform:

```
<!--Label comma which introduces a pre-sentential adjunct-->
<xsl:template match="Leafnode[@pos=',' and parent::Treenode/@cat0='S[dcl]' and following-sibling::Treenode/@cat0='S[dcl]']">
  ...
  </xsl:template>
```

Resulting category with discourse function semantics:

$, \vdash s_{\langle 1\rangle ind=X1, mod=M}/s_{\langle 1\rangle}\backslash_{\star}(s_{\langle 1\rangle}/s_{\langle 1\rangle}) : @_M(\langle \textsc{emph-intro}\rangle +)$

Implementation

- Advantages of converting the corpus using XSLT:
  - ◦ Our CCGbank translation and OpenCCG grammars are both in XML format.
  - ◦ No re-compilation required, as XSLT is interpreted.
  - ◦ Corpus conversion can be divided into as many XSLT transforms as desired (e.g., one for punctuation refactoring, one for derivation restructuring, etc.)
- We chose Ant for top-level process control because:
  - ◦ It allowed us to break the conversion and extraction steps into separate customizable Ant tasks.
  - ◦ Configuration requires no source editing or compilation, as code and configuration are separated.
  - ◦ Ant contains built-in support for JavaCC.
  - ◦ Ant's `FileSet` and `FileList` types allow flexible specification of sets of source files and series of XSLT transforms.
  - ◦ Both OpenCCG (whose libraries are used in grammar extraction) and Ant tasks are written in Java.

Experimental Impact

- System's flexibility allows a variety of different experiments to be performed.
- Ability to create corpora with various combinations of attributes.
- Enables extraction of training data for realization scoring and semantic dependency graphs (and features related to them).
- Our results have improved over time for section 23 of the CCGbank, including a state-of-the-art BLEU score of 0.8506 and the following single-rooted logical form (SRLF) performance:

Paper	*LF %*	*SRLF %*	*BLEU*
White et al. (2007)	94.3	69.7	0.5768
Espinosa et al. (2008)	96.1	76.7	0.6701
White and Rajkumar (2008)	96.46	92.12	0.7323
Current	**97.06**	**95.8**	**0.8506**

Acknowledgments

This work was supported in part by NSF grant no. IIS-0812297.

References

Stephen Boxwell and Michael White. Projecting Propbank roles onto the CCGbank. In *Proc. LREC-08*, 2008.

Dominic Espinosa, Michael White, and Dennis Mehay. Hypertagging: Supertagging for surface realization with CCG. In *Proc. ACL-08: HLT*, 2008.

Julia Hockenmaier and Mark Steedman. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, 2007.

Michael White and Rajakrishnan Rajkumar. A more precise analysis of punctuation for broad-coverage surface realization with CCG. In *Proc. of the Workshop on Grammar Engineering Across Frameworks (GEAF08)*, 2008.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. Towards broad coverage surface realization with CCG. In *Proc. of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT)*, 2007.