Department of Computer Science **החוג למדעי המחשב**
University of Haifa **אוניברסיטת חיפה**

Mount Carmel, 31905 Haifa, Israel Phone: +972-4-8240259 Fax: +972-4-8249331 הר הכרמל, חיפה 31905 טל. 04-8240259 פקס: 04-8249331

Laboratory in Natural Language Processing (203.4650)

Shuly Wintner

Semester A, 2004-5: Wednesday, 16:00–18:00

`http://cs.haifa.ac.il/~shuly/teaching/05/lab/`

1 Objectives

The Lab offers a number of practical projects in Natural Language Processing, mostly geared towards 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 Web crawler for collecting Hebrew and bilingual corpora

No background in NLP is required

Text corpora are the single 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 collect a large number of Hebrew documents by crawling the web. A secondary objective is to create a bilingual corpus of similar texts.

You will develop software for collecting Hebrew as well as Hebrew-English corpora. The main technique is web-crawling: a program which crawls the web and searches for relevant documents. For the Hebrew corpus, a sufficient condition is that the document be in Hebrew. You will develop a subroutine which determines the language of a document using simple cues. Once a document is determined to be in Hebrew, the software will store it and index it. Indexing is required in order to determine whether the same (or a very similar) document has already been collected. Note that while many web sites are relatively static, many change on a daily basis (e.g., newspapers).

Constructing a bilingual corpus is more involved. Search here 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). Using cues such as URLs and resources such as a bilingual dictionary, you will have to determine whether two documents are indeed “similar”, and if they are, collect and store them. 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.

2.2 Consolidation of bilingual dictionaries

No background in NLP is required

A crucial resource for almost any NLP application is a lexicon. For Hebrew, a large-scale, broad-coverage lexicon is currently being build at the Knowledge Center for Hebrew Telecommunication. It is represented in XML according to a well-defined and well-documented schema.

For multilingual applications, bilingual extensions of monolingual lexicons are required. The goal of this project is to extend the Hebrew lexicon with Hebrew-English translation pairs. We currently have three separate bilingual dictionaries, represented in tabular format. You will have to extend the XML schema of the Hebrew lexicon to allow for Hebrew-English translation pairs; to extend an existing tool for editing the lexicon such that it support the newly added fields; and to design a procedure for automatically populating the lexicon with the translation pairs in the existing bilingual dictionaries.

2.3 Hebrew tokenization

No background in NLP is required

The objective of this project is to design and implement a computer program which, when given a text document in Hebrew, where possible noise such as non-Hebrew words, HTML tags, pictures etc. may exist, returns a clean version of the text where each Hebrew “token” is identified. The output will be represented in XML.

The input to the program is a file which contains Hebrew text. This file is likely to be an HTML document with various types of noise. For example, Hebrew HTML pages typically contain HTML tags, non-Hebrew words, pictures, etc. Your goal is to clean these non-Hebrew elements. Then, the program must determine the boundaries of tokens. Usually, a token is defined as a sequence of characters delimited by spaces. However, the definition is language-dependent. For example, Hebrew words may contain punctuation marks (e.g., the double quote " is used for acronyms). Some Hebrew tokens may even include full stops (as in י.ב.מ.). Other punctuation marks are not part of the token, e.g., the full stop at the end of a sentence. More complex examples include strings such as ב־23 באפריל or ל־12 חודשים, vs. בית־ספר.

2.4 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).

The goal of this project is to develop a named entity recognizer for Hebrew, following the algorithm of Bikel, Schwartz, and Weischedel (1999). This task involves preparing a training corpus, i.e., manually annotating a large corpus of text according to the MUC guidelines; a straight-forward implementation of the algorithm; and a careful evaluation of the results.

2.5 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.

The goal of this project is to design and implement a grammar for such expressions in Hebrew. The result should be a program whose input is a Hebrew text, after tokenization, 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).

2.6 Transliteration of Hebrew named entities

No background in NLP is required

Transliteration is the process of replacing words and phrases in one language with their approximate spelling or phonetic equivalents in some other language. The goal of this project is to develop a system for transliterating Hebrew words and phrases into English. 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.*

You will implement the algorithm proposed by Al-Onaizan and Knight (2002). The development will include collection of the required resources, implementation and evaluation.

2.7 Chunking for Hebrew

Introduction to Computational Linguistics is required

Text chunking consists of dividing a text into syntactically correlated parts of words. For example, the sentence *He reckons the current account deficit will narrow to only $1.8 billion in September* can be divided as follows:

> [NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only $1.8 billion] [PP in] [NP September]

Text chunking is an intermediate step towards full parsing and is useful for a variety of applications. The goal of this project is to develop a system for chunking Hebrew text using two techniques: a finite-state cascade of transducers and machine learning. In order to evaluate the performance of either system, you will also develop a procedure which converts the Hebrew TreeBank format (Sima'an et al., 2001) to chunks.

The problem was first introduced by Abney (1991), who later developed a finite-state-technology solution which you will implement in this work (Abney, 1996). Later, it was addressed as a classification problem using machine learning techniques by various authors (Ramshaw and Marcus, 1995; Daelemans, Buchholz, and Veenstra, 1999; Punyakanok and Roth, 2001). You will implement one of the machine learning algorithms (using an existing toolbox) and compare the results to the finite-state case.

2.8 Statistical parsing of Hebrew

Introduction to Computational Linguistics and Statistical NLP are required

The best available parser for English is assumed to be the Collins parser (Collins, 1999). In this project you will experiment with the applicability of statistical parsing in general, and the Collins parser in particular, for a language with fewer resources than English. You will use the recently created Hebrew TreeBank (Sima'an et al., 2001) to train and evaluate an existing statistical parser (Bikel, 2004). The main question is whether the limited training data (approximately 2,000 sentences) are sufficient for reasonable results. Proposals for improvement will be highly regarded.

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 or Perl are recommended, C++ 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

Abney, Steven. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-based parsing*. Kluwer academic publishers.

Abney, Steven. 1996. Partial parsing via finite-state cascades. In *Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information*, pages 8–15, Prague, Czech Republic.

Al-Onaizan, Yaser and Kevin Knight. 2002. Machine transliteration of names in Arabic text. In *Proceedings of the ACL workshop on computational approaches to Semitic languages*.

Bikel, Dan. 2004. Multilingual statistical parsing engine. Available from `http://www.cis.upenn.edu/~dbikel/software.html#stat-parser`.

Bikel, Daniel M., Richard L. Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

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

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Daelemans, Walter, Sabine Buchholz, and Jorn Veenstra. 1999. Memory-based shallow parsing. In *Proceedings of CoNLL*.

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

Punyakanok, Vasin and Dan Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems 13*, pages 995–1001. MIT Press.

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

Sima'an, Khalil, Alon Itai, Yoad Winter, Alon Altman, and N. Nativ. 2001. Building a tree-bank of Modern Hebrew text. *Traitment Automatique des Langues*, 42(2).