Computational Linguistics Group
Department of Computer Science
University of Haifa

בלשנות חישובית
החוג למדעי המחשב
אוניברסיטת חיפה

Laboratory in Natural Language Processing

Shuly Wintner, shuly@cs.haifa.ac.il

Semester B, 2010: Wednesday, 18:00–21:00

1 Objectives

The Lab offers a number of practical projects in Natural Language Processing (NLP), 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 (except one) 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 Administration

Projects are to be implemented by groups of at most two students. All systems will be presented at the end of the semester for a final demo. A coordination meeting is planned for Wednesday, June 2nd; all work must be completed by Tuesday, August 31st. A project presentation meeting will be held on Wednesday, September 1st.

The programming language must be portable enough to be usable on a variety of platforms; Python is recommended, C++, Perl or Java will be tolerated, if you have a different language in mind discuss it with the instructor. Most projects will have to be executed in a Linux environment due to dependencies on external packages.

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(s), where applicable); Full implementation of a working solution; Presentation of a final working system; Comprehensive documentation.

3 List of projects

3.1 Morphological analysis of dotted Hebrew

Introduction to Computational Linguistics recommended but not mandatory. 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-scale and relatively accurate morphological system for Hebrew (Yona and Wintner, 2008; Itai and Wintner, 2008) 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 words, 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.

3.2 Converting dotted to undotted Hebrew

No prior knowledge 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. Ideally, your solution should be reversible, so as to (non-deterministically) generate dotted forms from undotted ones.

3.3 A web-based 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 web-based 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, the corpora will have to be stored in an efficient database; you will be able to use an existing infrastructure for storing corpora, such as The Corpus Workbench. 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.

3.4 A generic transliteration system

Introduction to Computational Linguistics recommended but not mandatory.

When texts are translated from one language to another, some words are not translated; rather, they are *transliterated*: rendered in the writing system of the target language in a way that retains or approximates the original pronunciation of the word. Transliterated words are frequently proper names or loan words. For example, when the Hebrew sentence ספרד הביסה את ברזיל **3 :0** is translated to English, the proper name ספרד is translated to *Spain*, but the proper name ברזיל is transliterated as *Brazil*.

You will develop a generic system for transliterating words in a large number of languages to English, following the methodology of Kirschenbaum and Wintner (2009, 2010). Transliteration will be based on statistical machine translation (Brown et al., 1990), in which the translation model maps characters in the source language to characters in English, and the language model is a unigram English word model (viewed as a character n-gram model). The language model will be provided to you. The translation model will be extracted from multilingual titles of Wikipedia documents.

In order to create a translation model for a given source language, you will have to extract from Wikipedia all the titles of the articles that occur both in the source language and in English, and to determine whether these titles are translations or transliterations. This can be done by comparing the characters in the title terms, given some possible mappings of characters from the source to English. For example, the Hebrew-English mapping will include the pairs ב–*b*, ב–*v*, פ–*p*, פ–*f*, ס–*s*, ר–*r*, ד–*d*, ז–*z*, ל–*l*. Based on such mapping, you will be able to determine that ברזיל–*Brazil* is a transliterated pair, whereas ספרד–*Spain* is not. You will have to prepare such character mapping tables for a few languages.

In order to evaluate the quality of your solution, you will have to prepare an evaluation corpus. This should consist of some 1000 hand-transliterated term-pairs (from various sources). You will evaluate the accuracy of your system on these held-out data.

Variant: a more generic system will allow transliteration to *any* language. Two additional resources will be required:

- a monolingual (target) language model: you will use the monolingual projection of Wikipedia on the target language to create such a language model.
- a mapping of characters between the source and target languages: you will have to provide such mappings for a few language pairs.

3.5 Identifying synonyms using multilingual parallel texts

Introduction to Computational Linguistics recommended but not mandatory.

Synonyms are words that carry similar meaning and can usually be freely used in the same contexts: for example, *car, auto, automobile* or *predict, foretell, prognosticate*. A database of synonyms can be useful for a variety of natural language applications. The most wide-spread repository of synonyms is WordNet (Fellbaum, 1998), and variants in many languages have been created in the last decade.

Identifying synonyms is a non-trivial task. In this project you will use parallel corpora and a simple algorithm (Dyvik, 2002, 2005, 2009) to solve the problem. A parallel corpus (Koehn et al., 2005) is a collection of translated texts in two languages, where each sentence in the source language is aligned to its translation in the target language. Standard (statistical) algorithms exist that can align the words in a parallel corpus such that each source-language word is mapped to its possible translations in the target language, with a probability measure that determines the plausibility of the translation pair, *without a bilingual dictionary* (Koehn et al., 2007).

Once a parallel corpus is word-aligned, the word translation pairs whose probability is high can be used to extract synonyms by identifying translation *loops*. Let $E = \{e_1, e_2, \ldots e_n\}$ be a set of words in the source language and $F = \{f_1, f_2, \ldots f_n\}$ a set of words in the target language, such that for all i, $1 \leq i \leq n$, e_i is translated to f_i with high probability, and f_i is translated to $e_{i+1\ (mod\ n)}$ with high probability. Then we can assume that E is a set of synonyms in the source and F is a set of synonyms in the target. For example, if *ask* is translated to ביקש, ביקש to *request* and *request* to דרש, then we can assume that *ask, request* are synonyms, as are ביקש, דרש.

You will implement this algorithm using off-the-shelf tools for word alignment (Och and Ney, 2003). You will have to determine the confidence level required for determining that a loop is a good one. To test and evaluate your implementation, you will use the Europarl corpus to extract synonyms, and WordNet to verify them.

3.6 A classifier for Translationese

Introduction to Computational Linguistics recommended but not mandatory.

Translated texts are known to have linguistic properties that set them apart from texts written originally in the target language. Given the same domain and genre, translated texts tend to have shorter sentences, lower type/token ratio (i.e., less rich language), more limited syntactic constructions, etc. In this project you will use machine learning techniques to construct a classifier that can distinguish between translated and original texts in English, following Baroni and Bernardini (2006).

You will be provided with a training corpus consisting of newspaper articles in a single domain in English. The articles will be tagged as either translated (from three different languages) or original. Your main task will be to define a set of distinctive features and implement the feature extractor. Features may include superficial characteristics, such as the average length of sentences or the type/token ratio in a document; n-gram features, such as unigrams of function words, or specific bigrams or trigrams; or more linguistically-informed features, such as n-grams of part-of-speech tags, ratio of active to passive verbs, complexity of syntactic structures, etc. You will be able to use off-the-shelf tools for processing the corpus, and publicly-available machine learning packages for implementing the classifier.

Once the feature extractor is implemented, you will train a classifier on the training material and conduct a robust evaluation of the results. The result of this project will be used in a research on selecting the best language models for machine translation.

3.7 Grammar induction

Introduction to Computational Linguistics recommended but not mandatory.

A *grammar* is a concise representation of a set of sentences (a *language*). When children acquire language, they are exposed to a finite sample of the utterances in the language they are learning, but they are somehow able to generalize the finite sample to a coherent representation that has the potential to generate infinitely many utterances, including many novel ones (that the child was never presented with). Exactly how children find patterns in the ambient language and construct their grammars is for the most part unknown. Several psycholinguistic theories attempt to explain this process, but we are far from fully understanding it.

At the same time, computational linguists develop algorithms that learn (formal) grammars from raw data (Adriaans and van Zaanen, 2006). While such algorithms are not generally targeted at modeling child language acquisition, they are nonetheless interesting in this context. In this project you will implement such an algorithm and evaluate it on child language data.

The input to the algorithm is a set of utterances, a *corpus*; you will have access to several corpora recording spoken interactions between children and their caretakers (MacWhinney, 2000). The data are all precisely and consistently formatted. You will have to pre-process the input in order to prepare it for the format expected by the algorithm. A certain portion of the training corpus will be held out for evaluation; after training the algorithm, you will execute it on the held-out data and evaluate its ability to account for novel utterances. In order to assess over-generation, you will also execute the algorithm on non-utterances using the methodology of Kol et al. (2009).

The algorithms to implement are the following:

- Bayesian Model Merging (Stolcke and Omohundro, 1994)
- EMILE (Adriaans and Vervoort, 2002), and see The EMILE Homepage
- Alignment-based learning (van Zaanen, 2000, 2002a,b), and see the ABL Homepage
- MK10/SNPR (Wolff, 1982, 1988, 2003), and see here

When all projects are submitted, we will hold a competition among the various systems.

3.8 Unification Grammars

Introduction to Computational Linguistics required.

This is a very different kind of project. You will be required to read and fully understand a textbook on Unification Grammars. Your main task will be to fully solve numerous exercises scattered throughout the text. Some of the exercises are technical and easy, some require more thought.

4 Available resources

You may freely use the following available resources:

- Wikipedia as a source of multilingual texts, in particular in order to extract transliterated term-pairs
- Weka, a toolbox of various general-purpose machine learning tools, in particular in order to implement classifiers
- Open NLP, a set of tools for natural language processing, in particular in order to pre-process English texts
- NLTK, a natural language processing toolkit in Python.

References

Pieter Adriaans and Marco Vervoort. The EMILE 4.1 grammar induction toolbox. In Pieter Adriaans, H. Fernau, and Menno van Zaanen, editors, *ICGI 2002*, volume 2481 of *LNAI*, pages 293–295. Springer, Berlin and Heidelberg, 2002. doi: 10.1007/3-540-45790-9_24. URL http://dx.doi.org/10.1007/3-540-45790-9_24.

Pieter W. Adriaans and Menno M. van Zaanen. Computational grammatical inference. In Dawn E. Holmes and Lakhmi C. Jain, editors, *Innovations in Machine Learning*, volume 194 of *Studies in Fuzziness and Soft Computing*, chapter 7. Springer-Verlag, Berlin Heidelberg, Germany, 2006. ISBN: 3-540-30609-9.

Marco Baroni and Silvia Bernardini. A new approach to the study of Translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259–274, September 2006.

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16 (2):79–85, 1990.

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

Helge Dyvik. Translations as a semantic knowledge source. In *Proceedings of the Second Baltic Conference on Human Language Technologies*, Tallinn, 2005. Institute of Cybernetics at Tallinn University of Technology, Institute of the Estonian Language. Unpublished manuscript.

Helge Dyvik. Semantic mirrors. Unpublished manuscript, 2009.

Helge Dyvik. Translations as semantic mirrors: From parallel corpus to Wordnet. Unpublished manuscript, 2002. URL http://www.hf.uib.no/i/LiLi/SLF/Dyvik/ICAMEpaper.pdf.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. MIT Press, 1998.

Alon Itai and Shuly Wintner. Language resources for Hebrew. *Language Resources and Evaluation*, 42:75–98, March 2008.

Amit Kirschenbaum and Shuly Wintner. Minimally supervised transliteration for machine translation. In *Proceedings of The 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, April 2009.

Amit Kirschenbaum and Shuly Wintner. A general method for creating a bilingual transliteration dictionary. Under Review, 2010.

Philipp Koehn, Joel Martin, Rada Mihalcea, Christof Monz, and Ted Pedersen, editors. *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. The Association for Computational Linguistics, June 2007.

Sheli Kol, Bracha Nir, and Shuly Wintner. Acquisition of abstract slot-filler schemas: Computational evaluation. Presented at the COGSCI-2009 Workshop on Psychocomputational Models of Human Language Acquisition, July 2009.

Brian MacWhinney. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates, Mahwah, NJ, third edition, 2000.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Andreas Stolcke and Stephen M. Omohundro. Inducing probabilistic grammars by bayesian model merging. In *ICGI '94: Proceedings of the Second International Colloquium on Grammatical Inference and Applications*, pages 106–118, London, UK, 1994. Springer-Verlag. ISBN 3-540-58473-0.

Menno van Zaanen. Implementing alignment-based learning. In *ICGI '02: Proceedings of the 6th International Colloquium on Grammatical Inference*, pages 312–314, London, UK, 2002a. Springer-Verlag. ISBN 3-540-44239-1.

Menno van Zaanen. ABL: alignment-based learning. In *Proceedings of the 18th conference on Computational linguistics*, pages 961–967, Morristown, NJ, USA, 2000. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/992730.992785.

Menno van Zaanen. *Bootstrapping Structure into Language: Alignment-Based Learning*. PhD thesis, University of Leeds, Leeds, UK, January 2002b.

J. Gerard Wolff. Learning syntax and meanings through optimization and distributional analysis. In I M Schlesinger Y Levy and M D S Braine, editors, *Categories and Processes in Language Acquisition*, chapter 7, pages 179–215. Erlbaum, Hillsdale, NJ, 1988.

J. Gerard Wolff. Language acquisition, data compression and generalization. *Language and Communication*, 2:57–89, 1982.

Jerry G. Wolff. Information compression by multiple alignment, unification and search as a unifying principle in computing and cognition. *Artificial Intelligence Review*, 19(3):193–230, 2003. ISSN 0269-2821. doi: http://dx.doi.org/10.1023/A:1022865729144.

Shlomo Yona and Shuly Wintner. A finite-state morphological grammar of Hebrew. *Natural Language Engineering*, 14(2):173–190, April 2008.