Computational Linguistics Group
Department of Computer Science
University of Haifa

בלשנות חישובית
החוג למדעי המחשב
אוניברסיטת חיפה

Laboratory in Natural Language Processing (203.4650)

Shuly Wintner

Semester B, 2005-6: Wednesday, 9:00–12:00

`http://cs.haifa.ac.il/~ shuly/teaching/06/lab/`

1 Objectives

The Lab offers a number of practical projects in Natural Language Processing, focusing on (but not limited to) processing of Hebrew. Some projects require previous knowledge of computational linguistics but some assume no previous background. All projects involve programming: the end result is a relatively large-scale, well-documented and efficient software package. Some of the projects may involve also some research (e.g., reading a research paper and implementing its ideas).

2 List of projects

2.1 A compiler from XFST to LexTools

Introduction to Computational Linguistics is required

Finite-state technology is widely considered to be the appropriate means for describing the phonological and morphological phenomena of natural languages. Several finite-state ”toolboxes” exist which facilitate the stipulation of phonological and morphological rules by extending the language of regular expressions with additional operators. Such toolboxes typically include a language for extended regular expressions and a compiler from regular expressions to finite-state devices (automata and transducers). Unfortunately, there are no standards for the syntax of extended regular expression languages.

In this project you will design and implement a compiler which translates grammars expressed in XFST (Beesley and Karttunen, 2003) to grammars LexTools, a simple language built on top of the FSM finite-state toolbox (Mohri, Pereira, and Riley, 2000). You will be able to use a front-end compiler of XFST (Cohen-Sygal and Wintner, 2005), but the back-end, generating the LexTools code, will have to be implemented from scratch.

The contribution of such a project lies in the fact that the Xerox utilities are proprietary; compilation to LexTools will enable us to use grammars developed with XFST on publicly available systems. Furthermore, parallel investigation of two similar, yet different, systems, is likely to result in new insights regarding the two systems and there interrelationships. Finally, such a compiler will enable us to compare the performance of the two systems on very similar benchmarks.

2.2 Implementation of registered FSAs

Introduction to Computational Linguistics is required. Due to the Unix-only availability of FSM, this project must be implemented in a Unix environment.

Finite-state registered automata (FSRA, Cohen-Sygal and Wintner (2006)) extend standard finite-state automata by adding very limited memory, in the form of a finite number of finitely-valued *registers*, to networks. Provably equivalent to finite-state automata, FSRA have been shown to be useful for naturally implementing several non-concatenative phenomena which are observed in natural languages.

In this project you will implement a package which supports FSRA. This will consist in two main phases:

- Extending the regular expression language of XFST by adding dedicated operators for FSRA. You will be able to use a front-end compiler of XFST (Cohen-Sygal and Wintner, 2005), and will have to extend it to support also the operators introduced by Cohen-Sygal and Wintner (2006).

- Compiling extended regular expressions to FSM. Extending and modifying the back-end of the XFST compiler (Cohen-Sygal and Wintner, 2005), you will support register operations by compiling extended regular expressions directly to FSM (Mohri, Pereira, and Riley, 2000), a finite-state low-level toolbox.

2.3 A user interface for KWIC in Hebrew

No prior knowledge is required. Understanding of SQL databases is recommended.

Key Word In Context (KWIC) is an algorithm which, given a text and a keyword, presents all the occurrences of the word in the text, allowing a few context words on both sides of the keyword to be displayed. Such a tool is very useful for linguistic research.

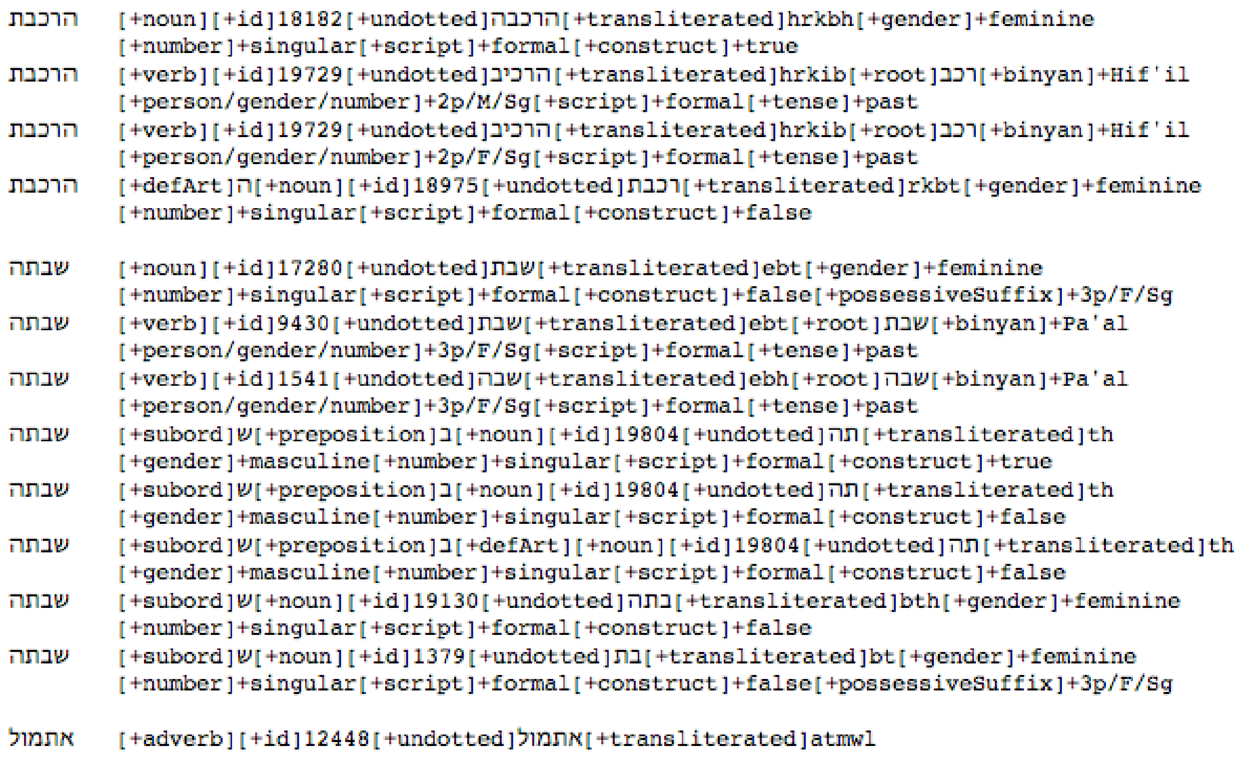
You will develop a KWIC system with a graphical user interface which will allow users to present queries referring not just to words, but also to their morphological features. This tool will be similar to an existing GUI for Arabic (Dror et al., 2004), but will be specific to Hebrew corpora. The underlying corpora will be XML documents of morphologically analyzed Hebrew texts. The GUI will enable users to specify a corpus to work with, and then search the corpus for combinations of words and/or their properties. To this end, once a corpus is loaded it will have to be stored in an efficient database, probably HSQLDB. The GUI should be accessible on the Web, and hence will have to be developed in a Web-supporting environment, e.g., JSP or PHP.

A detailed requirements specification will be available in a separate document..

2.4 Morphological analysis of dotted Hebrew

Introduction to Computational Linguistics recommended but not required. As you will be revising an existing Java code, knowledge of Java is mandatory.

Morphological analysis is the process of determining the base (also known as *lexeme*, or *lemma*) of a word, along with its morphological attributes. An example of the morphological analysis of a simple Hebrew sentence is depicted in Figure 1.



```
הרכבת    [+noun][+id]18182[+undotted]הרכבה[+transliterated]hrkbh[+gender]+feminine
         [+number]+singular[+script]+formal[+construct]+true
הרכבת    [+verb][+id]19729[+undotted]הרכיב[+transliterated]hrkib[+root]רכב[+binyan]+Hif'il
         [+person/gender/number]+2p/M/Sg[+script]+formal[+tense]+past
הרכבת    [+verb][+id]19729[+undotted]הרכיב[+transliterated]hrkib[+root]רכב[+binyan]+Hif'il
         [+person/gender/number]+2p/F/Sg[+script]+formal[+tense]+past
הרכבת    [+defArt]ה[+noun][+id]18975[+undotted]רכבת[+transliterated]rkbt[+gender]+feminine
         [+number]+singular[+script]+formal[+construct]+false

שבתה     [+noun][+id]17280[+undotted]שבת[+transliterated]ebt[+gender]+feminine
         [+number]+singular[+script]+formal[+construct]+false[+possessiveSuffix]+3p/F/Sg
שבתה     [+verb][+id]9430[+undotted]שבת[+transliterated]ebt[+root]שבת[+binyan]+Pa'al
         [+person/gender/number]+3p/F/Sg[+script]+formal[+tense]+past
שבתה     [+verb][+id]1541[+undotted]שבה[+transliterated]ebh[+root]שבה[+binyan]+Pa'al
         [+person/gender/number]+3p/F/Sg[+script]+formal[+tense]+past
שבתה     [+subord]ש[+preposition]ב[+noun][+id]19804[+undotted]תה[+transliterated]th
         [+gender]+masculine[+number]+singular[+script]+formal[+construct]+true
שבתה     [+subord]ש[+preposition]ב[+noun][+id]19804[+undotted]תה[+transliterated]th
         [+gender]+masculine[+number]+singular[+script]+formal[+construct]+false
שבתה     [+subord]ש[+preposition]ב[+defArt][+noun][+id]19804[+undotted]תה[+transliterated]th
         [+gender]+masculine[+number]+singular[+script]+formal[+construct]+false
שבתה     [+subord]ש[+noun][+id]19130[+undotted]בתה[+transliterated]bth[+gender]+feminine
         [+number]+singular[+script]+formal[+construct]+false
שבתה     [+subord]ש[+noun][+id]1379[+undotted]בת[+transliterated]bt[+gender]+feminine
         [+number]+singular[+script]+formal[+construct]+false[+possessiveSuffix]+3p/F/Sg

אתמול    [+adverb][+id]12448[+undotted]אתמול[+transliterated]atmwl
```

Figure 1: Example morphological analysis

Hebrew has a complex morphology and hence the design of a morphological analyzer for the language is a complex task. We currently have a large-sclae and relatively accurate morphological system for Hebrew (Yona and Wintner, 2005) which works for *undotted* texts. In this project you will create a variant of the morphological system for the *dotted* script.

The main task here is to understand the morphological rules that apply to works, as stipulated for the undotted case, and then revise and refine them for the dotted case. The greatest benefit of such a system is that it will facilitate, in conjunction with a morphological disambiguation system which is currently under development, an automatic vocalization of undotted texts.

2.5 Converting dotted to undotted Hebrew

Introduction to Computational Linguistics is required

The Hebrew script has two main standards: dotted (vocalized) and undotted. In this project you will develop a program which converts the dotted words to their undotted counterparts. Note that this does not simply imply removing the dots, as many times letters such as ו or י are inserted to replace the missing dots. The rules are available from The Academy of the Hebrew Language (`http://hebrew-academy.huji.ac.il/decision4.html`).

The conversion will be done using finite-state technology, with a common toolbox (such as the Xerox XFST package). This will facilitate also the reverse conversion, from undotted to dotted script.

2.6 A grammar for Hebrew numeric and date expressions

Introduction to Computational Linguistics is required

Numeric expressions (such as *nineteen hundred sixty three* or *three quarters*) and date expressions (such as *last weekend* or *the third quarter of 2004*) are abundant in natural language texts and their recognition is both important and relatively easy. Correctly identifying such expressions in texts can greatly reduce the complexity of further processing, such as parsing, and contribute to the computation of the text meaning.

In this project you will design and implement a grammar for such expressions in Hebrew. The result should be a program whose input is a Hebrew text, morphologically analyzed, and whose output is the same text, where numeric and date expressions are properly annotated. The input and the output will be represented in XML. The grammar will be developed using finite-state technology, with a common toolbox (such as the Xerox XFST package).

2.7 Collecting and aligning a bilingual corpus

No background in NLP is required

Text corpora are among the most important resource for a variety of NLP applications. They are used to provide word frequency counts for statistical NLP and information retrieval applications such as part-of-speech taggers, shallow parsers, categorization and summarization, to list just a few. Collecting corpora, representing and maintaining them are non-trivial tasks. The objective of this project is to build a parallel corpus of Hebrew and English documents by crawling the web. The documents in the corpus will then be sentence- and word-aligned.

You will develop software for collecting Hebrew-English corpora. The main technique is web-crawling: a program which crawls the web and searches for relevant documents. The main task is determining whether two documents are indeed possible translations, and you will be able to use some of the techniques reported in the literature (Resnik and Smith, 2003). Search will be limited to a number of dynamic web sites which are known to have similar documents in the two languages (e.g., some newspapers).

Once a parallel corpus is available, you will use an existing tool (GIZA++, Och and Ney (2000)) to implement sentence- and word-level alignment of the texts in the corpus.

2.8 Named entity recognition in Hebrew

Statistical NLP is recommended but not strictly required

The named entity task is to identify all named locations, named persons, named organizations, dates, times, monetary amounts, and percentages in text. Though this sounds clear, enough special cases arise to require lengthy guidelines, e.g., when is *The Wall Street Journal* an artifact, and when is it an organization? When is *White House* an organization, and when a location? Is a street name a location? Should *yesterday* and *last Tuesday* be labeled dates? In order to achieve human annotator consistency, guidelines with numerous special cases have been defined for the Seventh Message Understanding Conference, MUC-7 (Chinchor, 1997). These guidelines were adapted for Hebrew and we currently have a relatively large Hebrew corpus annotated with three types of named entities.

In this project you will develop a named entity recognizer for Hebrew using machine learning techniques. The main challenge of this task is to represent the problem in a way that will enable a general-purpose classification algorithm (in this case, SNoW (Roth, 1998)) to make the correct predictions. You will have to design the features with which instances of the problem are represented, train the classifier and then carefully evaluate the results.

2.9 Named entity transliteration in Hebrew

Statistical NLP is recommended but not strictly required

Transliteration is the process of replacing words and phrases in one language with their approximate spelling or phonetic equivalents in some other language. We distinguish between two types of transliteration:

Forward transliteration: When a Hebrew name is transliterated into English. For example, אריאל שרון is transliterated to *Ariel Sharon* and חיפה, ישראל to *Haifa, Israel.*

Backward transliteration: This is the reverse transliteration process where an English term which was transliterated to Hebrew has to be recovered. For example, ביל קלינטון to *Bill Clinton,* הוליווד to *Hollywood.*

When translating text from one language to another, proper names are sometimes translated, sometimes transliterated and sometimes a mixed approach is used. For example, when translating from Hebrew to English, names of people are always transliterated: אריאל שרון is transliterated to *Ariel Sharon* and ביל קלינטון to *Bill Clinton*. Other proper names, especially of organizations, are translated: הבית הלבן to *The White House*, האומות המאוחדות to *The United Nations*. Sometimes, however, proper names are partly translated and partly transliterated, as in הר חרמון *Mount Hermon* or מפרץ חיפה *Haifa bay*.

In this project you will develop a classifier of named entities which will determine, given a text, whether words should be translated or transliterated, using machine learning techniques. The main challenge of this task is to represent the problem in a way that will enable a general-purpose classification algorithm (in this case, SNoW (Roth, 1998)) to make the correct predictions. You will have to design the features with which instances of the problem are represented, train the classifier and then carefully evaluate the results. You will also have to annotate a large corpus for training and testing, using an existing annotation tool.

3 Administration

Projects are to be implemented by groups of at most two students. All work must be completed by the end of the semester. All systems will be presented at the end of the semester for a final demo. The programming language must be portable enough to be usable on a variety of platforms; Java is recommended, C++ or Perl will be tolerated, if you have a different language in mind discuss it with the instructor.

Grading will be based on comprehension of the problem, quality of the implementation and quality of the documentation. In particular, the final grade will be based on:

- Comprehension of the problem (and the accompanying paper, where applicable)
- Full implementation of a working solution
- Presentation of a final working system
- Comprehensive documentation

References

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite-State Morphology: Xerox Tools and Techniques*. CSLI, Stanford.

Chinchor, Nancy. 1997. MUC-7 named entity task definition. Available from `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/`, September.

Cohen-Sygal, Yael and Shuly Wintner. 2005. XFST2FSA: Comparing two finite-state toolboxes. In *Proceedings of the ACL-2005 Workshop on Software*, Ann Arbor, MI, June.

Cohen-Sygal, Yael and Shuly Wintner. 2006. Finite-state registered automata for non-concatenative morphology. *Computational Linguistics*, 32(1), March. Forthcoming.

Dror, Yehudit, Dudu Shaharabani, Rafi Talmon, and Shuly Wintner. 2004. Morphological analysis of the Qur'an. *Literary and linguistic computing*, 19(4):431–452.

Mohri, Mehryar, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32, January.

Och, F. J. and H. Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL-2000*, pages 440–447, Hongkong, China, October.

Resnik, Philip and Noah A. Smith. 2003. The web as a parallel corpus. *Comput. Linguist.*, 29(3):349–380.

Roth, Dan. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of AAAI-98 and IAAI-98*, pages 806–813, Madison, Wisconsin.

Yona, Shlomo and Shuly Wintner. 2005. A finite-state morphological grammar of Hebrew. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 9–16, Ann Arbor, Michigan, June. Association for Computational Linguistics.