Rule-Based Chunking and Reusability

Claire Grover and Richard Tobin

School of Informatics
University of Edinburgh
{C.Grover, R.Tobin}@ed.ac.uk

Abstract

In this paper we discuss a rule-based approach to chunking implemented using the LT-XML2 and LT-TTT2 tools. We describe the tools and the pipeline and grammars that have been developed for the task of chunking. We show that our rule-based approach is easy to adapt to different chunking styles and that the mark-up of further linguistic information such as nominal and verbal heads can be added to the rules at little extra cost. We evaluate our chunker against the CoNLL 2000 data and discuss discrepancies between our output and the CoNLL mark-up as well as discrepancies within the CoNLL data itself. We contrast our results with the higher scores obtained using machine learning and argue that the portability and flexibility of our approach still make it a more practical solution.

1. Introduction

Chunking is frequently assumed to be a mature technology with machine learning systems, especially statistical ones, considered to provide adequate results. However, there remain reusability issues when porting to new domains, porting to new definitions of the chunking task or reconfiguring interactions between processing steps. In this paper we describe a rule-based approach founded on XML tools and argue that it provides a degree of flexibility with respect to reusability that makes it preferable to machine learning approaches. In Section 2 we introduce the chunking task. In Sections 3 and 4 we describe our XML text processing tools and our use of them in implementing a chunker. We present evaluation results in Section 5 and discuss the implications of our findings in Section 6.

2. Chunking

Chunking is a kind of shallow syntactic analysis where certain sequences of words in a sentence are identified as forming phrases of various types, such as noun phrases, verb phrases, adjective phrases etc. (Abney, 1991; Ramshaw and Marcus, 1995; Tjong Kim Sang and Buchholz, 2000). Typically, the structures are flat and non-recursive and not all words in a sentence need belong to a phrase. The phrases are truncated versions of typical phrase-structure grammar phrases where no post-head material, whether argument or adjunct, is included. Since the phrases do not correspond to the traditional phrase-structure phrases, they are sometimes referred to as 'groups' and we will follow this convention here. To illustrate, the following sentence has been divided into noun, verb, prepositional and adjective groups. Punctuation and the initial "But" are not included in a group.

> But $_{NG}$[the rapid spread] $_{PG}$[of] $_{NG}$[avian flu] $_{VG}$[may seem] $_{AG}$[unlikely] .

In more complex examples there are many decisions to be made about where the boundaries of a group should lie and, as a consequence, there are many different 'styles' of chunking. We raise issues relating to chunking styles when we discuss evaluation in Section 5.

Chunking is a useful level of analysis for a number of tasks. It is closely related to the task of Named Entity Recognition (NER) and can contribute to improving the performance of NER taggers (Zhou and Su, 2002). With additional access to information about the heads of groups it can be used as a computationally inexpensive way of getting to a shallow semantic representation (Abney, 1996; McNeill et al., 2006). Chunking systems may be rule-based (Abney, 1996; Finch and Mikheev, 1997) or based on machine learning. Machine learning chunkers may be symbolic (Déjean, 2000; Johansson, 2000; Vilain and Day, 2000) or statistical (the majority of recent systems).

3. XML Tools for NLP

Over the past few years we have developed a suite of tools for generic XML manipulation (LT-XML, Thompson et al., 1997) as well as NLP specific XML tools (LT-TTT, Grover et al., 2000; LT-CHUNK, Finch and Mikheev, 1997). More recently we have been developing significantly improved upgrades of these tools, LT-XML2 and LT-TTT2[1].

The LT-XML2 distribution contains a number of very generic XML tools which are extremely useful connecting programs in pipelines of NLP-oriented processors. The programs are written in C and are generally very fast. In particular, they do not have the start-up costs of many Java programs and are therefore suitable for use in NLP pipelines that stream data through many such programs. A common feature of the programs is the use of standard XML technologies such as XPath. The LT-XML2 tools include *lxgrep*, an XML grep program; *lxsort*, which sorts elements in an XML file; *lxaddids*, which adds XML ID attributes to chosen elements; *lxt*, a fast XSLT 1.0 implementation and *lxreplace*, which performs small XML transformations such as the replacement of an attribute name or removal of chosen XML element tags.

The LT-TTT2 tools are a significant reimplementation of the previous toolset, with the old core program *fsgmatch* replaced by a new program, *lxtransduce*. As before, this

[1] Soon to be available under GPL from `http://www.ltg.ed.ac.uk/`

```
<s>
  <ng>
    <w p='JJ'>Australian</w> <w headn='yes' p='NNS'>scientists</w>
  </ng>
  <vg tense='pres' voice='act' asp='perf' modal='no'>
    <w p='VBP'>have</w><w p='VBN' headv='yes'>discovered</w>
  </vg>
  <w p='WRB'>why</w>
  <ng>
    <w p='JJ'>ancient</w> <w headn='yes' p='NNS'>plesiosaurs</w>
  </ng>
  <vg tense='past' voice='act' asp='simple' modal='no'>
    <w p='VBD' headv='yes'>lived</w>
  </vg>
  <w p='IN'>for</w> <w p='RB'>so</w> <w p='JJ'>long</w><w p='.'>.</w>
</s>
```

Figure 1: Example of Chunking Pipeline Output

performs rule-based transductions of XML structures using hand-written grammar rules but this implementation is more efficient, uses XPath queries and has a grammar rule syntax which is more transparent. The previous part-of-speech (POS) tagger, *ltpos*, has been replaced by a more generic mechanism to integrate off-the-shelf non-XML POS taggers which use CoNLL column format for training and runtime, e.g. the C&C POS tagger, (Curran and Clark, 2003). This has the advantage of allowing the tagger to be retrained on new data or allowing a choice of taggers. The LT-TTT2 tools will be distributed with pipelines and associated *lxtransduce* rule files for tokenisation, sentence splitting, POS-tagging, noun and verb group chunking and MUC-style NUMEX and TIMEX named entity recognition (Chinchor, 1998). The release replaces the previous distributions of LT-TTT, LT-CHUNK and LT-POS.

4. Rule-based Chunking

Our current chunking pipeline recognises noun groups and verb groups, but not prepositional, adjective or other groups. (The addition of other types of group is a planned extension.) By default the pipeline uses our tools to perform tokenisation, sentence splitting and POS tagging prior to chunking but it can also be used just for chunking if the input has already been processed appropriately and can be mapped to the XML input format that the chunking rules expect. (We performed such a mapping when evaluating on the CoNLL data, see Section 5.) The chunking rule files are targeted at sentence elements via an XPath query and are applied in two stages, first the verb group rules and then the noun group rules. Figure 1 shows example output from the chunking pipeline.

Note that in addition to noun and verb group identification, other features have been computed, namely identification of head nouns and verbs as well as a number of features on verb groups to capture information about tense, aspect, voice and modality. The addition of these features was achieved with very little extra overhead—the rules which identify word sequences as noun and verb groups also specify the annotations to be added to them and a linguist defining the rules can easily specify the extra features. To illustrate, Figure 2 shows the rule, *simplepastpass*, which is responsible for recognising past tense passive verb groups such as "was sent". The `wrap` attribute on a rule is an instruction for the results of the match to be wrapped in a new XML element, in this case `vg`, and the `attrs` attribute next to it determines what attributes the `vg` element will have. The `seq` element indicates that a sequence of elements must be matched, each of which is described by a reference (`ref`) to another rule. The required parts of the sequence are a past tense form of the verb "be" and a passive participle. Optional parts are achieved through a Kleene star mechanism (`mult='*'`) on the call to a rule for adverbials between the auxiliary and the participle ("was probably sent") and a similar call to a rule for adverbs or particles after the participle ("was sent away"). The passive participle is marked as the head of the verb group by means of the `attrs` attribute on the call to the *headverb-pass* rule, which is also shown in Figure 2. This rule contains a disjunction (`first`) of match queries de-

```
<rule name="simplepastpass" wrap="vg"
   attrs="tense='past' voice='pass'
   asp='simple' modal='no'">
 <seq>
  <ref name="be-past"/>
  <ref name="adv" mult="*"/>
  <ref name="headverb-pass"
    attrs="headv='yes'"/>
  <ref name="advorpart" mult="*"/>
 </seq>
</rule>

<rule name="headverb-pass">
 <first>
  <query match="w[@p='VBN']"/>
  <query match="w[@p='VBD']"/>
 </first>
</rule>
```

Figure 2: Example *lxtransduce* Rules

scribing word elements (w) and their POS tag attributes (p). The POS tag must be VBN or VBD (passive participle or past participle—both must be considered because POS taggers do not reliably make the distinction).

5. Evaluation

To evaluate our chunker's performance we have been working with the data provided for the CoNLL 2000 shared task (Tjong Kim Sang and Buchholz, 2000). This data was automatically derived from the Penn Treebank II corpus (Marcus et al., 1993) by flattening the syntactic trees into a sequence of chunks interspersed with non-chunk words. Here we disregard mark-up for chunks other than noun and verb groups. Our chunking rules were developed to reproduce the behaviour of our earlier system, LT-CHUNK, and there are differences in style between this and the CoNLL data. The two that have the biggest impact on evaluation are:

- Control verb groups and their complements are one group in the CoNLL data but separate groups for LT-CHUNK, e.g. CoNLL: <vg>want to go</vg> vs. LT-CHUNK: <vg>want</vg> <vg>to go</vg >;
- Possessive noun groups are treated differently, e.g. CoNLL: <ng>Tom</ng><ng>'s book</ng> vs. LT-CHUNK: <ng>Tom's book</ng>.

5.1. Initial System

In order to achieve a realistic score against the test data, we made minor modifications to the chunker to produce CoNLL-style mark-up for the two cases above. Evaluation on the noun and verb groups yields the results shown in Table 1. Our overall F-score of 89.18% compares favourably with some of the systems which took part in the CoNLL 2000 shared task (Vilain and Day (2000): 85.76%, Johansson (2000): 87.23%) but falls short of the best score achieved (Kudoh and Matsumoto (2000): 93.48%).

	Precision	Recall	F-score
Noun Group	89.21%	88.57%	88.89%
Verb Group	88.10%	91.86%	89.94%
Total	88.89%	89.47%	89.18%

Table 1: Evaluation of initial chunker tested on CoNLL 2000 test set.

Error analysis on the training data has revealed a number of classes of mismatch between our output and the gold standard. These are cases where there are errors either in the CoNLL data or in our output, as well as cases of genuine disagreement. Below we give illustrative examples falling into three groups.
The first group of mismatches can be attributed to errors and inconsistencies in the CoNLL data:

1(a) Errors arising from POS mis-tagging: e.g. "sound" mistagged as a noun in "any further drop in the government's popularity could swiftly make this promise sound hollow".

1(b) Inconsistency in the CoNLL markup of adverbs: adverbs are normally only included in verb groups if they occur between two verbs in the group, however verb group-initial adverbs are included consistently after conjunctions and inconsistently elsewhere.

1(c) Occasional violations of the CoNLL possessive style described above where the possessive marker is included with the 'possessor' noun group rather than the 'possessed': e.g. <ng>next year 's</ng><ng>first half</ng>.

1(d) Mistakes in attachment of prenominal adjective phrases. There are also several instances, such as the following, where the error is compounded by forcing a violation of the CoNLL possessive style:
<ng>The merged agency 's</ng> admittedly ambitious <ng>goal</ng>.

1(e) Inconsistent treatment of coordination: e.g. "Singapore, Sydney, Taipei, Wellington, Hong Kong and Manila" is marked up as six individual noun groups whereas "Frankfurt, Zurich, Paris and Amsterdam" is treated as one noun group.

The second group of mismatches are the result of errors or omissions in our chunker's rules, or of wrong resolution of ambiguity:

2(a) Our rules do not cover lone possessive markers and fail to recognise them as part of the preceding noun group in the way the CoNLL data does:
<ng>an agent of Mr. Jackson 's</ng>
<ng>Moody 's</ng> said ..."

2(b) Certain verbs or verb+preposition multi-word expressions are treated as prepositions in the CoNLL data but are recognised as verb groups in our output: e.g. "combined with", "given", "including".

2(c) Present participle verbs forms (ending "-ing") are highly ambiguous between verb, adjective and noun analyses and our output contains some misanalyses. For example we wrongly classify "weakening" as a verb group in "weakening rail-traffic levels", though a similar analysis of "reflecting" in "reflecting higher costs" is the correct one.

The third group of mismatches are cases where neither analysis is 'wrong' but where we prefer a different mark-up to that found in the CoNLL data.

3(a) The word "not" is not included in single word verb groups in the CoNLL data but is included in our output ("is not").

3(b) Auxiliaries and modals preceding the subject in inverted structures are not marked as verb groups in the CoNLL data but are marked in our output: e.g. "could" in "How could I be so stupid?".

In response to the error analysis, there are a number of steps which could be taken to improve our score against

- ... in <ng><TIMEX type='date'>February 1974 </TIMEX></ng>, it was priced at <ng><NUMEX type='money'>$10<NUMEX></ng> a share ...
- <ng>an <TIMEX type='date'>Oct. 19</TIMEX> letter</ng>
- <ng>a further <NUMEX type='percent'>25%</NUMEX></ng>
- <ng>a <NUMEX type='money'>$ 300 million</NUMEX> sale</ng>
- <ng>the <TIMEX type='date'>1987</TIMEX> market crash</ng>

Figure 3: Noun Groups Containing NUMEX and TIMEX Entities

the CoNLL data. For the examples in (3), we could abandon our analysis in favour of the CoNLL style of mark-up. We have decided against doing this just to improve our score since we prefer the more linguistically motivated mark-up encoded in our rules. (We would of course be able to make appropriate changes if it became clear that the CoNLL mark-up was preferable.) For our errors in (2) we can and will continue to revise our rules and hope gradually to improve our performance. The ideal solution to the errors in the CoNLL data in (1) would be a manual correction of the data but resources for this are not available and there is little we can do except note that it is frustrating to have correct analyses scored as incorrect because of errors in the gold standard.

5.2. POS Tag Errors

Manual correction of the errors in the CoNLL data is not an option but we can address the issue of POS tag errors as in (1a) in an automatic way. The CoNLL data does not contain the original POS tags from the Penn Treebank. Instead the organisers decided to retag it with the Brill tagger (Brill, 1994) "in order to make sure that the performance rates obtained for this data are realistic estimates for data for which no treebank POS tags are available" (Tjong Kim Sang and Buchholz, 2000). Machine learning systems have an advantage over rule-based ones in that they can overrule a POS tagging decision wherever other features strongly suggest that they should. By contrast, rule-based systems are reliant on good POS tagging and will perform poorly on badly tagged data. (Similarly, a machine learning chunker trained on perfectly tagged data might also perform poorly on a badly tagged test set.) To address the problem of tagging errors, we retagged the CoNLL data using the C&C POS tagger (Curran and Clark, 2003), using a model trained on the Penn Treebank. The results of testing our chunker on the retagged data are shown in Table 2 and they show nearly a 1% increase in performance. Investing effort into improving POS tagger performance therefore seems likely to yield further benefits.

	Precision	Recall	F-score
Noun Group	90.04%	89.18%	89.61%
Verb Group	90.16%	92.42%	91.28%
Total	90.07%	90.06%	90.07%

Table 2: Evaluation of initial chunker tested on *retagged* CoNLL 2000 test set.

At this point it is worth noting that although our rules are dependent on accurate POS tagging they have been designed to address certain systematic tagging errors that all POS taggers are prone to. For example, when verbal forms occur as adjectives or nouns they are sometimes still tagged as verbs (e.g. "hearing" in "a congressional hearing"; "marked" in "a very marked improvement"). Conversely, "-ing" forms are sometimes incorrectly tagged as nouns when they are actually verbs ("bickering" in "It's time to stop bickering"; "unwinding" in "traders who were unwinding positions"). Our rules attempt to rectify these kinds of mistakes wherever the context makes it clear that they are errors, though there are some cases where the context is insufficient. In predicative contexts the difference between adjective and verb is not always clearcut and either analysis would be acceptable, e.g. "divided" in "delegates were evenly divided" which is tagged as a verb and treated as part of an adjective group in the CoNLL data but which we mark up as part of a verb group.

5.3. Extended System

Error analysis on the training set revealed a class of errors which we have not discussed up until this point, namely, errors in noun groups relating to dates and numerical expressions. Since the CoNLL data is typical newspaper text, it contains large numbers of dates, times, percentages, sums of money etc. These expressions are typically treated as named entities, for example in MUC (Chinchor, 1998) they are treated as NUMEX and TIMEX entities. Many NUMEX and TIMEX entities are complete noun groups (as in the first example in Figure 3), however in many cases they are sub-parts of noun groups (e.g. the remainder of the examples in Figure 3).

While the existing noun group rules deal successfully with the majority of these cases, there are instances where the rules are insufficient. It would be possible to alter and extend the existing rules to deal with these cases but this would be a duplication of effort because we have already developed grammars for numbers and the MUC NUMEX and TIMEX entities (rule files that were part of LT-TTT ported to the new formalism of LT-TTT2). So instead we chose to add initial steps to our chunking pipeline to perform NUMEX and TIMEX named entity recognition before applying the verb and noun group rules. We then made small adjustments to the noun group rules so that they could match previously identified NUMEX and TIMEX entities, either on their own or as part of a larger sequence. The eval-

uation results for the extended system are shown in Table 3. Table 4 shows results for the combination of the extended system with retagging to address POS tag errors.

	Precision	Recall	F-score
Noun Group	89.51%	88.90%	89.20%
Verb Group	88.14%	91.93%	90.00%
Total	89.12%	89.72%	89.42%

Table 3: Evaluation of extended chunker tested on CoNLL 2000 test set.

	Precision	Recall	F-score
Noun Group	90.39%	89.55%	89.97%
Verb Group	90.18%	92.49%	91.32%
Total	90.34%	90.35%	90.34%

Table 4: Evaluation of extended chunker tested on *retagged* CoNLL 2000 test set.

As the tables show, the raw results seem to indicate that there is not a great deal of difference between the initial and the extended systems. Despite this apparent lack of progress, we prefer the output of the extended system for two reasons. First, although we discarded the NUMEX and TIMEX mark-up in order to evaluate against the test set, the XML output of our pipeline contains this mark-up nested inside the noun group mark-up and our extended system output is therefore more informative than it was before. A second reason to prefer the extended system is that there is more of an improvement in accuracy than the raw scores suggest: the addition of NUMEX and TIMEX leads to a further divergence in chunking style between our output and the CoNLL data, which is penalised in the scoring mechanism. To illustrate, the MUC TIMEX rules treat date-of-date sequences as one TIMEX while the CoNLL chunk data treats these as two noun groups, Furthermore, the CoNLL data does not include words such as "ago" and "earlier" at the end of date noun groups while in MUC they are part of the TIMEX:

MUC:
<TIMEX type='date'>the second quarter of 1988</TIMEX>
<TIMEX type='date'>four years ago</TIMEX>
CoNLL:
<ng>the second quarter</ng> of <ng>1988</ng>
<ng>four years</ng> ago

In both these cases we are happy to treat the TIMEX entities as noun groups and do not wish to adjust our rules to the CoNLL style simply to make a gain in our score.

6. Reusability

In the previous section we gave a detailed analysis of the performance of our chunker against the CoNLL test set. The aim of providing this amount of detail was to show that this method of evaluation is not as straightforward or clearcut as it may initially seem. We discussed the fact that the CoNLL test set contains a large number of errors and inconsistencies and that this can lead to our chunker being penalised when it is, in fact, correct. The large number of POS tag errors has a detrimental effect on our score, and we addressed this in part by retagging the data with a more accurate tagger (see Tables 2 and 4).

In general our concern is to produce a chunker which is reusable in the sense that it should not be time-consuming or difficult to configure it to new domains and applications. At the same time, we need to show that our chunker's performance in terms of accuracy is competitive with other state-of-the-art chunkers, and we have attempted to do this by evaluating on the CoNLL 2000 test set. This has raised issues about what it means to score a system against one particular data set and what it tells us about the relative virtues of one system compared to another. We have drawn a distinction between rule-based systems, such as ours, on the one hand, and machine learning systems on the other[2]. A survey of the test results for the CoNLL 2000 competition indicates that machine learning systems can significantly outperform our chunker. However, this is really only true where the training material and the test material are of exactly the same type and quality. If a machine learning system trained on the CoNLL training material was tested on a test set with a different style of mark-up its score would be reduced. A choice between rule-based and machine learning systems will therefore ultimately depend on the availability of appropriately annotated data: if there is sufficient high-quality POS-tagged and chunk-annotated data then it will be quicker to train a machine learning system than to develop hand-written rules from scratch. However, the cost of preparing such ideal training data should be factored into the equation. If only inconsistently and inaccurately annotated data is available, a machine learning system will learn to model the inconsistencies and inaccuracies and will achieve high scores on test material that has the same properties as the training material. However the mark-up it assigns on unseen data will only be as good as the mark-up in its training data.

Portability and reusability are important issues which have bearing on the question of machine learning versus rule-based systems. It is not currently clear how much the chunking task differs across domains: for example, will a chunker developed for newspaper text work adequately for biomedical text? It seems possible that the two domains do differ at least in the distribution of subtypes of chunk. Informally one can observe a high degree of complex nominal structures in biomedical text (e.g. "rotenone-blocked submitochondrial particles") while in newspaper text one can observe more phrases containing temporal expressions (e.g. "this week's trade figures"). Our treatment of the CoNLL newspaper text includes the insertion of a layer of NUMEX and TIMEX recognition before chunking; when moving to

[2]The distinction is not as clearcut as we may seem to imply: many hybrid systems combine machine learning with rules and symbolic machine learning techniques such as Brill-style transformation-based learning produce transparent rules which can be adjusted by hand. One of the CoNLL 2000 entrants (Déjean, 2000) created a system for learning rules which were then translated for parsing into our LT-TTT rule format: these too would be editable by a human rule writer.

biomedical text we can probably discard this layer (and reduce processing costs) since it is likely to be extraneous. If it becomes clear that a chunker developed for one domain is performing badly on another domain then the remedy will depend on the type of chunker: a machine learning chunker will need costly annotated training material while a rule-based system will need relatively minor modifications. In a similar vein, if a particular application requires a particular style of chunking (e.g. long verb groups vs. short ones as described above) then a machine learning system needs to be retrained on corrected data while a rule-based system requires a very minor alteration.

A further issue concerns embedded mark-up. Recently, since Ramshaw and Marcus (1995), chunking has predominantly been treated as a tagging problem but this does not allow for recursive structures. In our rule-based approach we are able to nest chunks quite straightforwardly and, indeed, our method of switching between chunking styles for possessives and long verb groups involves first building a nested structure and then deleting either the inner or outer chunk mark-up depending on the style we are producing. For example, we bracket a long verb group like this: *((may not have decided) (to leave))* and for CoNLL-style output we delete the inner mark-up while for LT-CHUNK-style output we delete the outer. If a particular application calls for it, we can modify our rules to produce more nested mark-up thereby bringing our chunker closer to a partial parser such as Abney's CASS (Abney, 1996).

In the previous section we showed how applying NUMEX and TIMEX recognition before chunking is useful for newspaper text and we gave examples that indicated that this had resulted in a useful style of chunking for noun groups containing dates and numerical expressions. There is an issue about the order in which components such as chunkers and named entity recognisers should apply. It is generally assumed that chunking is a stage to be performed before named entity recognition, and this may well be the case for person, organisation and location names. However, our experience has shown the benefits of performing at least some named entity recognition before chunking and our use of the LT-XML2 and LT-TTT2 tools has provided a practical means of exploring the interplay between levels of analysis.

7. Acknowledgements

This work was supported in part by a Scottish Enterprise Edinburgh-Stanford Link Grant (R37588), as part of the EASIE project. We would like to thank Ewan Klein for comments on the chunker output and on drafts of this paper. This work has built on the efforts of all those involved in the development of earlier versions of our software, LT-XML, LT-TTT and LT-CHUNK, in particular Andrei Mikheev.

8. References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-Based Parsing*. Kluwer Academic Publishers.

Steven Abney. 1996. Partial parsing via finite-state cascades. *Journal of Natural Language Engineering*, 2(4):337–344.

Eric Brill. 1994. Some advances in rule-based part-of-speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*.

Nancy A. Chinchor. 1998. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Fairfax, Virginia.

James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL-03)*, pages 91–98. Budapest, Hungary.

Hervé Déjean. 2000. Learning syntactic structures with XML. In *Proceedings of the Conference on Natural Language Learning (CoNLL-2000)*.

Steve Finch and Andrei Mikheev. 1997. A workbench for finding structure in texts. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*. Washington D.C.

Claire Grover, Colin Matheson, Andrei Mikheev, and Marc Moens. 2000. LT TTT—a flexible tokenisation tool. In *LREC 2000—Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pages 1147–1154.

Christer Johansson. 2000. A context sensitive maximum likelihood approach to chunking. In *Proceedings of the Conference on Natural Language Learning (CoNLL-2000)*.

Taku Kudoh and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of the Conference on Natural Language Learning (CoNLL-2000)*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2).

Fiona McNeill, Harry Halpin, Ewan Klein, and Alan Bundy. 2006. Merging stories with shallow semantics. In *Proceedings of KRAQ'06 (Knowledge and Reasoning for Language Processing) Workshop at EACL 2006*.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*. Cambridge MA, USA.

Henry Thompson, Richard Tobin, David McKelvie, and Chris Brew. 1997. LT XML. software API and toolkit for XML processing. `http://www.ltg.ed.ac.uk/software/`.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the Conference on Natural Language Learning (CoNLL-2000)*. Lisbon, Portugal.

Marc Vilain and David Day. 2000. Phrase parsing with rule sequence processors: an application to the shared CoNLL task. In *Proceedings of the Conference on Natural Language Learning (CoNLL-2000)*.

Guodong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of ACL'02*, pages 473–480.