Hebrew Computational Linguistics: Past and Future

Shuly Wintner
Department of Computer Science
University of Haifa

shuly@cs.haifa.ac.il

Abstract

This paper reviews the current state of the art in Natural Language Processing for Hebrew, both theoretical and practical. The Hebrew language, like other Semitic languages, poses special challenges for developers of programs for natural language processing: the writing system, rich morphology, unique word formation process of roots and patterns, lack of linguistic corpora that document language usage, all contribute to making computational approaches to Hebrew challenging. The paper briefly reviews the field of computational linguistics and the problems it addresses, describes the special difficulties inherent to Hebrew (as well as to other Semitic languages), surveys a wide variety of past and ongoing works and attempts to characterize future needs and possible solutions.

1 Introduction

Computational linguistics is a research area that lies in the intersection of linguistics and computer science. It can be viewed in two ways: on one hand, it is the application of various techniques and results from computer science to linguistics, in order to investigate such fundamental problems as what people know when they know a natural language, what they do when they use this knowledge, and how they acquire this knowledge in the first place. On the other, it is the application of various techniques and results from linguistics to computer science, in order to provide such novel products as computers that can understand everyday human speech, translate between different human languages, and otherwise interact linguistically with people in ways that suit people rather than computers. This latter view is usually known as *Natural Language Processing* (NLP).

Examples of NLP applications include machine translation from one natural language to another; conversion of speech to text and text to speech; natural language interfaces for computational systems; automatic summarization of documents; spelling and style checking; etc. We concentrate in this paper on natural language processing applications for the Hebrew language. We show that Hebrew poses additional problems for developers of programs for language processing, mainly due to its rich morphology and deficient script. In the next section we briefly review the field of computational linguistics and the problems it deals with, emphasizing the special problems involved in processing the Hebrew language. Section 3 surveys existing systems developed for Hebrew; to the best of our knowledge, this survey covers all works published to date. Finally, section 4 attempts to identify future needs and suggest directions for future progress.

2 The challenge

2.1 The complexity of Natural Language Processing

To understand the inherent complexity of natural language processing, consider the state of the art in two typical applications: question understanding and machine translation. The following example is taken from a web site which provides an Internet search engine based on natural language queries. The site (`http://www.ask.com`) provides references to documents which contain the answer to a user's question, and the references are displayed as questions, similar to the original query. Each reference is a link to a document in which the answer to the question can be found.

When a user poses the following question:

> *Who was the second president of the United States?*

the obtained answers are, as expected, precise (in other words, the references that the engine provides are very similar to the original question):

> *Where can I find information about U.S. President John Adams?*
>
> *Where can I find encyclopedic biographical resources on John Adams?*

However, when challenging the engine with the following question:

> *Who was the United States president following Washington?*

The obtained references are:

> *Where can I learn about the historical political figure George Washington?*
>
> *Where can I find information about U.S. President George Washington?*

In other words, it seems as if the software used by the search engine does not "understand" the meaning of the question; rather, it seems to rely on matching some key words, and since the term "George Washington" occurs in the question, it occurs also in the answers that the system provides.

A different experiment can be made by using the machine translation software of Systran Inc., publicly available at `http://babelfish.altavista.com`. The software provides translation of documents from eight different languages to English and vice versa. To check the performance of the software, we used it to translate the first paragraph of Carroll's *Alice in Wonderland* from English to Italian. The original paragraph is:

> *Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'*

The Italian translation provided by the system is:

> *Alice stava cominciando ad ottenere molto tired di seduta dalla sua sorella sulla banca e di non avere niente fare: una volta che o avesse pigolato due volte nel libro che la sua sorella stava leggendo, ma non avesse immagini o conversazioni in esso, 'e che cosa l'uso d'un libro,' pensiero Alice 'senza immagini o conversazione?'*

To test the quality of the translation, we translated it back to English with the same software. The result is:

> *Alice was beginning to very obtain tired of sitting from its sister on the bank and not to have nothing to make: once that or he had pigolato two times in the book that its sister was reading, but did not have images or conversations in it, 'and that what is the use of a book,' thought Alice 'without images or conversation?'*

The relatively low quality of the two systems demonstrated above stems from the inherent complexity of natural languages. The cognitive processes involved in understanding and generating utterances in natural languages are extremely complicated, and in any case lie beyond the comprehension we command today. Consequently, attempts to emulate some of these processes using a computer face obvious difficulties. To demonstrate the depth and complexity of natural languages, we survey below some necessary stages in computational implementations of complex applications, such as machine translation. The survey – which is language independent – is organized by different phases of linguistic processing, which correspond to different levels of linguistic knowledge. It emphasizes one phenomenon which makes computational processing difficult in each of the phases: *ambiguity*. We use the term ambiguity in the sequel to denote phenomena in which one form (e.g., a word or a sentence) can be analyzed in more than one way, and thus be associated with more than one representation.

It is worth reiterating that the following survey refers to complex applications. In recent years natural language technology has been successfully applied to several applications which require lower levels of language understanding. Examples of such applications include document categorization; automatic summarization of texts; automatic forwarding of e-mail messages; etc.

Phonology and phonetics These are the linguistic fields of research that deal with pronunciation; phonetics is concerned with the sounds that our articulators produce when we are speaking, their physical properties, their perception in our hearing organs etc. Phonology investigates these sounds in an abstract level, defines them and their combinations and investigates their function in the linguistic system. A computational system whose input or output is spoken language must include phonological knowledge in order to deal properly with phonological units (phonemes). Phonological ambiguity is expressed in various ways: the frequent case is homophones, i.e., different words with identical pronunciation, such as *week* and *weak* or *to, too* and *two* in English; or כלה and קלה in Hebrew. The problem is more interesting in Hebrew, in fact, as sometimes combinations of more than one word are pronounced similarly to other words or combinations thereof, as in הקלה (*relief*, nominalization of the verb הקל *relieve*) vs. הקלה (definite feminine form of the adjective קל *light*); or שלו (*his*, the possessive preposition *of* with a third person singular masculine pronoun) vs. שלא (the conjunction ש *that* followed by the negation לא *no*).

Other problems related to phonology include allophones (a single phoneme realized in different ways depending on its context, such as English /l/ that is realized differently depending on whether or not it follows a vowel); phonemes realized differently depending on the speaker (such as the differences in the pronunciation of the guttural sounds by different Israeli speakers); etc.

Morphology Morphology is concerned with the structure of words. It is almost inconceivable for a natural language application not to employ morphological knowledge: at the very least, applications require a lexicon or a dictionary. A system for intelligent internet search, for example, will also require a *morphological analyzer* in order to extract the base forms (lemmas) out of inflected forms that occur in web documents. Here, phenomena of ambiguity are especially evident, and are caused both by processes of derivation and by processes of inflection. For example, the suffix י- is a realization of two different morphemes in Hebrew: one is a possessive pronoun affix and the other is a derivational affix used to convert a noun to an adjective. Thus, the word ביתי can mean either *my home* or *homey*. One of the better examples for morphological ambiguity in Hebrew is the form שמנה, which has at least a dozen different analyses (such as the feminine form of the adjective שמן *fat*, or the noun שמן *oil* with a third person singular possessive pronoun, or the conjunction ש- *that* followed by the verb מנה *counted*, or even the conjunction ש-, followed by the noun מן *manna* with an attached possessive pronoun).

Syntax One of the major research areas in linguistics is syntax, which is concerned with the ways words combine to form phrases and phrases combine to form sentences. Syntax associates some structure with the utterances of a language, and can be highly ambiguous too. For example, in the sentence *I saw the spy with the black hat* the prepositional phrase *with the black hat* functions as an adjunct of the noun *spy*, whereas in *I saw the birds with a telescope* the prepositional phrase *with a telescope* functions as an adjunct of the verb *saw*. The syntactic structure of these two sentences is likely to be different, as the differences between the sentences must be reflected in their meanings. Therefore, any syntactic analyzer that has to deal with the sentence *I saw the spy with the telescope* will have to consider (at least) two different structures: *with the telescope* can either be an adjunct of *the spy* or of *saw*.

Many syntactic phenomena contribute to the difficulties of computational processing. One example is coordination phrases, which frequently cause ambiguity. For example, in the sentence *This paper describes research on analysis and generation of natural language*, the conjunction *and* can be interpreted as conjoining the noun *analysis* with the noun *generation*; but it can also be interpreted incorrectly, as conjoining *analysis* with the phrase *generation of natural language*. The difference can result in wrong translation to some other language, or in generating an incorrect meaning for the sentence.

Semantics Semantics deals with the meanings of words, phrases and sentences in natural languages. Semantic ambiguity starts in the lexicon: many a word has several different meanings, sometimes completely unrelated (such as the word *bank*, in the sense of a financial institution or a slope), and sometimes only slightly different, such as the many different senses of the verb *shake*. The Hebrew word פגישה can be translated to English as *meeting, appointment, date* or *rendezvous*, depending on the context. More interesting are phenomena of deep semantic ambiguity, such as the one that stems from the different possibilities to compute the scope of natural language quantifiers. For example, the sentence *all students did not pass the exam* can mean (in some idiolects of English) either that no student passed the exam, or that at least one student did not pass it.

Other problems related to semantic processing have to do with the resolution of pronominal references, or anaphora. For example, in the sentence *The conjunction 'and' can be interpreted as conjoining the noun 'analysis' with the noun 'generation', but it can also be interpreted incorrectly*, the pronoun *it* refers to a non-human, third person, singular noun phrase which occurrs earlier in the text. This phrase is *The conjunction 'and'*, but there is no principled reason why it couldn't be *the noun 'analysis'* or *the noun 'generation'* or even only *'generation'*. Of course, any computational system unable to resolve such anaphora might face severe difficulties.

Pragmatics Even when a sentence is semantically unambiguous, it sometimes carries secondary meanings, depending on the context in which it is uttered; pragmatics investigates such phenomena. For example, the sentence *I'll see you tomorrow* can be interpreted as a declarative sentence, denoting a fact; but in the right context, it can be interpreted as a promise; and in a different context it can be a threat. Another interesting phenomenon is presuppositions: the sentence *The current king of France is bald* implies that France is a monarchy; the sentence *I regret I voted for her* implies that I voted for her. A computational system designed to draw conclusions from natural language input will probably have to compute presuppositions in order to reach the right conclusions. Other phenomena with which pragmatics deals include non-verbal usage of language, such as irony, metaphor etc. Such phenomena pose extreme difficulties for computational processing.

2.2 The complexity of Hebrew

We surveyed above the main applications of natural language processing and the difficulties involved in them. It is not surprising, therefore, that the current state of the art is such that not all these problems can solved satisfactorily. Nevertheless, many problems connected to computational processing of natural languages are completely solved now, and others are partly solved, for languages that benefitted from extensive efforts. Thus, languages such as English, German or Japanese, but to a large extent also languages with a smaller set of users, such as Dutch, benefit from various applications that include contemporary technology based on computational linguistics.

The Hebrew language, on the other hand, is not part of the list. Of course, the fact that Hebrew has only a limited set of users makes the language less attractive for developers of new technologies. As shown above, many natural language processing applications require deep knowledge of the language, and hence transferring a technology from one language to another requires intensive effort and substantial resources. Moreover, many problems are unique to Hebrew (and other Semitic languages, which in general are less investigated); the result is that language processing is less advanced for Hebrew than it is for other languages.

We describe below the special difficulties that Hebrew poses for developers of natural language applications. In addition to the inherent difficulties that stem from the complexity of the human linguistic capacity, Hebrew adds two additional levels of complexity: the script and the morphology.

The Hebrew script The Hebrew writing system poses two main difficulties for developers of natural language processing applications: first, it differs from the Latin script, for which most of the existing technologies and applications were developed; second, it is written from right to left, and hence blocks simple adjustment of existing applications developed for languages that are written left to right.

The different alphabet requires special consideration even in simple applications, such as internet search engines or spell checkers. Of course, many European languages are written in scripts that differ from the one used for English, either completely (e.g., Russian or Greek) or by adding a small number of diacritics (such as the German umlaut). Therefore, every software developed for one script requires localization and adjustment for a different alphabet. But the Hebrew script poses more severe problems, especially because it is so lacking. The deficiencies of the Hebrew writing system are described in detail by Ornan (Ornan, 1977; Ornan, 1985a; Ornan, 1987; Ornan, 1994; Ornan and Katz, 1995); the major one is the fact that much of the information carried by words is not explicit in the script. In particular, in the unvocalized script, which is the standard one, most of the vowels are not explicit. On the other hand, many of the symbols are used for more than one purpose, for example as both consonant and vowel. As an example, consider the string שמנה. Depending on the context, it can be pronounced as *[$mena, $imna, $amna, $e-mana]* or even *[$mone]* or *[$amenna].* Similarly, מספר can be read *[mispar, mesapper, mi-sefer, mi-sapar]* or *[mi-sfar].*

Another problem is that morphemes which are regularly expressed as stand-alone words in many languages, such as prepositions, conjunctions, coordinations, articles etc., are realized as prefixes to other words in Hebrew. Thus, for example, the string וכשמהבית should be analyzed as ו-כש-מ-ה-בית (*and when from the house*), and שלביתך can be analyzed as ש-ל-בית-ך (*that for your house*).

As a consequence, the degree of ambiguity in written Hebrew is exceptionally high. According to Ornan (1977), the properties of the Hebrew script cause approximately half of the word forms to be homographs (i.e., belong to more than one lexical entry), and on the average more than four readings are available for every word. Bentur, Angel, and Segev (1992) claim that close to sixty percent of the words have more than one analysis, and one third of the words have more than two, while the mean number of readings for a set of 300 frequent words turned out to be 2.7. In an analysis of large newspaper texts containing 40,000 word tokens, the mean number of analyses per word was found to be 2.1, while 55 percent of the words had more than one analysis (Levinger, Ornan, and Itai, 1995; Sima'an et al., To appear).

An additional dimension of complexity stems from the various writing styles in Hebrew. There exists a traditional system of diacritics (*nikkud*) which is used to indicate the vocalization of written forms. The system is used rarely today, mostly for children books and texts for language learners. The unvocalized script (sometimes referred to as “undotted”), does not have an accepted standard with respect to which of the vowels are actually depicted. Thus, several words have more than one accepted written form; and while some texts try to adhere to a particular style, many others use different styles interchangeably, thus increasing the degree of ambiguity.

Hebrew morphology An additional reason for the difficulties of computational processing of Hebrew is the morphology of the language. Inflectional processes in Hebrew are based primarily on affix concatenation, as is the case in many European languages (and, in fact, in most languages of the world). However, Hebrew has both prefixes and suffixes, and sometimes both kinds of affixes combine with a single base form (as in the verb inflection ת+שמר+ו *guard*, second person, plural, future tense). Furthermore, the base form itself can be modified in the different paradigms. Thus, for example, it is useful to view the past tense as the base of the inflection, but this base is altered in the future tense (for example, שמר is changed to שמור as in אשמור, תשמור, etc.).

But problems of inflection are dwarfed by the complexities posed by derivational processes in Hebrew, and in particular the major word formation process which is based on interdigitation of roots (sequences of three, four or sometimes even five or six consonants) and patterns (sequences of vowels and possibly consonants, with “empty slots” into which consonants of the root fit). These processes cannot be characterized in a straight-forward way using concatenation only, and more elaborate mechanisms are used in order to account for them. In particular, while concatenative processes can be described (and implemented) using regular expressions and finite-state automata, thus guaranteeing their efficient implementation, it is more difficult (albeit not impossible) to describe root and pattern morphology using such machinery.

3 Past

This section describes existing computational systems for processing Hebrew. In spite of the difficulties, and notwithstanding the relatively little commercial interest in the Hebrew language, considerable effort has been invested in developing Hebrew processing applications, especially in academia but to some extent also in industry. Due to the complexities involved in semantic and pragmatic processing, existing systems concentrate on phonological and morphological aspects of the language, and to a lesser extent also on syntax. So far, the publicly available computational infrastructure for processing Hebrew proves highly insufficient; in other words, linguistic corpora, on-line lexicons and dictionaries, morphological analyzers and generators, morphological disambiguators, part of speech taggers or word sense disambiguation programs that can be used by researchers interested in deeper processing of Hebrew are few in number and limited in their usefulness. It is therefore almost impossible to construct more elaborated systems such as machine translation programs, which must have access to such infrastructure.

Many of the earlier works dealing with Hebrew appeared in the journal *Hebrew Linguistics*, published by “The computational linguistics committee of Bar Ilan University” since 1969. Unfortunately, in recent years the percentage of papers dealing with *computational* linguistics in this journal decreased. First we find a paper describing an algorithm for generating Hebrew words (Price, 1969). A subsequent algorithm for morphological analysis was published two years later (Price, 1971a). The second issue includes an abstract of a doctoral thesis, probably the first work ever to deal with automatic translation from Hebrew to English (Price, 1970), and a paper describing an algorithm for extracting the root of Hebrew words (Lazewnik, 1970). One of the early issues even includes an automatic method for Hebrew hand writing recognition, although no computational implementation of the algorithm is mentioned (van der Toorn, 1971). Another source for papers describing research on Hebrew computational linguistics is the volume of this name (Ornan, Arieli, and Doron, 1992), which includes several papers presented in symposia held by the Israeli Ministry of Science and Technology between 1988 and 1990. As a side note we should mention that these symposia were recently resumed, and they serve as a forum for presenting works done in Israel, with an emphasis on works dealing with Hebrew and Arabic (Wintner, 2001). The following survey lists, to the best of our knowledge, all the papers dealing with computational processing of Hebrew ever published.

Morphology Ornan (1977) describes the historical dictionary project of the Hebrew Language Academy (starting 1964) as the first serious project for Hebrew processing. This project, however, is better classified as the use of a computer as an aid in a linguistic application (lexicography), as it lacks any significant computational linguistic processing. The same class includes works whose objective is to use a computer in order to collect statistical information on Hebrew verbs (Morgenbrod and Serifi, 1976; Morgenbrod and Serifi, 1977; Morgenbrod and Serifi, 1978). At the other extreme stands the work of Azar (1970), describing a full algorithm for the analysis of all the Hebrew words occurring in the Old Testament, which was never implemented on a computer.

Shapira and Choueka (1964) were probably the first to construct a computational system for processing Hebrew. The system was written in assembler on a Philco computer. Naturally, this is a very basic system, but its achievements are considerable: the paper describes both a full morphological analysis of the nominal system (the verbal part was not completed in that version) and an application of the analysis for preparing a concordance and a bibliography list. In a subsequent paper, Choueka (1966) describes the part of the system that deals with verbal words.

The work described above was the introduction to a large-scale project dealing with various aspects of computational linguistics, natural language processing and information retrieval: the Responsa project which began in 1967, headed first by Aviezry Fraenkel and later by Yaacov Choueka (Choueka, 1972; Fraenkel, 1976; Choueka, 1980). The project was based on a data base of 102 volumes of religious questions and answers, collected in the span of some fourteen centuries. Its objective was information retrieval, construction of concordances and provision of query services that use the data base. To this end, tools for linguistic processing, and in particular algorithms for morphological analysis, needed to be developed (Attar et al., 1978). Algorithms were developed for automatic generation of all the possible inflected and derived forms of all the bases in Hebrew, including those obtained by the combination of prepositions, conjunctions, articles etc. Based on the generation algorithm, a file was created which included all the possible Hebrew word forms, approximately 2,500,000 words. The analyzer implements a program which strips the possible affixes off the input word and checks whether the obtained result is indeed a legal word. Thus, morphological analysis and generation are incorporated in a complete system for computational processing of Hebrew (albeit not Modern, contemporary Hebrew).

Later, Choueka (1990) developed a system, called MLIM, in the framework of the Rav-Milim project of the Center for Educational Technology. This system was adapted for Modern Hebrew, and was used as the infrastructure for two major applications: a program for vocalization (adding the vowel diacritics to the unvocalized script of Hebrew) and an on-line dictionary which also includes a morphological analyzer. These applications were converted into commercial systems and therefore were never made accessible to researchers in Hebrew computational linguistics; this is unfortunate as this system offers a morphological processor that is the most broad-coverage and robust ever to be developed for Hebrew. See Choueka (1993) for a short description of this system.

A different approach to Hebrew morphology was developed by Ornan. Based on his observations regarding the limitations and the difficulties that the Hebrew script poses for computational processing (see section 2.2), Ornan (1986) proposes the *Phonemic Script*, which is an unambiguous writing system for Hebrew, preserving the deep structure of the words. Based on this script, a wide variety of programs were developed, including a program for vocalization (Ornan, 1985b), a program for the preparation of concordances and indexes (Ornan, 1985a), especially developed for a data base of legal texts (Ornan, 1987), a series of programs for morphological analysis and generation (Ornan and Kazatski, 1986; Shany-Klein, 1990; Goldstein, 1991; Shany-Klein and Ornan, 1992) and programs for converting phonemic script to the standard Hebrew script (Ornan and Katz, 1995). Recently, some of the above mentioned linguistic algorithms were integrated into a number of commercial products, such as programs for dictating texts (to be used by the blind), a multi-lingual search engine and systems for information retrieval.

Two other systems for morphological analysis of Hebrew were integrated into more complex systems, aimed at syntactic processing. One was created as part of a doctoral dissertation (Cohen, 1984; Cohen, 1985); morphological analysis is done in two phases: first, the program finds all the possible "stems" of a word, after which all the possible inflections of each stem are generated from the lexicon and compared with the input word. In a sense, this is a similar algorithm to the one described above (Attar et al., 1978). The system also selects the most "probable" analysis, using short-context considerations described as preference rules. The reported success rate is fifty percent, but no detailed evaluation of the results is provided. The other system, HUHU (Nirenburg and Ben-Asher, 1984), was developed at the Hebrew University of Jerusalem and includes a component of morphological analysis which precedes the syntactic analysis. The algorithm is based on a dictionary which contains, for each entry, linguistic information concerning all the possible inflections of the word; and a set of rules describing the affixes and the way they combine with the various lexemes. In addition, the affixes and clitics that can be combined with other words are independently handled.

Morphological analysis is one aspect of a commercial system, Context, designed for information retrieval (Pinkas, 1985). The objective of the system is to retrieve documents (in Hebrew and English) from a pre-defined database, according to a given user query. To this end, "semantic proximity" of words is defined which links together different words that share morphological, phonetic or semantics properties. The system does not use a dictionary, and the paper does not provide any evaluation of its coverage or the quality of its results.

Another commercial system, Avgad, was developed at IBM Research Center in Haifa (Bentur et al., 1992; Bentur, Angel, and Segev, 1992). It is based on a dictionary of 25,000 entries, which form the base for "hundreds of thousands" of Hebrew words (including inflected forms). The words in the lexicon were selected "out of the common words of Modern Hebrew". The system was incorporated into a number of word processors, its main usage being spelling checkers. It was used by Segal (1997) in order to construct a freely available morphological analyzer: the analyzer was built by automatically generating possible base forms, inflecting them in all possible ways and verifying the results against the existing analyzer.

Morphological disambiguation In most of the systems described above the functionality of the morphological analyzer is limited to producing all the valid analyses of each word. As noted above, Hebrew has a high degree of morphological ambiguity (compared with European languages). Therefore, many applications call for morphological disambiguation: determining, using the context in which a word occurs and a set of preference rules, the most likely analysis of the word, given the set of its valid analyses.

Three main ways exist for using short context in order to disambiguate: the first, which is the most likely to succeed, is by collecting an exhaustive list of word sequences, each containing an ambiguous word, manually indicating the correct analysis of the word in its context. Such a solution, where word sequences are of length 2, is proposed by Choueka and Lusignan (1985); clearly, in the case of Hebrew this would require enormous resources in order to list all the possible sequences of two words, and it seems that the solution is inapplicable to Hebrew. A similar proposal is made also by Cohen (1984) and Levinger (1992).

Another way is based on extracting information about the short context of words from an existing grammar. The idea underlying this technique is that if one could syntactically analyze a given sentence, the analysis would have disambiguated many of the words in the sentence. Syntactic analysis is computationally expensive, but if a grammar is available which describes the syntax of the language, then it can be used for automatically generating information on the basis of which words can be efficiently disambiguated. Such a technique, with an application to Hebrew spell checking, is proposed by Herz and Rimon (1991) and Herz and Rimon (1992).

The third approach uses statistical methods. If a corpus is given in which all analyses of each word are listed, with the correct one indicated, statistical methods can be applied to learn data from the corpus and extrapolate them to new examples that were never tagged. This approach was implemented by Levinger, Ornan, and Itai (1995): the core of this work is an algorithm which automatically computes a good approximation for the morpho-lexical probability of each word in a given text: this is the conditional probability that an analysis is indeed the correct one for the word. The paper presents a systematic way for computing these probabilities given an untagged corpus. Based on the morpho-lexical probabilities, the paper suggests to select for each word the analysis with the highest probability. The work was implemented using the Avgad morphological analyzer (Bentur, Angel, and Segev, 1992), evaluated in detail and the paper reports high success rates.

A similar approach is used by Segal (1999): texts are analyzed using the morphological analyzer of Segal (1997); then, each word in a text is assigned its most likely analysis, defined by probabilities computed from a small tagged corpus. In the next phase the system corrects its own decisions by using short context (one word to the left and one to the right of the target word). The corrections are also automatically learned from the tagged corpus. In the last phase, the analysis is corrected by the results of a syntactic analysis of the sentence. The reported results are excellent, but the performance of the program is unacceptable (the reported running time on “two papers” is thirty minutes). Omitting the syntactic analysis, the results are good and the performance is reasonable.

Another disambiguator that is based on Avgad is described by Carmel and Maarek (1999). To overcome the problem of data sparseness, this system makes decisions based on the frequencies of “morphological patterns” associated with the analyses of input words, rather than the analyses themselves. The patterns consist of parts of speech and other morphological information, but essentially not the lexeme. Here, too, good results are reported.

A different approach, also based on statistical methods, is proposed by Dagan and Itai (1994). It is based on differences between the mappings of words to senses in different languages, and assumes that in one language – for example, English – statistical data exist which can be used to determine the right analysis of ambiguous words. Given a syntactic analyzer which can recognize functional relations (such as subject–verb, or verb–object) in the source language (here, Hebrew), the different readings of the words in each instance of such a relation can be mapped to the other language; the data of the target language can then be used for disambiguation of the source language word. This approach was tested on Hebrew and German, using English as the target language, and yielded good results. The major obstacle of this method is that it crucially relies on a syntactic analyzer for Hebrew.

Syntax No broad-coverage, linguistically motivated syntactic analyzer for Hebrew exists. The main obstacle in achieving this goal is the absence of a broad-coverage, large-scale computational grammar for the language. Existing grammatical descriptions of Modern Hebrew, including those who claim to be formal (Rosen, 1966; Rubinstein, 1968; Rubinstein, 1970; Chayen and Dror, 1976; Ornan, 1979; Glinert, 1989), are insufficient for computational processing.

An interesting attempt to formulate a computational grammar for Hebrew was made by Price (1971b), who describes a phrase structure grammar of the syntax of Modern Hebrew. 179 rules are stipulated, out of which only 111 were tested. The paper describes 26 sentences correctly parsed by the grammar, all simple and short (at most 10 words) and most of them of a similar structure. This was a pioneering work, but its achievements were minor.

“Mechanical” (non-computational) syntactic analysis of Hebrew texts is described by Azar (1972). The work describes an algorithm for syntactic analysis and stipulates forty rules. The grammar is over generative: the rules are not sufficiently refined and will permit generation of many ungrammatical structures. The major drawback of the work, however, is that the algorithm was never implemented and hence never tested practically.

The first syntactic analyzer for Hebrew is probably HUHU (Nirenburg and Ben-Asher, 1984), developed at the Hebrew University of Jerusalem. The system is based on augmented transition networks (ATN), and receives as input sentences in unvocalized Hebrew. It first performs a morphological analysis of the input words and then a syntactic analysis based on the output of the previous stage. The coverage of the grammar developed for the system is unclear, but according to the list of categories and the description of some rules it seems to be rather limited. The analyzer works in a non-deterministic fashion and produces all the possible analyses for each sentence.

Cohen (1984), also, attacked the problems of syntax and morphology in tandem, and his system, too, uses as input unvocalized Hebrew script. High rates of success are reported, but the analyzer was tested on fifty sentences only. The number of syntactic rules in the grammar is sixty. One of the major problems of this system is that the grammar is integrated with the parsing algorithm, so that any change in or modification of a particular grammar rule requires an update of the entire program.

Albeck (1995) proposes a new approach for syntactic analysis. Observing the cognitive process of human sentence processing, and in particular disambiguation, she suggests an algorithm in which words are assigned unique senses immediately upon their processing, with no look-ahead and no backward error correction. This is done using a variety of rules, both mandatory and optional, which create, update and cancel expectations pertaining to the function of the current word in a sentence. The described system consists of 192 syntactic rules, sufficient for producing the correct analysis for 98 percent of the words in a text consisting of 100 sentences. The work limits the input sentences to those that have “accepted” word order, “ordered and fixed” sentence structure and are “full and clear”. Also, the paper does not explain or exemplify the syntactic structure assigned to sentences, but concentrates rather on the morphological disambiguation. The computational performance of the system is not discussed either.

Clearly, syntactic analysis depends crucially on the choice of linguistic theory. Over the last two decades, some linguistic theories emerged that are sufficiently formal and which lend themselves more easily to computational implementation. Such theories are characterized by relatively high expressivity, and in any case are more expressive than context-free grammars. Most of them represent linguistic information – the lexicon, the rules, the representation of phrases and even the result of the analysis – using complex structures, called *feature structures* (Shieber, 1986); and most of them rely, to some extent, on *unification* as the major operation on feature structures. Such theories include *LFG – Lexical Functional Grammar* (Kaplan and Bresnan, 1982; Dalrymple et al., 1995), *HPSG – Head-Driven Phrase Structure Grammar* (Pollard and Sag, 1987; Pollard and Sag, 1994), *TAG – Tree-adjoining Grammar* (Joshi, 1987) and *CG – Categorial Grammar* (Haddock, Klein, and Morill, 1987; Steedman, 2000). These theories serve as a platform for theoretical linguistic investigations, resulting in computational grammars that describe a wide variety of phenomena in a wide variety of natural languages. Most importantly, such grammars can be tested and executed computationally.

A stand-alone computational grammar for Hebrew, independent of a particular parsing algorithm, was first written by Wintner (1991). The work included a survey of a few grammatical formalisms and their fitness for designing a computational grammar for Hebrew (Wintner and Ornan, 1991a), a small grammar written in PATR, the simplest unification-based formalism (Wintner, 1992) and a broader grammar, based on the principles of the linguistic theory LFG (Wintner and Ornan, 1991b; Wintner and Ornan, 1996). The grammar produces a dual description: the output lists both the phrase structure of the input sentences and a functional structure. It was tested on a small corpus of sentences in simplified Hebrew, taken from the newspaper *Sha'ar*, and the accuracy on this corpus is 75 percent.

Yizhar (1993), focusing on nominal phrases, worked in a similar framework. This work provides a comprehensive description of noun phrases in Hebrew, in an approach influenced by LFG, and the grammar was implemented and tested. Other works dealing with the syntax of Modern Hebrew, with computational implementations of grammar fragments inspired by HPSG, include a computational lexicon of the verbal system (Skoblikov, 2000); a small computational grammar (Wintner, 1997) which is expanded to a detailed analysis of noun phrases (Wintner, 1998); and an analysis of relative clauses (Vaillette, 2001).

Although computational grammars developed in unification based theories can be used, in theory, for both parsing and generation, the complications involved in sentence generation are substantial, and in practice grammars designed for analysis are not suitable for generation. The problems involved in the generation of Hebrew sentences are tackled in a sequence of works: Dahan Netzer (1997) adapted a system developed by Elhadad for generation of noun phrases in Hebrew; other works are concerned with the specific problems involved in generation of construct-state phrases (Dahan Netzer and Elhadad, 1998b), quantifiers (Dahan Netzer and Elhadad, 1998a) and prepositional phrases (Dahan Netzer and Elhadad, 1999).

Other works A first program for speech synthesis in Hebrew is described by Laufer (1976). It is a system which generates artificial speech from input given in a phonetic representation (which includes the location of the stress), using transfer rules (implementing relationships among adjacent phonemes) and prosodic rules. It is interesting to note that IBM Haifa Research Lab is currently developing a system for speech recognition in Hebrew, a problem which is much harder than synthesis.

Very few works deal directly with semantics. Samuelsdorff (1980) describes semantic analysis for machine translation, but the work is not based on any theory and does not report on its performance beyond the single word level. Still at the word level, Nissan (1993) proposes a system for coining new Hebrew terms, but its practical importance is unclear. Bashkansky and Ornan (1998) propose a set of tools for supporting machine translation, with an application to translation from Hebrew to Russian. A system for machine translation using semantic features, with an application to translation from Hebrew to English and Spanish, is described by Ornan and Gutter (2000). In both cases, the systems are limited and can be viewed as human aided machine translation tools.

4 Future

As shown in the previous section, many systems for processing Hebrew exist, but a wide, well-founded and freely available infrastructure that will facilitate the development of modern software as well as support advanced research in Hebrew computational linguistics is still missing. In this section we survey the requirements that such an infrastructure has to satisfy.

As noted above, almost any application that requires linguistic knowledge makes some use of a full lexicon of the language. Natural languages are dynamic: it is impossible to determine what the full word list of a given language is. The main reason is that foreign words, and in particular proper names, are constantly added to the language, and are very frequent in common texts (such as newspaper reports or internet documents). Furthermore, foreign words (including names) are frequently integrated into the language, and obey to some extent the morphological rules of the language. In Hebrew this phenomenon is even more frequent: the word formation process of Semitic languages enables the extraction of a root from foreign words; this root can then be conjugated in a variety of patterns. Thus, the noun טלפון *telephone* is the basis for the addition of the root *t.l.p.n* to Hebrew, facilitating the formation of the verb טלפן *to place a phone call* with its entire paradigm. Computational lexicons must be sensitive to the dynamics of the language, and it is therefore advantageous to invest in the development of technology that will enrich existing lexicons with new words, using knowledge acquired by constantly searching new documents (for example, by exploring internet documents and producing a list of previously unseen words).

But the lexicon is only the first stage in computational processing of words. In a language such as Hebrew, with a rich and productive morphology, it is a mistake to maintain a lexicon of all the inflected forms of all lexemes. The morphological rules of Hebrew are relatively simple, and it is beneficial to represent them formally and precisely, so as to facilitate the construction of modern morphological processors that are easy to maintain, modify, extend and improve as the lexicon expands. Most of the works discussed in the previous section view the problems of Hebrew morphology as inherently different from those of other languages, presumably because for many years computational morphology was a non-existent research topic, due to the intense investigation dedicated to English, a language with a very simple morphology. Since the 1980's, however, a unique research area has developed which deals with computational approaches to natural language morphology (and phonology), based on finite-state technology (Sproat, 1992; Kaplan and Kay, 1994; Roche and Schabes, 1997). This approach was pioneered by Koskenniemi (1983), who developed a computational model for describing morphological and phonological processes with finite-state machines, known as the *two-level model*. The first attempt to test the suitability of this model to Hebrew was made by Lavie et al. (1988a), and included a new implementation of the model in Prolog (Lavie et al., 1988b; Lavie, 1989). The conclusions of the experiment are that while the model is suitable for describing the inflectional morphology of the verbal system, certain derivational processes, such as the formation of minor bases from the major ones, are very difficult to express in this framework.

Recently, however, finite-state based morphology has been progressing fast, and extensions of finite-state models were suggested which provide better expressivity (Karttunen et al., 1996; Mohri, 1996; Mohri, Pereira, and Riley, 1998; van Noord and Gerdemann, 2001; Beesley and Karttunen, 2003). These extensions facilitate the description of morphological processes using a convenient description language and automatic compilation of expressions in the language to finite-state automata and transducers. These description languages are basically extended regular expression languages, but they include a vast variety of additional operators that are extremely useful to the linguist. Of course, since the expressions are compiled into finite-state devices, which can then be determinized and minimized, computational efficiency is guaranteed.

The extended models were used for the construction of morphological analyzers for a few Semitic languages, including Arabic (Beesley, 1996; Beesley, 1998; Kiraz, 2000). It is possible – indeed, necessary – to adopt this technique and use it for Hebrew, producing modern morphological analyzers and generators for all levels of the language which will be freely available and constantly maintained.

The techniques for morphological disambiguation discussed above are promising, and they should be further developed as components of a modern morphological analyzer. Based on their results, *shallow parsers* can be constructed which will produce a skeleton of the syntactic structure of a sentence much faster than what full syntactic analysis would have required. Finite-state technology can certainly be used for such applications as well. At Ben Gurion University of the Negev ongoing projects aim at transferring existing methods for part of speech tagging to Hebrew (Adler and Tebeka, 2001). For Hebrew, part of speech tagging will most likely require full morphological disambiguation.

A benefit of such works is their applicability to other Semitic languages. The morphology of Hebrew is very close to that of other Semitic languages, notably Arabic. Therefore, tools and techniques developed for Hebrew are very likely to be usable for linguistic research and development of systems for processing Arabic. Indeed, many systems exist for processing Arabic; but it seems that the development of technology that will facilitate applications in more than one language will be useful. For example, an ongoing project at the University of Haifa aims at developing technology that will facilitate automatic acquisition of a lexicon and a morphological analyzer from a given corpus (Talmon and Wintner, 2001). Hopefully, the same technology will be usable for learning a lexicon and morphological rules for Hebrew.

In addition to linguistic applications such as lexicons and morphological processors, resources must be dedicated to the development of tools for constructing computational systems for language processing. In a wide variety of applications the current state of the art is such that approaches based on statistical knowledge, acquired automatically, reach better performance than methods based on deep linguistic understanding (speech recognition is such an application). In order to collect the statistical data, and in particular in order to *train* the systems, large linguistic corpora are required. Such corpora, of various levels of the language, both written and spoken, are a necessity for statistical processing. Furthermore, in order to train the systems, tagged corpora are frequently needed where linguistic information (be it part of speech, or the entire morphological analysis of words, or even basic sentence structure) is indicated. Well investigated languages, such as English or German, can benefit also from syntactically parsed and annotated corpora, where each sentence is associated with a tree describing its structure. Such corpora are known as *tree banks* and are invaluable for applications based on shallow linguistic knowledge. Preparation of such corpora is a long, arduous and expensive process, requiring mostly the labor of computational linguists with sufficient knowledge of the language and its syntax and at least some computational background. Undoubtedly such corpora for Hebrew are extremely important and well worth investing in.

Two ongoing projects are concerned with the construction of linguistic corpora for Hebrew. As they are still under development, we only describe their objectives here. The first project is concerned with the construction of a tree bank for Hebrew (Sima'an et al., To appear). In the first stage, a set of 500 sentences was selected, which were automatically analyzed morphologically and then manually corrected. Then, the sentences were analyzed syntactically by hand, which resulted in the assignment of a single parse tree for each sentence. The aim of the project is to develop a semi-automatic procedure for expanding the set of trees in the future.

A different corpus deals with spoken Hebrew (Izre'el, Hary, and Rahav, To appear): the objective of this work is to construct a representative corpus of recordings of spoken Modern Israeli Hebrew that will be used for research purposes in a variety of areas. The corpus will be accompanied by transcriptions of the recordings and by tools that will facilitate search, comparisons and analyses of both the spoken data and their transcriptions. This project is still in its infancy.

For many applications, a good lexicon, accompanied by a disambiguating morphological analyzer will suffice. For example, the most successful methods for automatic summarization (Mani and Maybury, 1999; Mani, 2001) rely on shallow linguistic knowledge and do not require deeper understanding. The same applies to applications of intelligent knowledge-base search, information extraction etc. For deeper processing, however, such as question answering, machine translation etc., it is necessary to understand the structure of utterances – in other words, syntactic analysis is required.

As noted in the previous section, very few computational grammars were developed for Hebrew. In order to develop natural language applications that will include an understanding component, development of large scale, broad coverage computational grammars is imperative. It was proven several times that for such complex applications, the performance of systems based on deep linguistic knowledge is better than that of systems based on statistical data only. Such approaches require substantial effort and resources, usually necessitating collaboration among several teams, including theoretical linguists, computational linguists and computer scientists, but this effort is always worthwhile.

When a broad-coverage computational grammar of Hebrew exists, based on a disambiguating morphological analyzer, and capable of dealing with complex input, both grammatical and ungrammatical, the greatest challenge of natural language processing research can be attempted: understanding the meaning of utterances, starting with simple phrases, through whole sentences and culminating in full discourses. However, we believe that this challenge will be easier than the previously discussed tasks. Linguistic phenomena exhibit higher variation the lower the linguistic level is. Thus, while the morphology of English and Hebrew are completely different, their syntax – while considerably different – still shows some universal similarities, and their semantics – with the exception of lexical semantics, i.e., the meanings of words – is substantially more language-independent.

When such a system for processing Hebrew exists, the road will be clear for a wide variety of applications to be developed, as well as for initiating challenging research projects that cannot be dealt with today. Only then will Hebrew become an equal-rights member in the class of languages that can be processed computationally. Needless to say, such developments cannot be driven by the needs of the software industry, as the market for Hebrew processing programs is bound to be very limited. Such developments, both under basic research and under more applicative research, must be driven by those of us who are concerned with the future of the Hebrew language in an era of fast globalization.

Acknowledgments

I am grateful to Mori Rimon, Uzzi Ornan and Alon Itai for commenting on earlier drafts of this paper. I am especially indebted to Shlomo Izre'el for his encouragement and many useful comments. This work was supported by the Israeli Science Foundation (grant no. 136/1).

References

Adler, Meni and Miki Tebeka. 2001. Unsupervised Hebrew part-of-speech tagging. In Shuly Wintner, editor, *Israeli Seminar on Computational Linguistics (ISCOL'01)*, pages 19–20, Haifa, February.

Albeck, Orly. 1995. A formal method for analyzing a Hebrew sentence. *Hebrew Linguistics*, 39:5–27, August. In Hebrew.

Attar, R., Y. Choueka, N. Dershowitz, and A. S. Fraenkel. 1978. KEDMA - linguistic tools for retrieval systems. *Journal of the Association for Computing Machinery*, 25(1):52–66, January.

Azar, Moche. 1970. Analyse morphologique automatique du texte hébreu de la Bible. Technical Report 12 et 19, Faculte des Lettres et des Sciences Humaines, Nancy.

Azar, Moche. 1972. Automatic syntactical analysis: the method and its application to the Book of Ruth. *Hebrew Computational Linguistics*, 5:1–50, February. In Hebrew.

Bashkansky, Guy and Uzzi Ornan. 1998. Monolingual translator workstation. In *MT and the Information Soup: Proceedings of AMTA'98*, pages 136–149. Springer, October.

Beesley, Ken. 1996. Arabic finite-state morphological analysis and generation. In *Proceedings of COLING-96, the 16th International Conference on Computational Linguistics*, Copenhagen.

Beesley, Kenneth R. 1998. Arabic morphology using only finite-state operations. In Michael Rosner, editor, *Proceedings of the Workshop on Computational Approaches to Semitic languages*, pages 50–57, Montreal, Quebec, August. COLING-ACL'98.

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite-State Morphology: Xerox Tools and Techniques*. Cambridge University Press.

Bentur, Esther, Aviella Angel, and Danit Segev. 1992. Computerized analysis of Hebrew words. *Hebrew Linguistics*, 36:33–38, December. in Hebrew.

Bentur, Esther, Aviella Angel, Danit Segev, and Alon Lavie. 1992. Analysis and generation of the nouns inflection in Hebrew. In Ornan et al. (Ornan, Arieli, and Doron, 1992), chapter 3, pages 36–38. In Hebrew.

Carmel, David and Yoelle Maarek. 1999. Morphological disambiguation for Hebrew search systems. In *Proceedings of the 4th international workshop, NGITS-99*, number 1649 in Lecture notes in computer science, pages 312–325. Springer, July.

Chayen, M. J. and Z. Dror. 1976. *Introduction to Hebrew Transformational Grammar*. University Publishing Projects Ltd., Jerusalem. In Hebrew.

Choueka, Yaacov. 1966. Computers and grammar: mechnical analysis of Hebrew verbs. In *Proceedings of the Annual Conference of the Israeli Association for Information Processing*, pages 49–66, Rehovot. In Hebrew.

Choueka, Yaacov. 1972. Fast searching and retrieval techniques for large dictionaries and concordances. *Hebrew Computational Linguistics*, 6:12–32, July. In Hebrew.

Choueka, Yaacov. 1980. Computerized full-text retrieval systems and research in the humanities: The Responsa project. *Computers and the Humanities*, 14:153–169.

Choueka, Yaacov. 1990. MLIM - a system for full, exact, on-line grammatical analysis of Modern Hebrew. In Yehuda Eizenberg, editor, *Proceedings of the Annual Conference on Computers in Education*, page 63, Tel Aviv, April. In Hebrew.

Choueka, Yaacov. 1993. Response to “Computerized analysis of Hebrew words”. *Hebrew Linguistics*, 37:87, December. in Hebrew.

Choueka, Yaacov and Serge Lusignan. 1985. Disambiguation by short context. *Computers and the Humanities*, 19:147–157.

Cohen, Daniel. 1984. *Mechanical Syntactic Analysis of a Hebrew Sentence*. Ph.D. thesis, Hebrew University of Jerusalem. In Hebrew.

Cohen, Daniel. 1985. Analysis of unvocalized texts. In *Proceedings of the Ninth World Congress of Jewish Studies*, pages 117–122, Jerusalem, August. World Union of Jewish Studies. In Hebrew.

Dagan, Ido and Alon Itai. 1994. Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics*, 20(4):563–596, December.

Dahan Netzer, Yael. 1997. HUGG – unification-based grammar for the generation of Hebrew noun phrases. Master’s thesis, Ben-Gurion University of the Negev, Department of Computer Science, Faculty of Natural Sciences, Be’er Sheva, Israel, November.

Dahan Netzer, Yael and Michael Elhadad. 1998a. Generating determiners and quantifiers in Hebrew. In Michael Rosner, editor, *Proceedings of the Workshop on Computational Approaches to Semitic Languages (COLING/ACL’98)*, pages 82–88, Montreal, Canada.

Dahan Netzer, Yael and Michael Elhadad. 1998b. Generation of noun compounds in Hebrew: Can syntactic knowledge be fully encapsulated? In Eduard Hovy, editor, *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 168–177, New Brunswick, New Jersey. Association for Computational Linguistics.

Dahan Netzer, Yael and Michael Elhadad. 1999. Hebrew-English generation of possessives and partitives: Raising the input abstraction level. In *Proceedings of the 37th meeting of the ACL*, pages 144–151, Maryland.

Dalrymple, Mary, Ronald M. Kaplan, John T. Maxwell, and Annie Zaenen, editors. 1995. *Formal Issues in Lexical-Functional Grammar*, volume 47 of *CSLI lecture notes*. CSLI, Stanford, CA.

Fraenkel, Aviezri S. 1976. All about the Responsa retrieval project – what you always wanted to know but were afraid to ask. *Jurimetrics Journal*, 16(3):149–156, Spring.

Glinert, Lewis. 1989. *The Grammar of Modern Hebrew*. Cambridge University Press, Cambridge.

Goldstein, Lyor. 1991. Generation and inflection of the possession inflection of Hebrew nouns. Master's thesis, Technion, Haifa, Israel. In Hebrew.

Haddock, Nicholas, Ewan Klein, and Glyn Morill, editors. 1987. *Categorial Grammar, Unification and Parsing*, volume 1 of *Working Papers in Cognitive Science*. University of Edinburgh, Center for Cognitive Science.

Herz, J. and M. Rimon. 1991. Local syntactic constraints. In *Proceedings of the Second International Workshop on Parsing Technologies*, Cancun, Mexico.

Herz, J. and M. Rimon. 1992. Lexical disambiguation and other applications of short context automata. In Ornan et al. (Ornan, Arieli, and Doron, 1992), chapter 7, pages 74–87. In Hebrew.

Izre'el, Shlomo, Benjamin Hary, and Giora Rahav. To appear. Designing CoSIH: The corpus of Spoken Israeli Hebrew.

Joshi, Aravind K. 1987. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

Kaplan, Ronald and Joan Bresnan. 1982. Lexical functional grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, Mass., pages 173–281.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, September.

Karttunen, Lauri, Jean-Pierre Chanod, Gregory Grefenstette, and Anne Schiller. 1996. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):305–328.

Kiraz, George Anton. 2000. Multitiered nonlinear morphology using multitape finite automata: a case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105, March.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production*. The Department of General Linguistics, University of Helsinki.

Laufer, Asher. 1976. Computer generated artificial Hebrew speech. *Leshonenu*, 40:67–78. In Hebrew.

Lavie, Alon. 1989. Two-level morphology for Hebrew. Master's thesis, Technion, Haifa, Israel. In Hebrew.

Lavie, Alon, Alon Itai, Uzzi Ornan, and Mori Rimon. 1988a. On the applicability of two-level morphology to the inflection of Hebrew verbs. Technical Report 513, Department of Computer Science, Technion, 32000 Haifa, Israel.

Lavie, Alon, Alon Itai, Uzzi Ornan, and Mori Rimon. 1988b. On the applicability of two-level morphology to the inflection of Hebrew verbs. In *Proceedings of the International Conference of the ALLC*, Jerusalem, Israel.

Lazewnik, Rabbi Grainom. 1970. Construction of an algorithm for stem recognition in the Hebrew language. *Hebrew Computational Linguistics*, 2:84–101, May.

Levinger, Moshe. 1992. Morphologic disambiguation in Hebrew. Master's thesis, Technion, Haifa, Israel. In Hebrew.

Levinger, Moshe, Uzzi Ornan, and Alon Itai. 1995. Learning morpho-lexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics*, 21(3):383–404, September.

Mani, Anderjeet. 2001. *Automatic Summarization*. John Benjamins, Amsterdam.

Mani, Anderjeet and Mark T. Maybury, editors. 1999. *Advances in Automatic Text Summarization*. The MIT Press, Cambridge, Mass.

Mohri, Mehryar. 1996. On some applications of finite-state automata theory to natural language processing. *Natural Language Engineering*, 2(1):61–80.

Mohri, Mehryar, Fernando Pereira, and Michael Riley. 1998. *A Rational Design for a Weighted Finite-State Transducer Library*. Number 1436 in Lecture Notes in Computer Science. Springer.

Morgenbrod, M. and E. Serifi . 1976. Computer-analysed aspects of Hebrew verbs. *Hebrew Computational Linguistics*, 10:E1–17, April.

Morgenbrod, M. and E. Serifi . 1977. Computer-analysed aspects of Hebrew verbs: Mathematical models. *Hebrew Computational Linguistics*, 12:E1–18, August.

Morgenbrod, M. and E. Serifi . 1978. Computer-analysed aspects of Hebrew verbs: The binjanim structure. *Hebrew Computational Linguistics*, 14:V–XV, November.

Nirenburg, Sergei and Yosef Ben-Asher. 1984. HUHU – the Hebrew University Hebrew understander. *Computer Languages*, 9(3/4).

Nissan, Ephraim. 1993. Onomaturge: An expert system for word formation. *Hebrew Linguistics*, 36:39–49. In Hebrew.

Ornan, Uzzi. 1977. Report on linguistic research in the computer carried on in Israel. *Hebrew Computational Linguistics*, 11:121–127, February. In Hebrew.

Ornan, Uzzi. 1979. *The Simple Sentence*. Academon, Jerusalem, Israel. In Hebrew.

Ornan, Uzzi. 1985a. Indexes and concordances in a phonemic Hebrew script. In *Proceedings of the Ninth World Congress of Jewish Studies*, pages 101–108, Jerusalem, August. World Union of Jewish Studies. In Hebrew.

Ornan, Uzzi. 1985b. Vocalization by a computer: a linguistic lesson. In Ben-Zion Luria, editor, *Avraham Even-Shoshan Book*. Kiryat-Sefer, Jerusalem, pages 67–76. In Hebrew.

Ornan, Uzzi. 1986. Phonemic script: A central vehicle for processing natural language – the case of Hebrew. Technical Report 88.181, IBM Research Center, Haifa, Israel.

Ornan, Uzzi. 1987. Computer processing of Hebrew texts based on an unambiguous script. *Mishpatim*, 17(2):15–24, September. In Hebrew.

Ornan, Uzzi. 1994. Basic concepts in “Romanization” of scripts. Technical Report LCL 94-5, Laboratory for Computational Linguistics, Technion, Haifa, Israel, March.

Ornan, Uzzi, Gideon Arieli, and Edit Doron, editors. 1992. *Hebrew Computational Linguistics: Papers presented at seminars held in 1988, 1989, 1990*. Ministry of Science and Technology. In Hebrew.

Ornan, Uzzi and Israel Gutter. 2000. Machine translation by semantic features. In Derek Lewis and Ruslan Mitkov, editors, *Machine Translation and Multilingual Applications in the New Millennium*, Exester, UK, November.

Ornan, Uzzi and Michael Katz. 1995. A new program for Hebrew index based on the Phonemic Script. Technical Report LCL 94-7, Laboratory for Computational Linguistics, Technion, Haifa, Israel, July.

Ornan, Uzzi and Wadim Kazatski. 1986. Analysis and synthesis processes in Hebrew morphology. In *Proceedings of the 21 National Data Processing Conference*. In Hebrew.

Pinkas, Gadi. 1985. A linguistic system for information retrieval. *Maase Hoshev*, 12:10–16, December. In Hebrew.

Pollard, Carl and Ivan A. Sag. 1987. *Information Based Syntax and Semantics*. Number 13 in CSLI Lecture Notes. CSLI.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications.

Price, James D. 1969. An algorithm for generating Hebrew words. *Hebrew Computational Linguistics*, 1:51–54, September. Reprinted from *Computer Studies in the Humanities and Verbal Behavior*, **1**(2):84–102, 1969.

Price, James D. 1970. The development of a theoretical basis for machine aids for translation from Hebrew to English. *Hebrew Computational Linguistics*, 2:65–83, May. Abstract of a Doctoral Dissertation, The Dropsie College for Hebrew and Cognate Learning, Philadelphia.

Price, James D. 1971a. An algorithm for analyzing Hebrew words. *Computer Studies in the Humanities and Verbal Behavior*, 3(2):137–165.

Price, James D. 1971b. A computerized phrase structure grammar (Modern Hebrew). Report F-C2585-1/2/3/4, Franklin Institute.

Roche, Emmanuel and Yves Schabes, editors. 1997. *Finite-State Language Processing*. Language, Speech and Communication. MIT Press, Cambridge, MA.

Rosen, Haiim B. 1966. *Ivrit Tova (Good Hebrew)*. Kiryat Sepher, Jerusalem. in Hebrew.

Rubinstein, Eliezer. 1968. *Ha-mishpat Ha-shemani (The Nominal Sentence)*. Merhavia: Ha-Kibbutz Ha-Me’uxad. In Hebrew.

Rubinstein, Eliezer. 1970. *Ha-cerup Ha-pooliy (The Verb Phrase)*. Merhavia: Ha-Kibbutz Ha-Me’uxad. In Hebrew.

Samuelsdorff, Paul Otto. 1980. Computational analysis of Modern Hebrew. *Hebrew Computational Linguistics*, 16:IV–XVI, August.

Segal, Erel. 1997. Morphological analyzer for unvocalized hebrew words. Unpublished work, available from `http://www.cs.technion.ac.il/˜erelsgl/hmntx.zip`.

Segal, Erel. 1999. Hebrew morphological analyzer for Hebrew undotted texts. Master's thesis, Technion, Israel Institute of Technology, Haifa, October. In Hebrew.

Shany-Klein, Michal. 1990. Generation and analysis of Segolate noun inflection in Hebrew. Master's thesis, Technion, Haifa, Israel. In Hebrew.

Shany-Klein, Michal and Uzzi Ornan. 1992. Analysis and generation of Hebrew Segolate nouns. In Ornan et al. (Ornan, Arieli, and Doron, 1992), chapter 4, pages 39–51. In Hebrew.

Shapira, Meir and Yaacov Choueka. 1964. Mechanographic analysis of Hebrew morphology: possibilities and achievements. *Leshonenu*, 28(4):354–372. In Hebrew.

Shieber, Stuart M. 1986. *An Introduction to Unification Based Approaches to Grammar*. Number 4 in CSLI Lecture Notes. CSLI.

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

Skoblikov, Victoria. 2000. Feature-based computational lexicon of Hebrew verbs. Master's thesis, Technion, Israel Institute of Technology, Haifa, Israel, January.

Sproat, R. W. 1992. *Morphology and Computation*. MIT Press, Cambridge, MA.

Steedman, Mark. 2000. *The Syntactic Process*. Language, Speech and Communication. The MIT Press, Cambridge, Mass.

Talmon, Rafi and Shuly Wintner. 2001. Computational processing of spoken North Israeli Arabic. In *Arabic Language Processing: Status and Prospects*, pages 124–126, Toulouse, France, July. Association for Computational Linguistics.

Vaillette, Nathan. 2001. Hebrew relative clauses in HPSG. In Dan Flickinger and Andreas Kathol, editors, *Proceedings of the 7th International Conference on Head-Driven Phrase Structure Grammar*. CSLI Publications.

van der Toorn, A. J. 1971. Automatic reading of handwritten Hebrew. *Hebrew Computational Linguistics*, 4:83–99, July.

van Noord, Gertjan and Dale Gerdemann. 2001. Finite state transducers with predicates and identity. *Grammars*, 4(3).

Wintner, Shuly. 1991. Syntactic analysis of Hebrew sentences. Master's thesis, Technion, Israel Institute of Technology, Haifa, Israel, July. In Hebrew, abstract in English.

Wintner, Shuly. 1992. Syntactic analysis of Hebrew sentences using PATR. In Ornan et al. (Ornan, Arieli, and Doron, 1992), chapter 9, pages 105–115. In Hebrew.

Wintner, Shuly. 1997. *An Abstract Machine for Unification Grammars*. Ph.D. thesis, Technion – Israel Institute of Technology, Haifa, Israel, January.

Wintner, Shuly. 1998. Towards a linguistically motivated computational grammar for Hebrew. In Michael Rosner, editor, *Proceedings of the Workshop on Computational Approaches to Semitic Languages (COLING-ACL'98)*, pages 82–88, Université de Montréal, Quebec, Canada, August. Association for Computational Linguistics.

Wintner, Shuly, editor. 2001. *Israeli Seminar on Computational Linguistics (ISCOL'01)*, Haifa, February.

Wintner, Shuly and Uzzi Ornan. 1991a. Computational models for syntactic analysis – their fitness for writing a computational grammar for Hebrew. In *Proceedings of the Bar-Ilan Symposium on Foundations of Artificial Intelligence*, June. Also as CIS Report 9103, Center for Intelligent Systems, Technion, May 1991.

Wintner, Shuly and Uzzi Ornan. 1991b. Syntactic analysis of Hebrew sentences. In *Proceedings of the 8th Israeli Symposium on Artificial Intelligence and Computer Vision*, pages 201–230. Information Processing Association of Israel, December.

Wintner, Shuly and Uzzi Ornan. 1996. Syntactic analysis of Hebrew sentences. *Natural Language Engineering*, 1(3):261–288, September.

Yizhar, Dana. 1993. Computational grammar for Hebrew noun phrases. Master's thesis, Computer Science Department, Hebrew University, Jerusalem, Israel, June. In Hebrew.