Hindi Text Normalization

K. Panchapagesan*, Partha Pratim Talukdar, N. Sridhar Krishna, Kalika Bali, A. G. Ramakrishnan†

Hewlett-Packard Labs India
24 Salarpuria Arena, Hosur Road, Bangalore, India
Email: {*partha.talukdar, nsridhar, kalika*}*@hp.com*

† Indian Institute of Science
Bangalore, India
Email: *ramkiag@ee.iisc.ernet.in*

*Birla Institute of Technology & Science
Pilani, Rajasthan, India
Email: *panchapagesan.k@gmail.com*

Abstract

All areas of language and speech technology, directly or indirectly, require handling of real (unrestricted) text. For example, Text-to-Speech systems directly need to work on real text, whereas Automatic Speech Recognition systems depend on language models that are trained on text. This paper reports our ongoing effort on Hindi Text Normalization. In that, a novel approach to text normalization, wherein tokenization and initial token classification are combined into one stage followed by a second level of token sense disambiguation, is described. Tokenization and initial token classification are performed using a lexical analyser that is derived from various token definitions in the form of regular expressions. For second level of token sense disambiguation, application of decision lists and decision trees are explored. Token-to-word rules are then applied, which are specific for each token type and also for each format within a token type.

1 Introduction

All areas of language and speech technology, directly or indirectly, requires handling of real (unrestricted) text [Sproat *et al.*, 2001]. For example, Text-to-Speech (TTS) systems directly need to work on real text, whereas Automatic Speech Recognition (ASR) systems depend on language models that are trained on text. In real text, many non-standard representation of words appear, for e.g., numbers (year, time, ordinal, cardinal, floating point), abbreviations,

acronyms, currency, dates, URLs. All these non-standard representations must typically be normalized, or in other words converted to standard words, which would then be processed in various applications.

Text normalization typically involves tokenization of input text into tokens, identification of Non Standard Words (NSWs) and their categories, and expansion of the NSWs into standard word representations. Tokenization is typically done based on white spaces [Sproat *et al.*, 2001]. Once tokenization is done, each token has to be identified for its corresponding category. Identification of token category involves a high degree of ambiguity, for example, '1974' could refer to *nineteen seventy four* as a year, or *one thousand nine hundred (and) seventy four* as a cardinal number. Disambiguation is generally handled by hand-crafted context-dependent rules. However, such hand-crafted rules are very difficult to write, maintain, and adapt to new domains. Yarowsky [Yarowsky, 1996] demonstrated encouraging results on homograph disambiguation in text normalization using *decision-list* based data-driven techniques. After identifying the category, expansion of NSWs is accomplished by a combination of rules (e.g., for expanding numbers, currency, dates) and look-up tables (e.g., for abbreviations, acronyms).

In principle, any system that deals with unrestricted text need the text to be normalized. Most work on text normalization has been done in the context of two particular applications, namely, Text-to-Speech systems, and *text conditioners* for Automatic Speech Recognition applications. Work on text normalization in TTS systems mostly involves a set of hand-constructed ad-hoc rules that are tuned to a particular application domain [Allen *et al.*, 1987]. In the ASR community, "Text Conditioning Tools" [LDC, 1998] from Linguistic Data Consortium (LDC) for English Language are widely used for the text normalization task. The Johns Hopkins University Summer Workshop research project [Sproat *et al.*, 2001] made a systematic effort to build a general solution to the text normalization problem for English. In India, the emergence of language and speech technology areas are crucial to make computers accessible to a far broader spectrum of the population. There is a greater need for work on text normalization, as it forms an important component of all areas of language and speech technology. However, for Indian languages, to our knowledge, there has not been any concerted effort on text normalization.

This paper describes our ongoing effort on Hindi Text Normalization. The emphasis is also on creating a generic framework (set of tools) for text normalization, so that the tools can easily be used/extended to new domains and new languages. In this paper, an effort is made to identify the various non-standard representations of the words in real (unrestricted) Hindi text and ways of converting them to standard words. Hindi text offers many peculiarities, which need to be taken into account during text normalization. Some of the peculiarities include: 1. Text might contain interspersed English words using Roman script, making it a bilingual text. 2. Numbers could be represented by Arabic numerals at one place, and the native Devanagari numerals at another place, in the same text. 3. Special attention to foriegn language tokens, when the tokens are written partly or fully in Hindi (e.g., AK-47, MIG-29).

We propose a novel approach to text normalization, wherein tokenization and initial token classification are combined into one stage, followed by a second level of token sense disambiguation. The architecture of the proposed approach is shown in Figure 1. Tokenization and initial token classification are performed using a lexical analyser that is derived from various token definitions in the form of regular expressions. For the second level of token sense disambiguation,

application of decision lists and decision trees are explored. Token-to-word rules are then applied, which are specific for each token type and also for each format within a token type.

The organization of the paper is as follows: In Section 2, a detailed description of tokenization and initial token classification is given. Section 3 deals with token sense disambiguation as a general case of homograph disambiguation, and in that the application of decision lists and decision trees are explored. In Section 4, conversion of non standard words to standard word representation is described. Section 5 concludes the paper, after indicating the issues still to be resolved.

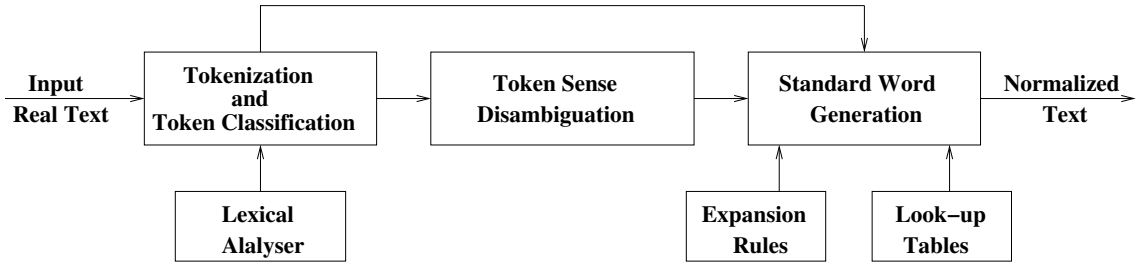

Figure 1: Stages involved in Text Normalization.

2 Tokenization and Initial Classification

Whitespace is the most commonly used delimiter between words and is extensively employed for tokenization. But using whitespace as the only delimiter has a number of shortcomings: a token type which allows the occurrence of whitespace within the token will not be recognized as a single token, but split up into two or more tokens. For example, consider a telephone number, +91 044 1234 5678. This should be identified as a single token of type 'Telephone Number', but if tokenization is exclusively based on whitespace, then we end up having 4 tokens. Further, an important shortcoming is that every token so obtained will then have to go through a token identification process that identifies its token type/category. This approach might not even be feasible for some languages. For example, Chinese and Japanese do not use any form of white space between words.

In our approach to text Normalization, tokenization and initial token classification/identification are achieved in a single step. We have used Flex, an automatic generator tool for high-performance scanners [Mason, 1990], which is primarily used by compiler writers to develop scanners that break up a character stream (program source code) into a sequence of tokens in the front-end of a compiler. Flex takes a set of regular expressions as input and generates a scanner as output that will scan an input stream for the tokens represented by the regular expression. A scanner works as a lexical analyser, recognises lexical patterns in the input text, and thereby groups input characters into tokens. Tokens are specified using patterns (regular expressions).

An effort is made to identify various non-standard representations of the words in real (unrestricted) Hindi text. Various formats of each Non Standard Word (NSW) category are defined

through regular expressions. English language characters and Arabic numerals are also processed as they appear frequently in real Hindi text. The following non-standard representations are addressed:

- Cardinal numbers and Literal Strings
- Ordinal numbers
- Roman Numerals
- Fractions
- Ratios
- Decimal Numbers
- Time
- Telephone Numbers
- Date
- Year
- URL and E-mail (not complete)
- Range
- Percentage
- Alphanumeric strings

Initially the input text is chunked into sentences based on the Hindi sentence delimiter '*PurN Viram*'. When the generated lexical analyser is run on each sentence, it analyses the text looking for strings which match one of its patterns. If it finds more than one match, it selects the one that matches the largest chunk of text. If it finds two or more matches of the same length, the first matching rule is chosen. So, by defining regular expressions that match the formats of the various token types, we can automatically extract the token that best fits the given token description. In case of ambiguity between two or more token categories/types for a particular token, the lexical analyser has been configured to output the possible categories with the token to facilitate token sense disambiguation at a later stage. Using this approach, with a single pass on the input text, we can complete tokenization and initial token classification (most of the NSW category identification). An example output from the lexical analyser (scanner) is shown in Figure 2.

The gains of the approach described above can be summarised as follows: the approach could be extended to new domains and new languages, by defining suitable patterns (regular expressions) for each NSW category. As the tokenization and initial token classification is done in one stage itself, the overhead involved in token (NSW category) identification gets reduced. The information obtained at this stage (the complete meaningful token name) can be used effectively in the prosody modules of a TTS system thereby helping in appropriate rendering of the speech. For example, as the telephone number +91 080 1234 5678 is tokenized and identified as a single meaningful token (NSW) type "Telephone Number" (instead of tokenizing the telephone number into 4 tokens and a further token identification process for each of the tokens), the complete meaningful token name can be used effectively in the linguistic analysis modules of a TTS system.

INPUT TO LEXICAL ANALYSER:

मंगल प्रसाद लोधा ने बताया कि शुक्रवार दोपहर से ई-मेल पता ataljigetwellsoon@yahoo.com पर मेल भेजे जा सकते हैं। (`Mangal prasad lodha said that e-mails can be sent at the e-mail address ataljigetwellsoon@yahoo.com from Friday afternoon.').

अमरीकी राष्ट्रपतियों का गौरवशाली निवास स्थल व्हाइट हाऊस एक नवम्बर को अपनी २००वीं वर्षगांठ मनायेगा । (`The prestigious residence of American presidents, the whitehouse, is celebrating its 200th anniversary on 1st November.').

२१ अगस्त १९९० को गवली के दगडी चाल (भायखला) स्थित निवास से पुलिस को एक एके-५६ असाल्ट राइफल मिली थी। (`On 21st August 1990, the police found an AK-46 assault rifle, at a residence in dagrichawl (bhairwala) in Gawali.').

OUTPUT OF LEXICAL ANALYSER:

मंगल प्रसाद लोधा ने बताया कि शुक्रवार दोपहर से ई-मेल पता
<EMAIL> ataljigetwellsoon@yahoo.com </EMAIL> पर मेल भेजे जा सकते हैं ।

अमरीकी राष्ट्रपतियों का गौरवशाली निवास स्थल व्हाइट हाऊस एक नवम्बर को अपनी
<NORD> २००वीं </NORD> वर्ष गांठ मनायेग

<NDATE> २१ </NDATE> अगस्त <NYEAR> १९९० </NYEAR> को गवली के दगडी चाल (भायखला) स्थित निवास से पुलिस को एक <ALPHANUM> एके-५६ </ALPHANUM> असाल्ट राइफल मिली थी।

Figure 2: Example input/output from lexical analyser

3 Token Sense Disambiguation

Once the tokens are extracted from the input text, the category of each token needs to be identified. This is accomplished by the lexical analyser itself when there is no ambiguity arising from the format(s) of the token, as explained in Section 2. In case of ambiguity, the token with the possible token types/categories is output to facilitate further disambiguation. Identification of token category involves high degree of ambiguity. For example, '1824' could be of the type 'Year', or of the type 'Cardinal Number', and '1.25' could be of the type 'Float', or of the type 'Time'.

Disambiguation is generally handled by hand-crafted context-dependent rules. However, such hand-crafted rules are very difficult to write, maintain, and adapt to new domains. Token sense disambiguation can be mapped to a general homograph disambiguation problem [Yarowsky, 1996]. We have used decision tree based data-driven techniques to address the same. Section 3.1. gives a general description of decision tree and decision list based learning. In Section

3.2., the corpus used for learning is described. Section 3.3. gives an objective evaluation of the decision-list and decision-tree based sense disambiguation models.

3.1 Decision Trees and Decision Lists

Decision trees are models based on self learning procedures that sort the instances in the learning data by binary questions about the attributes that the instances have. It starts at the root node and continues to ask questions about the attribute of the instance down the tree until a leaf node is reached [Mitchell, 1997]. At each node the decision tree algorithm selects both the best attribute and the question to be asked about that attribute. The selection is based on what attribute and question about it divide the learning data so that it gives the best predictive value for the classification. When a token is input to the tree for disambiguation, a decision is made by traversing the tree starting from the root node, taking various paths satisfying the conditions at intermediate nodes, till the leaf node is reached. The path taken depends on various contextual features defined for the token. The leaf node contains the predictive value for the decision.

Decision lists are a special class of decision trees. Decision lists maybe the simplest model for hierarchal decision making. Despite their simplicity, they can be used for representing a wide range of classifiers. A decision list can be viewed as a hierarchy of rules. When a classification is needed, the first rule in the hierarchy is addressed. If this rule suggests a classification, then its decision is taken to be the classification of the decision list. Otherwise, the second rule in the hierarchy is addressed. If that rule fails to classify as well, the third rule is addressed, and so on. Often, programmers prefer presenting decision lists as sequences of if-then-else statements, intended for classifying an instance x. For example:

```
if condition1(x) is true then output = output1(x)
else if condition2(x) is true then output = output2(x)
else if condition3(x) is true then output = output3(x)
. . .
. . .
. . .
else output = default_output(x)
```

3.2 Tagged Corpus

We have identified two very common type of homographs in our analysis on real Hindi Text: 1. A four digit number could be of the type ‘Year’, or of type ‘Cardinal-Number’. 2. A number with the format (<[Digit]>[Digit]“.”<[Digit]>[Digit]) could be of the type ‘Float’, or of type ‘Time’. Part of the Emille corpus [Emille, 2003] is taken for study, and the two token formats are tagged manually with the correct classification (token category). Distribution of the two groups, Cardinal-Number/Year and Float/Time, in the tagged corpus are given in Table 1 and Table 2, respectively.

Group	Cardinal-Number	Year	Total Records
Cardinal-Number/Year	7612	4602	12214

Table 1: Distribution of Cardinal-Number and Year in the tagged corpus

Group	Time	Float	Total Records
Float/Time	1873	3616	5489

Table 2: Distribution of Float and Time in the tagged corpus

3.3 Learning

Each token in the corpus is annotated with a context window together with the actual token category. To assess the usefulness and contribution of context, analysis is done with context window sizes of 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15 and 20. By context window size, we mean the number of tokens to the left and the right side of the token under consideration. The feature data is divided randomly into training data (80% of the total feature data) and test data (rest of 20% of the total feature data) for each of the two groups, Cardinal-Number/Year and Float/Time. For decision list and decision tree building process, 'Wagon' classification and regression tree tool [SpeechTools , 2002] is used.

3.3.1 Results

For objective evaluation, both decision list and decision tree models for each of the two groups, Cardinal-Number/Year and Float/Time, are trained with the training data and evaluated on the test data. The prediction accuracy for the Cardinal-Number/Year group, obtained using decision lists and decision trees, is shown in Figure 3 and Figure 4 respectively, as a function of the size of the context window. It is observed that decision trees perform slightly better than decision lists, giving a prediction accuracy of 97.8% with a CWS of 3. The best performance using the decision list learning is 96.1% with a CWS of 2.

The prediction accuracy for the Float/Time group, obtained using decision lists and decision trees, is shown in Figure 5 and Figure 6 respectively, as a function of the size of the context window. It is observed that decision lists perform slightly better than decision trees, giving a prediction accuracy of 99.2% with a Context Window Size (CWS) of 3. The best performance using the decision tree learning is 98.6% with a CWS of 1.

It is observed that, in general, Float/Time disambiguation is very accurate and also the disambiguation requires very little context (CWS of 1). The Cardinal-Number/Year disambiguation is quite good but relatively less accurate compared to the Float/Time disambiguation. Another interesting observation is that the Cardinal-Number/Year disambiguation requires more context (CWS of 3) for good prediction accuracy.

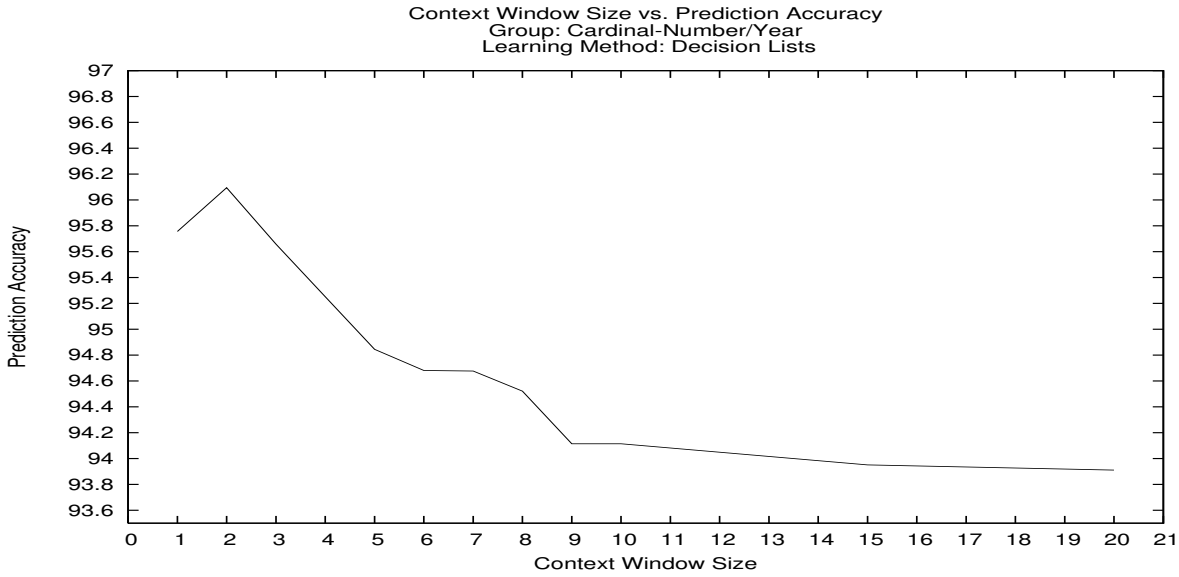

Figure 3: Prediction accuracy of the Cardinal-Number/Year disambiguation, using Decision Lists, as a function of context window size

4 Token-to-Words

Having identified, and disambiguated the token categories when required, expansion of NSWs is accomplished by a combination of rules (e.g., for expanding numbers, currency, dates) and look-up tables (e.g., for abbreviations, acronyms). Token-to-Word rules are written, which are specific for each token type, and for each format within a token type. The token types handled presently are Numbers (cardinal and ordinal), Telephone Numbers, Time, Date, Year, and Literal Strings.

5 Conclusions and Future Work

A general framework for Hindi text normalization has been presented. A novel approach to text normalization, wherein tokenization and initial token classification are combined into one stage followed by a second level of token sense disambiguation, is described. Tokenization and initial token classification are performed using a lexical analyser that is derived from various token definitions in the form of regular expressions. For second level of token sense disambiguation, application of decision lists and decision trees are explored. Token-to-word rules are then applied, which are specific for each token type and also for each format within a token type.

The work reported in this paper is only a preliminary attempt towards a more generalised solution to Hindi Text Normalization. There are many issues to be resolved and addressed in the token types definition, in the token sense disambiguation, and in the expansion of NSWs to standard words. This paper reports our ongoing effort for the Hindi Text Normalization problem and we believe there is still a long a way to go for a complete or near perfect solution.

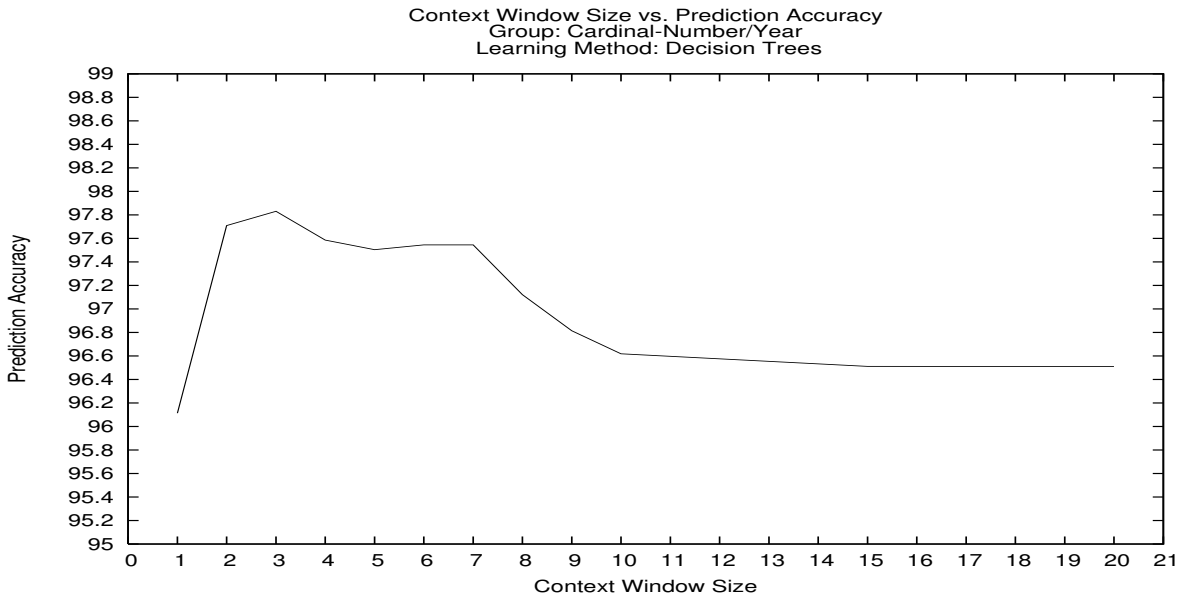

Figure 4: Prediction accuracy of the Cardinal-Number/Year disambiguation, using Decision Trees, as a function of context window size

References

[Sproat *et al.*, 2001] Richard Sproat, Alan Black, Stanley Chen, Shankar Kumar, Mari Ostendorf and Christopher Richards. Normalization of Non-Standard Words. *Computer Speech and Language*, 15(3):287–333, 2001.

[Allen *et al.*, 1987] Jonathan Allen, M. Sharon Hunnicutt, and Dennis Klatt. *From Text to Speech: The MITalk System* Cambridge University Press, Cambridge, 1987.

[LDC, 1998] Linguistic Data Consortium: Text Conditioning Tools http://morph.ldc.upenn.edu/Catalog/LDC98T31.html, 1998

[Yarowsky, 1996] Yarowsky, David. Homograph Disambiguation in Text-to-Speech Synthesis In Jan van Santen, Richard Sproat, Joseph Olive, and Julia Hirschberg, editors, *Progress in Speech Synthesis*, Springer, New York, pages 157–172, 1996.

[Mason, 1990] T. Mason and D. Brown. Lex & Yacc. In *O'Reilly and Associates, Inc.* , 632 Petaluma Ave, CA, 1990.

[SpeechTools , 2002] Taylor, P., R. Caley, and A.W. Black. The Edinburgh Speech Tools Library http://www.cstr.ed.ac.uk/projects/speechtools.html, 2002

[Emille, 2003] Emille corpus, http://www.emille.lancs.ac.uk, 2003.

[Mitchell, 1997] Mitchell, T.M. *Machine Learning* McGraw-Hill, New York, 1997

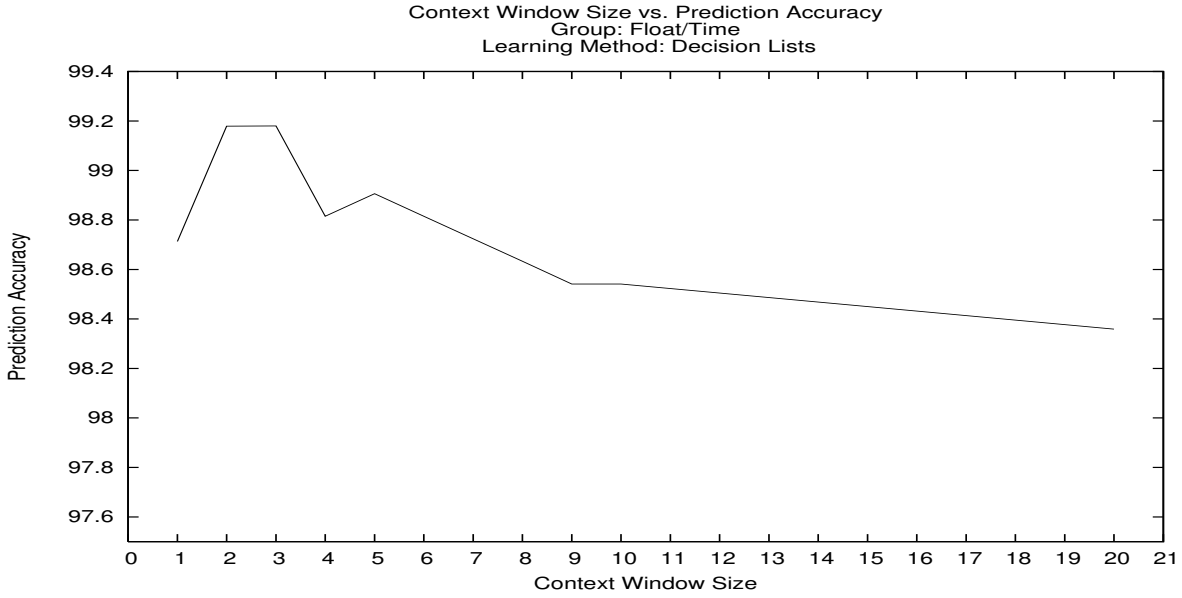

Figure 5: Prediction accuracy of the Float/Time disambiguation, using Decision Lists, as a function of context window size

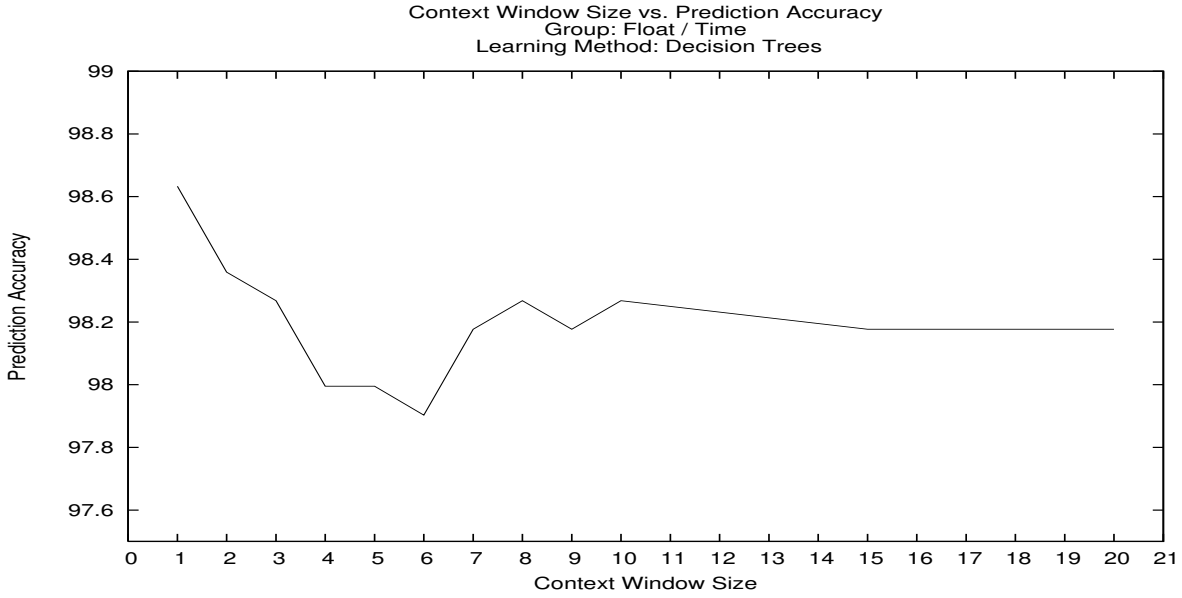

Figure 6: Prediction accuracy of the Float/Time disambiguation, using Decision Trees, as a function of context window size