Applying Transformation-Based Error-Driven Learning to Structured Natural Language Queries

Alan Woodley, Shlomo Geva
School of Software Engineering and Data Communications
Faculty of Information Technology
Queensland University of Technology
ap.woodley@student.qut.edu.au, s.geva@qut.edu.au

Abstract

XML information retrieval (XML-IR) systems aim to provide users with highly exhaustive and highly specific results. To interact with XML-IR systems, users must express both their content and structural requirement, in the form of a structured query. Traditionally, these structured queries have been formatted using formal languages such as XPath or NEXI. Unfortunately, formal query languages are very complex and too difficult to be used by experienced, let alone casual users. Therefore, recent research has investigated the idea of specifying users' content and structural needs via natural language queries (NLQs). In previous research we developed NLPX, a natural language interface to an XML-IR system. Here we present additions we have made to NLPX. The additions involve the application of transformation-based error-driven learning (TBL) to structured NLQs, to derive special connotations and group words into an atomic unit of information. TBL has successfully been applied to other areas of natural language processing; however, this paper presents the first time it has been applied to structured NLQs. Here, we investigate the applicability of TBL to NLQs and compare the TBL-based system, with our previous system and a system with a formal language interference. Our results show that TBL is effective for structured NLQs, and that structured NLQs a viable interface tor XML-IR systems.

1. Introduction

Information retrieval (IR) systems respond to user queries with a ranked list of relevant results. Traditionally, these results have been whole documents but since XML documents separate content and structure, XML-IR systems are able to return highly specific information to users, lower than the document level. However, to take advantage of this capability XML-IR users require an interface that is powerful enough to express their content and structural requirements, yet user-friendly enough that they can express their requirements intuitively.

Historically, XML-IR systems have used two types of interfaces, keyword-based and formal query language-based. Keyword-based systems are user-friendly, but lack the sophistication to properly express users' content and structural needs. In comparison, formal query language-based interfaces are able to express users' content and structural needs, but are too difficult to use, especially for casual users [24,27] and are bound to the physical structure of the document. Recently, investigation has begun into a third option for interfacing with XML-IR systems via a natural language interface that will allow users to fully express their content and structural needs in an intuitive and easy to use manner.

We have previously presented NLPX [29,30], an XML-IR system with a natural language interface. NLPX accepts natural language queries (NLQs) and translates them into NEXI queries. NEXI is an XPath-like formal query language that is used as a frontend to many existing XML-IR systems. NLPX participated in the natural language processing track of the 2004 INitiative for the Evaluation of XML Retrieval Workshop. INEX [14] is comparable to TREC and is a benchmark for the evaluation of XML-IR systems. INEX evaluates two types of queries: Content Only (CO) and Content and Structure (CAS). CO queries express users' information need solely in terms of a subject area, while CAS queries specify both a subject area and location within the document where the area might be present.

Proceedings of the 2005 International Conference on Cyberworlds (CW'05)
0-7695-2378-1/05 $20.00 © 2005 IEEE

Since the 2004 INEX workshop we have made two changes to NLPX. First, we replaced our existing method of identifying significant terms in a NLQ from basing them solely on their word value to a method that also takes into account a word's context. Second, we introduced a shallow parser that groups individual words into an atomic unit of information. Both of the changes are based upon an approach called transformation-based error-driven learning. This is an approach that has shown promise in other areas of natural language processing and we have investigated if it is also applicable to structured NLQs. Our results indicate that it is a beneficial addition.

The rest of this paper is organised as follows. Section 2 lists the motivation for development of a XML-IR natural language interface. Section 3 summarises our previous work. Section 4 describes the paradigm of transformation-based error-driven learning. Sections 5 and 6 detail the extensions we have made to our system. Sections 7 and 8 describe how our extensions interact with our existing system. Finally, Section 9 reports the results of our extensions, using the standard INEX 2004 dataset.

2. Motivation

We have already outlined the motivations for an XML-IR natural language interface in our previous work [29, 30]; however, for completeness we include them here. The motivations stem from the problems with formal XML-IR query languages and are two fold: first, formal query languages are difficult to use, and second, they are too tightly bound to the physical structure of documents.

First, formal query languages are too difficult for many users to correctly express their information need. Two very good examples of this have occurred at the 2003 and 2004 INEX Workshops. In 2003 INEX used the XPath [10] formal language to specify structured queries; however, 63% of the proposed queries had major semantic or syntactic errors. Furthermore, the erroneous queries were difficult to fix, requiring 12 rounds of corrections. In response to this problem, O'Keefe and Trotman [24] designed a simplified version of XPath called NEXI, which was used in INEX 2004. When NEXI was used, the error rate dropped to 12%, with the number of topic revision halved [27]. While these figures are limited to two formal languages, O'Keefe and Trotman investigated other structured query languages such as HyTime, DSSSL, CSS and XIRQL and concluded that all of them are very complicated and difficult to use. Therefore, if experts in the field of structured information retrieval are unable to correctly use complex query languages, one cannot expect an inexperienced user to do so. However, we feel that users would be able to intuitively express their information need in a natural language.

Secondly, formal query languages are too tightly bound to the physical structure of documents; hence, users require an intimate knowledge of documents' composition in order to express their structural requirements properly. So, in order for users to retrieve information from abstracts, bodies or bibliographies, they will need to know the actual names of those tags in a collection (for instance: *abs*, *bdy*, and *bib*). While this information may be obtained from a document's DTD or Schema there are situations where the proprietor of the collection does not wish users to have access to those files. Or, in the case of a heterogeneous collection, a single tag can have multiple names (for example: abstract could be named *abs*, *a*, or *abstract*). Alternatively, structural requirements in NLQs are expressed at a higher conceptual level, allowing the underlying document's structure to be completely hidden from users.

3. Previous Work by Authors

This paper expands on the previous work of the authors presented in [29, 30]. We submitted our system, NLPX, to the 2004 INEX Natural Language Processing Track where it performed very successfully (1st in CAS, 2nd in CO). INEX's NLP track used the same topics and assessments as its Ad-hoc track; however, participating systems used a natural language query as input, rather than a query a formal language (NEXI) query. Examples of both query types are expressed in Figure 1. Note that the query actually contains two information requests, first, for sections about compression, and second, for articles about information retrieval. However, the user only wants to receive results matching the first request. We refer to the former as returned requests/results and the latter as support requests/results.

NEXI: //article[about(.,'information retrieval')]
//sec[about(./, compression)]

NLQ: Find sections of articles about image and text compression in articles about efficient information retrieval

Figure 1. A NEXI and Natural Language Query

We had previously participated in INEX's Ad-hoc track with GPX, a system that accepted NEXI formatted queries. Therefore, we decided to use GPX as a backend system. This allowed us to concentrate on developing a frontend that translated natural language queries to NEXI. Translation involved three steps that derived syntactic and semantic information from the natural language query (NLQ). These three steps are outlined below:

1. First we tagged words in the NLQ as either a special connotation or by their part of speech. Special connotations are words of implied semantic significance. We differentiated between three types: Structures (such as section, abstract) that specified structural requirements, Boundaries (such as contains, about) that separated structural and content requirements, and Instructions (such as find, retrieve) that indicated if we had a return or support request. Words corresponding to special connotations were hard-coded into the system and matched to query words by a dictionary lookup. Remaining words were tagged by their part of speech (such as noun, verb, conjunction) via a Brill Tagger [4].
2. Second, we matched the tagged NLQ to query templates. The templates were derived from the inspection of previous INEX queries. Since the NLQs occurred in shallow context they required only a few templates, significantly less than if one wished to capture natural language as a whole. Each template corresponded to an information request. Each request had three attributes: Content, a list of terms/phrases expressing content requirements, Structure, a logical XPath expression expressing structural requirements, and an Instruction, "R" for return requests, and "S" otherwise.
3. Finally, the requests were merged together and output in NEXI format. Return requests were output in the form **A[about(.,C)]** where **A** is the request's structural attribute and **C** is the request's content attribute. When all return requests were processed, support requests were inserted. The insert position was located by comparing the structural attributes of return and support requests and by finding their longest shared descendant. The output of support requests had the form **D[about(E,F)]** where **D** is the longest matching string, **E** is the remainder of the support's structural attribute and F is the support's content attribute.

The rest of this paper concerns extensions we made to the first step, that is, the lexical and semantic tagging of the natural language query. Readers wanting extensive overviews of the other steps or the GPX backend are recommended to read our previous work.

4. Transformation-Based Error-Driven Learning

Transformation-based error-driven learning (TBL) has been applied to many areas of natural language processing such as part of speech tagging [4], propositional phrase attachment [5], shallow parsing [26], word sense disambiguation [21] and syntactic parsing [6]. TBL works by recognizing and remedying its weakness, thereby incrementally increasing its performance. Figure 2 was originally presented in [7] and illustrates the learning process.

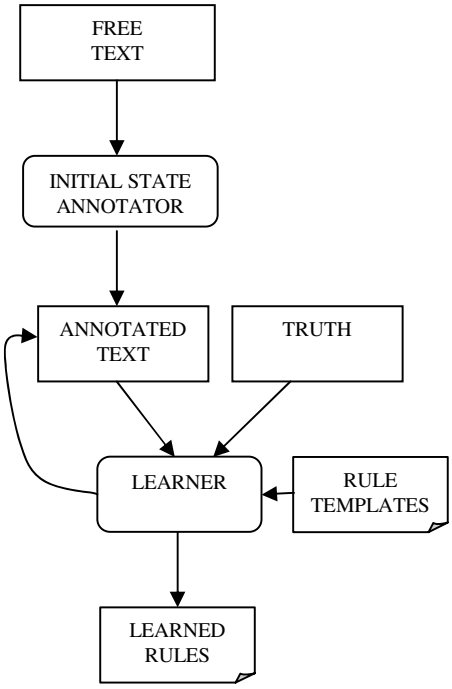

Figure 2: Transform-Based Learning

The paradigm is explained as follows:

1. A corpus is manually annotated with metadata. Examples of metadata are a word's part of speech or meaning. We refer to the manually annotated corpus as the truth, and the unannotated corpus as the free text.

2. The free text is input into an initial state annotator, which appends metadata to each word in the corpus. The initial state annotator can be as simple as assigning the same metadata value to each word, or as complex as assigning the output of a stochastic annotator.
3. The learner compares the annotated text with the truth. The learner applies a set of transformation templates to the annotated text to make it better resemble the truth. The transformation that best corrects the annotated text is saved to a rules file. This transformation is applied to the annotated text.
4. Step 3 is reiterated until no transformations can be found that improve the annotated text beyond some pre-specified threshold.

To learn a specific application of TBL one must specify: the initial state annotator, the transformation templates and the scoring function for comparing the annotated text to the truth, and choosing a transformation. Once a list of rules is learned new text can be annotated, first, by applying the initial state annotated, and then by applying each of the rules, in order.

5. Part of Speech Tagging

Part of speech tagging is the process of augmenting words in a corpus with their form grammatical class (for example noun; verb; adjective). Part of speech tagging is a fundamental first step in many computational linguistic applications. There exist many automatic part of speech taggers that can be used to tag a corpus, and they are loosely grouped into two classes: grammatical-based and statistical-based. Here, we describe the part of speech tagger we used and how we incorporated it into NLPX.

5.1. The Brill Tagger

Originally, automatic part of speech taggers used manually engineered grammatical rules [18, 16]. During the 1980s, large corpora became available and researchers developed trainable Markov-based stochastic taggers [8, 9, 11, 17, 20, 23]. These stochastic taggers were very accurate, but did not perform any syntactic analysis. Instead, they assigned a sentence the tag sequence that maximises *Prob(*word|tag*)* * *Prob(*tag|previous n tags*)*. The advantages of stochastic taggers over manually built taggers include eliminating the time-consuming manual rule construction and possibly capturing useful information that might have been missed by a human engineer. However, they have the disadvantage that linguistic information is calculated indirectly, in large tables of statistics.

Brill developed a grammatical-based tagger that performed comparably to stochastic taggers [4, 7]. This was a significant breakthrough, since previous grammatical-based taggers [15, 18] had error rates substantially higher than state of the art stochastic taggers. Training the tagger was fully automatic, and used Transformation-Based Learning (TBL). Thus, unlike stochastic taggers, linguistic information was captured directly in a set of rules, rather than large tables of statistics. Further improvements over stochastic taggers included better portability from one tag set or corpus genre to another and ease of finding and implementing improvements to the tagger.

Here we describe the algorithm for the final version of the Brill tagger [7], which extended his previous work [4].

1. The corpus is tagged by the initial state annotator. Known words are assigned the most likely tag as derived from the training corpus. Unknown words are naively tagged as nouns or as proper nouns if they begin with a capital.
2. Lexical based transformations are performed on unknown words derived from their suffixes, infixes and adjacent word co-occurrence.
3. Lexical and syntactic based transformations are performed on words using an ordinal set of templates, based upon from their neighbouring words and tags.

5.2. Application to NLPX

We choose to apply the Brill tagger within NLPX as opposed to a stochastic tagger for several reasons. First, it is easier to port to another genre than stochastic taggers. This is important since we were working with structured natural language queries, a genre that part of speech taggers are not specifically designed for. Second, it requires much smaller copra for training. This is important since structured NLQ copra currently suffer from a sparse data problem. We used approximately one hundred INEX queries during training, totally 1,500 words. However, tens of thousands of words are needed to train stochastic taggers Finally, the Brill tagger makes decisions based on features of localised context, this is fortunate, since the majority of our queries are relatively small (between 25-50 words).

The Brill tagger defines tags as specified by the Penn Treebank [22]. We augmented these tags with our own set of special connotations to specify structures (e.g. article, section, paragraph), boundaries (e.g. about, containing) and instructions (e.g. find, retrieve). Figure 3 presents some of the tags used in the Penn Treebank and Figure 4 presents the tags that denote a special connotation.

CC	Coordinating Conjunction
DT	Determiner
IN	Preposition / Subordinating Conjunction
JJ	Adjective
NN	Noun, Singular or Mass
NNS	Noun Plural

Figure 3: Some Penn Treebank Tags

XIN	Instruction Word
XST	Structural Word
XBD	Boundary Word

Figure 4: Special Connotation Tags

The Brill tagger was used in our previous work [29, 30]; however, it was only used to define standard parts of speech. Special connotations were transformed at a later stage by a simple dictionary lookup. The dictionary was hard-coded to contain words assumed to be special connotations. So after annotating our natural language query (NLQ) using the Brill tagger, the dictionary was searched for each query word. If the query word was found in the dictionary, then its tag was transformed into a special connotation regardless of its context. This caused problems for ambiguous words. For instance, in both of the following NLQs the word abstract would be tagged as a structural connotation, although the second instance should remain tagged as a noun.

*Retrieve articles' titles from documents with **abstracts** that discuss personal privacy concerns.*

*Retrieve figure captions about **abstract** paintings.*

However, we could correct this error by using the following rule.

Change the tag from **structure** to **noun** if the previous tag is **boundary**.

To correct this problem we extended the Brill tagger to classify special connotations as well as parts of speech. This required us to retrain the Brill tagger using the process discussed in section 4. Our training corpus consisted of manually annotated INEX queries. Each year INEX participants propose a set of topics, called candidate topics, to be used as evaluate systems, and each year a subset is chosen as official INEX topics. For training we used the INEX 2003 CAS candidate topics and the 2004 INEX candidate topics not chosen as official topics. Our test set consisted of the INEX 2004 official topics. Figure 5 is the NLQ introduced earlier, after it has been tagged.

Find/XIN sections/XST of/IN articles/XST about/XBD image/NN and/CC text/NN compression/NN in/IN articles/XST about/XBD efficient/JJ information/NN retrieval/NN

Figure 5: Tagged NLQ

6. Shallow Parsing

Shallow parsing, also called text chunking, is the process of dividing sentences into atomic, non-overlapping segments (called chunks), and then classifying them into grammatical classes. Shallow parsing is usually performed after part of speech tagging, and as demonstrated by Abney [1] can be used as a precursor to full parsing. Alternatively, it can be used in other tasks such as index term generation, information extraction, text summation and bilingual alignment. Initial research into shallow parsing focused on identifying noun phrases; however, more recent work has extended its reach to include general clause identification. Here we describe the shallow parser we used, and how we incorporated it into NLPX.

6.1. Ramshaw and Marcus

Initial shallow parsing techniques focused on identifying low-level noun chunks, either using linguistic techniques that combined lexical data with finite state or other grammar constraints [3, 28], or stochastic techniques that automatically constructed a language model from a labelled and bracketed corpus [9]. As pointed out by Pla *et al.* [25] it is difficult to

compare the effectiveness of these techniques since each had different definitions of what constituted a valid chunk, used different test collections or even performed evaluation manually. However, Ejerhed [13] reported that the stochastic techniques outperformed linguistics techniques both in terms of identifying noun phrases (97.8% to 93.5%) and general clauses (98.6% to 97.8%).

Ramsaw and Marcus [26] conducted landmark research that approached shallow parsing as if it was a tagging task. While previous methods encoded chunks using non-recursive bracket markers Ramshaw and Marcus encoded chunks using a separate tag, so that each word had both a part of speech tag and a chunk tag. They used the chunk tag set {I,O,B} where I indicated that words were in a chunk, O indicated that words were outside of a chunk and B indicated that words were inside a chunk, but the preceding word was in another chunk. This approach was advantageous since chunk structure could be derived solely from tag sequence, whereas methods using the previous encodings had the additional complexity of correctly pairing brackets. This new method of encoding meant that many more machine learning algorithms could be applied to shallow parsing than previously thought [2, 12, 19].

Like Brill, Ramsaw and Marcus applied a transform-based learning algorithm. Their initial state annotator assigned the most likely chunk tag according to a word's part of speech. During their training phrase they used 100 templates built from the cross product between 20 word and part of speech patterns and 5 chunk tags patterns. While Ramshaw and Marcus were the first to envision shallow parsing as a tagging task, research by Pla *et al.* indicates that their approach is not as accurate as later approaches. However, we wanted to investigate how well the transformation-based learning paradigm would apply to structured natural language queries; hence, we decided to use both a TBL tagger and shallow parser.

6.2. Application to NLPX

We waned to use Ramshaw and Marcus' shallow parser to recognise three types of chunks: instruction chunks, structure chunks and content chunks. Not surprising these are logical extensions of the same three special connotations introduced earlier. In particular, recognising content chunks allowed us to perform other interesting lexical analysis by deriving phrases based on the nouns, verbs and adjectives that occur in the chunk. In order to recognise these chunks we had to retrain Ramshaw and Marcus' shallow parser. We input the same training data used to retrain the Brill tagger, but it was manually annotated with the specified chunk tags (I,O,B). We also used the same test set that we did for the Brill tagger. From this process, the Ramshaw and Marcus' shallow parser was able to learn several new rules to identify chunks. Figure 6 is what the NLQs introduced earlier would look like after processing by the shallow parser.

```
[ Find/XIN ] [ sections/XST of/IN articles/XST]
about/XBD  [ image/NN and/CC text/NN
compression/NN ] in/IN [ articles/XST ]
about/XBD [ efficient/JJ information/NN
retrieval/NN ]
```

Figure 6: Chunked NLQ

7. NLPX Backend

Once the NLQ is tagged and chunked it is transformed into a NEXI query using the existing NLPX system. This is a two stage process. First, information requests are derived by matching the NLQ to a set of query templates outlined in Figure 7. Then the information requests are merged, and output in NEXI format. Figure 8 shows a NEXI query derived from the earlier NLQ. Notice that the query has been expanded to included several phrases that do not occur in the actual NLQ, based on the grammatical properties of each phrase.

Query: Request+
Request : CO_Request | CAS_Request
CO_Request: NounPhrase+
CAS_Request: SupportRequest | ReturnRequest
SupportRequest: Structure [Bound] Content+
ReturnRequest: Instruction Structure [Bound] Content+

Figure 7: Query Templates

```
//article[about(.,'efficient information retrieval'
'information retrieval')] //sec[about(./, 'image text
compression')]
```

Figure 8: NLQ-to-NEXI Queries

8. GPX Backend

When the NLQs have been translated into NEXI format we send them to our existing GPX system for processing. GPX accepts NEXI queries, and returns a ranked list of XML elements. To produce results GPX collects leaf elements from its index and dynamically creates their ancestors. GPX's ranking scheme rewards leaf elements with specific terms and penalises leaf elements with common terms. It also rewards ancestors with multiple relevant children and penalises ancestors with a single relevant child. Finally, phrases are heavily rewarded, where an occurrence of a phrase in a result is defined as all phrase words in the query occurring in the leaf element, even if they do not occur continuously.

9. Results

We conducted our test experiments using the 2004 INEX dataset. INEX is comparable to TREC and is the most authoritative benchmark for XML retrieval. The INEX collection consists of a set of IEEE journal articles, topics, relevance assessments and an evaluation module. INEX accepts two types of topics: Content Only (CO) and Content and Structure (CAS). Both types contain hints about a user's requested subject matter (content); however, CAS topics are referred to as structured queries since they also contain hints about the elements that are most likely to satisfy a user's information need. Here we present the results of the CAS topics. We preformed four experiments, (1) the NLPX system that incorporated the TBL additions; (2) an upper-bounds baseline that simulated a 'perfect' TBL system, were manually tagged and chunked NLQs were submitted to NLPX; (3) a baseline that used NLPX without TBL and (4) a second baseline that used NEXI input into GPX.

Figure 9 shows the Recall-Precision plots for our experiments. There are four lines of significance. The dashed line is the NLPX baseline, without TBL (NLPX04), the solid line is the new NLPX system with TBL (NLPX-TBL), the dotted line is the NLPX system with perfect input (NLPX-Perfect), and the dashed-dotted line is the GPX system using NEXI input (NEXI). The grey lines are the plots of the other participants. Table 1 shows the Mean Average Precision of all runs and the system rank they would have received if they had participated in the INEX 2004 Ad-hoc task.

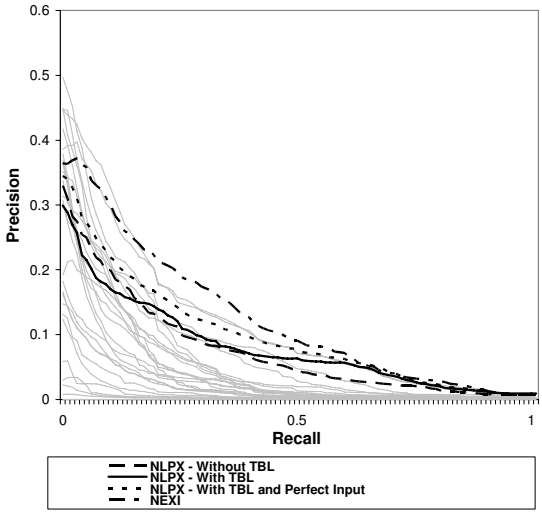

Figure 9: Recall-Precision Curve

	NLPX-04	NLPX-TBL	NLPX-Perfect	NEXI
MAP	0.757	0.0804	0.981	0.1242
INEX Rank	7	5	5	1

Table 1. Mean Average Precision

The results presented here are interesting. Both of TBL systems outperform the NLPX system we produced for INEX 2004; however, they are outperformed by the NEXI system. The fact that the 'perfect' TBL outperforms the 'automatic' TBL shows that TBL is an imperfect technology. However, with more training it is possible that the gap between these two systems could close. Particularly encouraging was the fact that all of the systems performed well in comparison with the other INEX participants. This shows that structured NQLs are a viable alternative to formal query languages.

10. Conclusion

Here, we presented NLPX, a natural language interface to an XML-IR system. We also incorporated the transformation-based error-driven learning paradigm to structured natural language queries. We applied TBL to two areas: part of speech tagging and shallow parsing. The results are encouraging, and show that TBL is worthwhile incorporating into structured NLQs. Furthermore, our results indicate that structured

NLQs are a viable alternative to XPath-like formal language queries.

11. References

[1] S. Abney, "Parsing by Chunks", In *Principle-Based Parsing*, Kluwer Academic Publisher, 1991.

[2] S. Aragamon, I. Dagan, Y. Krymolowski, "A Memory Based Approach to Learning Shallow Natural Language Patterns", In *Proceedings of the Joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computation Linguistics*, COLING-ACL, Montreal, Canada, pp. 67-73, 1998.

[3] D. Bourigault, "Surface Grammatical Analysis for the Extraction of Terminological Noun Phrases", In *Proceedings of the Fifteenth International Conference on Computational Linguistics*, pp. 977-981, 1992.

[4] E. Brill, "A Simple Rule-Based Part of Speech Tagger", In *Proceedings of the Third Conference on Applied Computational Linguistics (ACL),* Trento, Italy, 1992.

[5] E. Brill, "Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach", In. *Proceedings of the 31st Meeting of the Association of Computational Linguistics,* 1993.

[6] E. Brill, "A Corpus-Based Approach to Language Learning", PhD Dissertation, Department of Computer and Information Science, University of Pennsylvania, 1993.

[7] E. Brill, "Some Advances in Transformation-Based Part of Speech Tagging", In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94),* Seattle, Washington, United States, 1994.

[8] E. Charniak, C. Henrickson, N. Jacobson, M. Perkowitz, "Equations for Part of Speech Tagging", In. *Proceedings of the Conference for the American Association for Artificial Intelligence*, Washington D.C., United States, July 1993.

[9] K. Church, "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text", In *Second Conference on Applied Natural Language Processing*, ACL, pp.136-143, 1988.

[10] J. Clark and S. DeRose, "XML Path Language XPath Version 1.0", W3C Recommendation, The World Wide Web Consortium, November 1999 available at http://www.w3.org/TR/xpath.

[11] D. Cutting, J. Kupiec, J. Pedersen, P. Silbun, "A Practical Part-Of-Speech Tagger", In *Proceedings of the Third Conference on Applied Natural Language Processing, ACL*, Trento, Italy, 1992.

[12] W. Daelemans, S. Buchholz, J. Vennstra, "Memory Based Shallow Parsing", In Proceedings of EMNLP/VLC-99, Maryland, United States, pp. 239-246, 1999.

[13] E. I. Ejerhed, "Finding Clauses in Unrestricted Text by Finitary and Stochastic Methods", In *Second Conference on Applied Natural Language Processing*, ACL, pp. 219-227.

[14] N. Fuhr and S. Malik. "Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003", In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 1-11. 2004.

[15] B. Green and G. Rubin, *Automatic Grammatical Lagging of English*, Department of Linguistics, Brown University, 1971.

[16] Z. Harris, *String Analysis of Language Structure*, Mouton and Co., The Hague, Holland, 1962.

[17] F. Jerlink, "Markov Space Modelling of Text Generation", In *Impact of Processing Techniques on Communication*, Dordrecht, 1985.

[18] S. Klien and R. F. Simmons, "A Computational Approach to Grammatical Coding of English Words", Journal of the ACM (*JACM)* 10, pp. 334-347, 1963.

[19] T. Kudo, and Y. Matsumoto, "Chunking with Support Vector Machines", In Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL) 2001, Pitsberg, United States, 2001.

[20] J. Kupiec, "Robust Part of Speech Tagging Using a Hidden Markov Model", In *Computer Speech and Language*, 1992.

[21] T. Lager, "A Logic Programming Approach to Word Expert Engineering". In *Proceedings of the First International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications (ACIDCA 2000): Workshop on Corpora and Natural Language Processing*, Monastir, Tunisia, March 22-24 2000.

[22] M. Marcus, N. Santorini, and M. Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank", In Computational Linguistics, 1993

[23] B. Merialdo, "Tagging Text with a Probabilistic Model", In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1991.

[24] R. O'Keefe, and A. Trotman, "The Simplest Query Language That Could Possibly Work", In *INEX 2003 Workshop Proceedings*, Dagstuhl, Germany, December 15-17 2003, pp. 167-174.

[25] F. Pla, A. Molina, N. Preito, "Tagging and Chunking with Bigrams", In. *Proceedings of the 17th conference on Computational linguistics*, Saarbrukecken, Germany, pp. 614-620, 2000.

[26] L. Ramshaw and M. Marcus, "Text Chunking Using Transformation-Based Learning", *Proceedings of the Third Workshop on Very Large Corpora*, pp 82-94, 1995.

[27] A. Trotman and B. Sigurbjörnsson, "NEXI: Now and Next", In *Advances in XML Information Retrieval. Third Workshop of the INitiative for the Evaluation of XML Retrieval INEX 2004*, LNCS 3493, Schloss Dagstuhl, Germany, 6-8 December 2004, to appear in 2005

[28] A. Voutilainen, "NPTool, A Detector of English Noun Phrases", In. Proceedings of the Workshop on Very Large Corpa", ACL, June, pp. 48-57, 1993.

[29] A. Woodley and, S. Geva, "NLPX– An XML-IR System with a Natural Language Interface" , In *Proceedings of the Australasian Document Computing Symposium*, Melbourne, Australia, December 13 2004, pp. 71-74.

[30] A. Woodley and S. Geva, "NLPX at INEX 2004", In *Advances in XML Information Retrieval. Third Workshop of the INitiative for the Evaluation of XML Retrieval INEX 2004*, LNCS 3493, Schloss Dagstuhl, Germany, 6-8 December 2004, to appear in 2005, pp. 393-406.