

Morphological Disambiguation for Hebrew Search Systems

David Carmel and Yoëlle S. Maarek

IBM Haifa Research Laboratory, MATAM, Haifa 31905, Israel,
{carmel,yoelle}@haifa.vnet.ibm.com

Abstract. In this work we describe a new approach for morphological disambiguation to enable linguistic indexing for Hebrew search systems. We describe a Hebrew Morphological Disambiguator (HMD or Hemed for short) based on statistical data gathered from large Hebrew corpora. We show how to integrate HMD with a search engine to enable linguistic search for Hebrew. We report some experimental results demonstrating the the superiority of linguistic search over string-matching search, and the contribution of morphological disambiguation to the quality of search result.

1 Background and Motivation

With the advent of the Web, more and more textual information is being made available on line, and Information Retrieval (IR) systems are becoming of crucial importance to search through the vast amount of information. Most state-of-the-art IR systems operate on a canonical representation of documents called a *profile* that consists of a list (or a vector in the commonly used vector space model [13]) of indexing units and other representative terms¹. Using a canonical representation makes the processing of documents more convenient. Profiles can be stored in an inverted index for fast retrieval, compared to each other for automatic clustering, abstracted into rules for categorization, etc. Since profiles are intended to provide a “conceptual representation” of documents, it is highly desirable that identical concepts be mapped into the same indexing unit. Thus, inflections of the same word such as plural and singular forms of a noun, or different tenses of a verb, should be represented by the same base unit (*lemma*). Indeed, a user issuing a query about “vitamins” on the Web, does not expect that only articles containing the exact string “vitamins” match his/her query but rather that any article dealing with the concept will be returned. For instance, a document dealing with “Vitamin A” should be considered relevant.

While researchers have been proposing various approaches to start addressing the problem of “conceptual ambiguity”, the problem of “morphological ambiguity”, at least for English, has been mostly solved by using either a stemmer or a morphological analyzer. Morphological analyzers use a set of declination rules and a dictionary, while stemmers use ad-hoc suffix stripping rules and exception

¹ Representative terms can be words, phrases, syntactic constructs [4], lexical affinities [8], etc.

list for these rules depending on the desired quality of the output and the language. One of the most widely used public-domain stemmers for English is the Porter's stemmer [11] which has been shown to give fast and good enough results for most applications such as search. However, stemmers, being less precise than analyzers, cannot be used for all applications. They can be less effective for other languages such as German, or even simply useless for languages such as Hebrew.

Indeed, Hebrew as well as other Semitic languages, is a highly synthetic language with rich vocalic changes such as deletion, insertion, substitution, and affixation. In standard Hebrew writing not all the vowels are represented, several letters represent both consonants and vowels, and gemination is not represented at all [10]. The major problem posed by the extensive morphology is the difficulty to identify the lexical base of a given word since for many input strings it is not clear which letters belong to the base and which have been affixed. This does not only make the definition of stemmers rules much too complex (they would require such as large exception list that it would be preferable to use a dictionary instead), but also significantly complicates the task of Hebrew morphological analyzers that typically return multiple possible analyses [6, 7]. For example, the Hebrew word *mishtara* can be analyzed as:

1. *mishtara* (police).
2. *mishtar + a* (her regime).
3. *mi + shtar + a* (from her bill).

The result of the complex morphology of the Hebrew language is a high level of word ambiguity in comparison to other languages. Highly ambiguous Hebrew words can have up to 13 different analyses. The average number of possible analyses per word is 2.15. More precisely, 55% of the words have more than one reading, and 33% of the words have more than two readings [7]. With such level of ambiguity, regular morphological analyzers cannot be used as such with indexing systems in order to generate document profiles. In a search application for instance, if all analyses returned by the analyzer are stored inside the index, the precision of the results will be very low. For example, a user looking for information about "police stations" (*mishtara* in analysis 1) will be utterly confused to get answers relevant to regimes or bills (analyses 2 and 3).

Therefore, indexing requires ideally a unique lemma (or at most a very small set of possible lemmas) in order not to reduce precision to which users are immediately sensitive. Hence, most Hebrew search engines simply disable morphological analysis to come back to a more primitive string matching search to circumvent this problem. They prefer missing relevant documents (low recall) than exposing the user to totally irrelevant ones (low precision). Note that for conceptual ambiguity, low precision is less annoying to users as they can easily figure out that the word "bank" in English² can be ambiguous, while morphology can be so complex that users will not understand at first sight how a document

² Bank has several senses, one refers to a financial establishment and another to the bank of a river.

dealing with the violinist *Izhak Stern* relates to their query *Mishtara*³.

In this paper, we propose to reduce the problem of morphological ambiguity of Hebrew via a statistical approach that takes advantage of an existing morphological analyzer, as well as statistical data automatically derived from large Hebrew corpora. Section 2 describes related work on morphological disambiguation for Hebrew. In Section 3 we describe our disambiguation procedure in more detail and explain the role of its basic components. We also describe our Hebrew Morphological Disambiguation algorithm (HMD, nicknamed Hemed) and the way we gather the necessary statistical data. In Section 4, we show how Hemed has been integrated with an existing Hebrew search engine and we describe some experiments that demonstrate the necessity and contribution of morphological disambiguation to Hebrew search. Finally, Section 5 summarizes the contribution of this work.

2 Related Work

The Responsa Project at Bar-Ilan University [12] has been a twenty-year endeavor to compile an electronic database of Jewish scriptural texts. The database includes the Tanach and its commentators, the Babylonian Talmud with Rashi's commentary, the Jerusalem Talmud, Rambam, Midrashim and more. It includes a Hebrew search engine that extends the user's query by adding all grammatical forms of words included in the query, using a morphological engine. Such a query extension improves the search recall significantly but deteriorates search precision. In order to improve precision, Responsa lets the user to refine her query interactively [2].

Reducing the morphological ambiguity of Hebrew is crucial for linguistic indexing as well as for other natural language applications. Ornan [9] developed a new writing system for Hebrew called "Phonemic Script". This script enables Hebrew writing that is morphologically unambiguous. However, this script has not become popular and widely used.

Choueka and Lusignan [3] present a morphological disambiguator based on *short contexts* of words in order to resolve ambiguity, but this disambiguator depends heavily on human interaction. Levinger et al. [7] propose an approach for acquiring *morpho-lexical probabilities* from an un-tagged corpus. The probabilities can be used as an information source for morphological disambiguation. Their *context-free* approach which handles each word separately is motivated by the observation that

"... in many cases a native speaker of the language can accurately guess the right analysis of a word, without even being exposed to the concrete context in which it appears ..." [7].

³ The Hebrew term *Stern* can also be read as *Shtaran* (their bill in feminine form – an inflection of analysis 3 of the input word *Mishtara*).

In this work we describe a similar context-free approach for morphological disambiguation that reduces the Hebrew morphological ambiguity problem so as to enable linguistic indexing of Hebrew text-files.

3 The Hebrew Morphological Disambiguator, Hemed

Our Hebrew Morphological Disambiguator, Hemed, receives the output of an Hebrew Morphological Analyzer and prunes the number of candidate analyses. The chosen analyses are kept inside the index. A similar disambiguation process is performed to analyze the user's query. We propose a new method for disambiguation based on the following principle. Instead of dealing with words, we deal with morphological patterns as basic elements for disambiguation. Pruning is done by evaluating the likelihood of each analysis pattern, using statistical data which reflects the relative frequency of the morphological patterns in a typical Hebrew text. The statistical data will be gathered from a large un-tagged Hebrew corpus using only unambiguous words.

The general architecture for Hemed is illustrated in Figure 1.

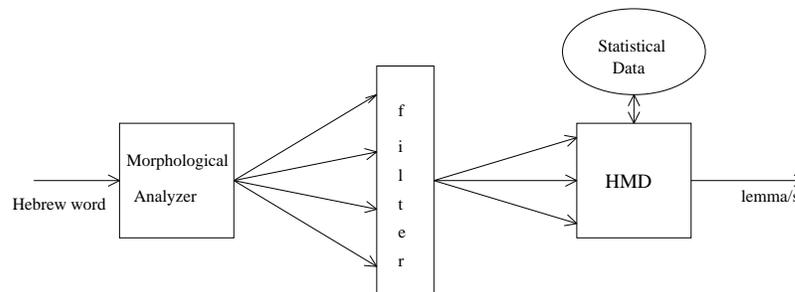


Fig. 1. The general architecture for Hemed.

The input word is analyzed by a morphological analyzer which returns all legal morphological analyses. The latter are passed to a “simple filter” which eliminates the impossible or extremely improbable ones using some basic morphological rules. The remaining analyses are fed to Hemed which does the core of the disambiguation work and returns a very limited number of most likely analyses per word. Hemed estimates the probability of each analysis by using statistical data on the relative frequency of Hebrew morphological analyses.

In a search application for instance, only the best base form(s), or lemma(s) of the analyses returned by the disambiguator are stored inside the index for later retrieval, and the same procedure is applied at querying time. A key feature of Hemed, is that the number of valid analyses kept can be controlled via a threshold parameter ϵ , and thus the precision/recall rate can be fixed at the time the query is issued as long as enough information has been stored in the index.

Given an ambiguous word with more than one analysis, Hemed filters out all analyses with lower relative frequency than this threshold. As ϵ becomes larger, more candidate analyses are filtered out but the probability of filtering out the “true” analysis also increases. In the following, we describe the basic components of this architecture in more detail.

3.1 The Morphological Analyzer

Hemed requires the output of a morphological analyzer. In our implementation we have taken advantage of the existing *Avgad* product – a Hebrew morphological analyzer developed at IBM - Haifa Research Laboratory [1]. However, Hemed can easily be modified to work with any other morphological analyzer that provides similar information in a different format.

For each legal Hebrew string, Avgad returns all legal lexical candidates as possible analyses of the given string. The information returned by Avgad for each analysis includes:

- category – categorization of the base forms according to part of speech, gender, plural inflections, legal set of prefixes and legal set of suffixes.
- part of speech.
- prefix – attached particles.
- lemma – the base form used for indexing.
- correct form - (optional) the input word with additional vowel letters added by Avgad to enable the given analysis.
- number, gender, person.
- status – (for non verbs only) – whether this lemma is in its construct or absolute form.
- tense – (for verbs only)
- inf_num , inf_gen, inf_person – for pronoun suffix.

For example, Table 1 shows the six analyses returned by Avgad for the string *Inyany*:

lemma	pos	number	gender	person	status	example
<i>Inyany</i>	adj.	sing.	masc.			<i>Mammar Inyany</i> – a practical paper
<i>Inyan</i>	noun	sing.	masc.		const.	<i>Inyany Shely</i> – my interest
<i>Inyan</i>	noun	plural	masc.		const.	<i>Inyanay Shely</i> – my interests
<i>Inyan</i>	noun	plural	masc.		const.	<i>Inyaney hamedina</i> – the state interests
<i>Anyen</i>	verb	sing.	fem.	2		<i>Anyeny</i> – make them interested
<i>Ana</i>	verb	sing.	fem.	2		<i>Anyny</i> – (<i>Any li</i> – answer me, or <i>Any oti</i> – torture/ me)

Table 1. The analyses returned by Avgad for the word *Inyany*

The sixth analysis illustrates well Avgad’s feature of returning infrequent morphological patterns that are generally not used in a typical modern Hebrew text, but are yet valid ones.

3.2 The Simple Filter

The simple filter component eliminates some of the analyses that are not relevant for indexing. The first rule consists of filtering out all the corrected forms, i.e., all analyses inferred by adding vowel letters to the original string, to enable the given analysis. This rule is motivated by the assumption that only the original string is a candidate to be indexed (or to be searched). The second rule consists of filtering out all analyses with the same lemma and the same category, leaving out only one representative base form for this set. The intuition behind this rule is that different inflections of the same lemma do not add information that should be stored in the index. For example, analyses 2,3,4, in the example shown in Table 1, are all inflections of the noun *Inyan* and therefore can be represented by one single analysis inside the index. Using the simple filter reduces the average number of analyses per word from 2.15 to 1.91.

3.3 Hebrew Morphological Patterns

Our disambiguation method uses morphological patterns of the analyses returned by Avgad as the basic elements for disambiguation. The morphological disambiguator makes decisions based on the frequency of the morphological patterns associated with the analyses of the input word. Infrequent patterns are pruned using statistical data gathered from a large Hebrew corpus and only the most likely set of analyses are returned.

A morphological pattern is defined according to the information returned by the morphological analyzer. Table 2 shows all legal values for all analysis entries:

field	number	values
pos	12	<i>Shem Etzem, Poal, Shem Toar, Mispar, Milat Yachas, Milat Guf, Milat Shela, Milat Hibur, Mlit, Toar Hapoal, Notricon, Shem Prati</i>
prefix	7	<i>moshe-vecalev</i> letters (keeping only the last letter of the prefix)
number	2	<i>yachid, rabim</i>
gender	3	<i>zachar, nekeva, zachar/nekeva</i>
person	4	1, 2, 3, all
tense	5	<i>Avar, Hove, Atid, Tzivui, Shem Poal</i>
Binyan	7	<i>Paal, Nifal, Piel, Pual, Hitpael, Hifil, Hufal</i>

Table 2. Legal values of the analysis entries returned by Avgad.

For non-verbs, a morphological pattern is defined to be a tuple of (pos, prefix, number, gender, person, status, inf_num, inf_gen, inf_person). For verbs, we add the *Binyan* field to the morphological pattern.

Using morphological patterns instead of words avoids many of the problems related to accumulating word statistics. Pattern statistics are much more reliable and are much easier to manage since there are much fewer patterns than words. 2300 different patterns were found in a corpus of 10 million Hebrew words compared to 25,000 unique Hebrew words.

Pattern statistics are computed by scanning the corpus concerning only unambiguous words. Since 45% of the Hebrew words are unambiguous [6], the sample size includes approximately 4.5 million words. For each identification of an unambiguous word with one legal analysis we increment the counter of its associated pattern. The pattern counters are stored in a global hash table for efficient retrieval.

3.4 Context-free Analysis

Pattern statistics related to the morphological patterns are used as follows. Analyses associated with infrequent morphological patterns are filtered out by the context-free Hemed algorithm which receives a threshold parameter from the user and filters out all analyses with lower relative frequency than this threshold.

The relevance score of any document retrieved for a given input query depends on the frequency of the query terms inside the document. Since we might keep more than one lemma per word inside the index, there might be a bias for ambiguous words since all their lemmas contribute to the accumulated score. In order to avoid this effect, we store each lemma with its relative frequency and this value is used while computing the relevance score for the retrieved documents.

The Context-free Hemed algorithm, described in Figure 2, takes as input a word w and a threshold parameter ϵ and returns a set of (lemma, weight) pairs:

4 Experiments with Hemed

We conducted a set of experiments to test the contribution of Hemed to Hebrew search which is the main information retrieval application⁴. In the first experiment we tested the accuracy of Hemed. In the second we added Hemed to an existing Hebrew search engine and tested the effect on the search results.

4.1 Evaluation of the accuracy of Hemed

In order to evaluate the accuracy of Hemed we manually tagged a set of 16,000 Hebrew words. Each word was associated with its “true” analysis, chosen manually from the set of analyses return by Avgad. The accuracy of Hemed computed

⁴ Comparable tests could be conducted for other IR applications such as categorizing but we believe that search is the application that suffers most from the ambiguity problem.

```

Context-free-HMD( word w, threshold  $\epsilon$ )
  Analysis[]  $\leftarrow$  Avgad(w)
  n  $\leftarrow$  |Analysis|
  if n = 0 /* for illegal words we use the input word as a base form.*/
    return (w,1)
  else
    if n = 1 /* one analysis: no dilemma, use its lemma as a base form */
      return (Analysis[1].lemma,1)
    else /* more than one analysis */
      Lemmas  $\leftarrow$  {}
      for i = 1 to n
        let Freq[i] be the relative frequency of pattern(Analysis[i])
        if Freq[i]  $\geq$   $\epsilon$ 
          Lemmas  $\leftarrow$  Lemmas + (Analysis[i].lemma, Freq[i])
      return Lemmas

```

Fig. 2. Hemed: The Context-Free HMD algorithm

by counting the number of words for which the set of analyses returned by Hemed includes the “true” analysis.

Figure 3 shows the accuracy of Hemed as a function of the threshold parameter ϵ . The upper curve shows the results for all the tagged words. The lower curve shows the results when tested only for ambiguous words. We can see that for low thresholds, only a few analyses are pruned and the accuracy is close to 100%⁵. For large threshold parameters more analyses are pruned and the accuracy decreases. For $\epsilon = 0.5$, Hemed returns the most likely analysis and prunes all other candidates. In this case the accuracy deteriorates to 86%.

Figure 4 shows the percentage of words with different number of analyses as a function of the threshold ϵ . As ϵ becomes lower the number of ambiguous words increases as expected but only few words have more than three analyses. For example, consider $\epsilon = 0.1$, we see that the accuracy of Hemed is 95%, according to figure 3, but the average number of analyses per word is kept low: 75% of the words are assigned one analysis, 20% of the words have two and 5% of the words have more than two analyses. Only a negligible number of words has 4 analyses and no one has more than four. These are very reasonable results to be applied for linguistic indexing.

4.2 Hemed contribution to Hebrew Search Engine

Most free-text search engines comprise two components: an indexing component and a retrieval component. The indexing component analyzes a set of documents from a given collection and extracts from each of them a set of meaningful

⁵ The reason that the accuracy is not 100% for $\epsilon = 0$ is that the simple filter has also few errors in pruning.

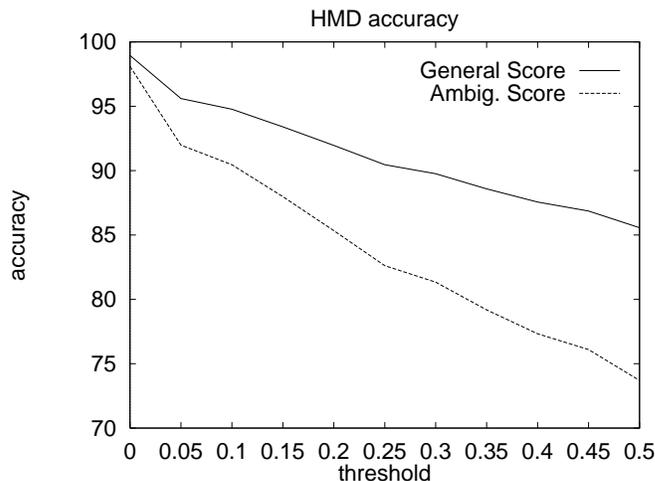


Fig. 3. The accuracy of Context-free Hemed as a function of the threshold parameter. The upper curve shows the general accuracy measured for all words. The lower curve shows the number of successes among ambiguous words.

indexing units. The retrieval component is given as input a query, and returns a list of references to those documents that are most relevant to the query. This list can be ranked according to a numerical score that represents the relevance to the query.

The performance of a search engine is usually measured by two evaluation criteria:

Recall: the ratio between the number of relevant items retrieved to the number of relevant items in collection.

Precision: the ratio between the number of relevant items retrieved to the number of retrieved items.

The most common measures used for evaluating the retrieval effectiveness of search engines in the information retrieval community are the average recall and precision of the search results and the Recall/Precision graph which shows both criteria simultaneously. Typically, when precision goes up, recall goes down and vice versa. Such measures can be evaluated for individual queries, or averaged over a set of queries as described in [14]. Recall and precision are not absolute measures in the sense that they strongly depend on the chosen test collection and therefore can only be used for comparative purpose. The performance of two search engines can be compared by comparing the average recall/precision measures of the engines for the same set of queries.

We have integrated Hemed into *BabaGuru*, a Hebrew search engine based on the search engine *Guru* [8], to evaluate its contribution. Each word in the text collection is analyzed by Hemed⁶ while creating the index. We used 3 different

⁶ Not including stop-words, i.e., common words not used for indexing.

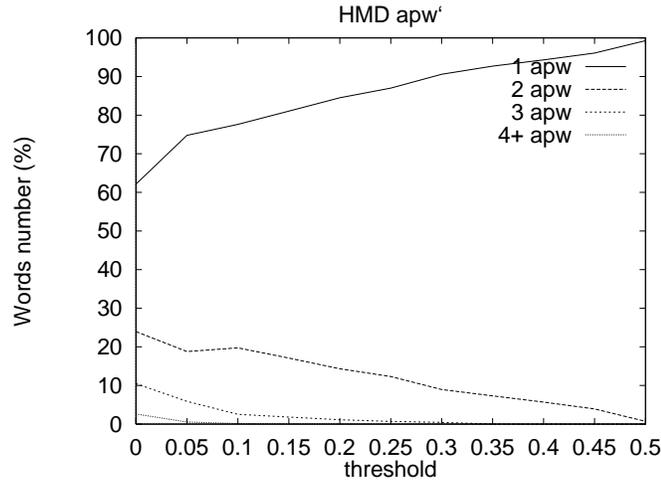


Fig. 4. The percentage of words with different analyses number (apw) as a function of the threshold parameter.

thresholds, 0, 0.1, 0.5 to create three different repositories for the same collection of documents. For $\epsilon = 0$ all analyses are kept inside the repository (not including those filtered out by the simple filter). For $\epsilon = 0.5$ only the most likely analysis is kept inside the repository. Hemed is also applied for the query terms, using a threshold parameter determined by the user. The analyses returned by Hemed are searched inside the index repository. In addition, we experimented with a string-matching search engine which applies *precise indexing* and a simple string-matching retrieval function.

A collection of 900 news articles, including approximately 400K Hebrew words, were indexed by BabaGuru and by the string-matching search engine. Table 3 shows the search results of the different search engines for the query *Mishtara*. The string-matching search engine retrieved only 17 documents, compared to the 123 documents retrieved by Hemed versions with a threshold 0.1 and 0.5, and 162 documents retrieved by Hemed with a threshold of $\epsilon = 0$. String matching had poor results since it ignores all inflections of the query term – *Hamishtara*, (the police), *Mehamishtara* (from the police), *Mishtarot* (plural form), *Mishteret*, (constructed form), and many others. All versions of Hemed succeeded to retrieve all relevant documents dealing with *Mishtara*. However, the $\epsilon = 0$ version also retrieved 39 non-relevant documents who deal with *Mishtar* (regime) and *Shtar* (bill). Among those documents, 4 deal with the violinist *Itzhak Stern* whose name can be read as *Shtaran* (their bill in feminine form), and 10 include the adverb *Sheterem* which can be read as *Shtaram* (their bill in masculine form).

To evaluate the quality of search engines more systematically, we used a set of 22 queries, each one is associated with a relevant list of all relevant documents

String	String-matching	$\epsilon = 0.5$	$\epsilon = 0.1$	$\epsilon = 0.0$
<i>Mishtara</i> (police)	17	17	17	17
<i>Hamishtara</i> (the police), <i>Mehamishtara</i> , <i>Lamishtara</i> , <i>Bamishtara</i> etc.	-	100	100	100
<i>Mishteret</i> (constructed form)	-	50	50	50
<i>Mishtarot</i> (plural form)	-	1	1	1
<i>Mishtarti</i> (adjective)	-	6	6	6
<i>Mishtar</i> (regime)	-	-	-	25
<i>Sheterem</i> (before) (also can be read as <i>Shtaram</i> , their bill)	-	-	-	10
<i>Stern</i> (Private Name) (also can be read as <i>Shtaran</i> , their bill)	-	-	-	4

Table 3. The number of documents retrieved by the different search engines for the query *Mishtara*.

manually tagged by a human judge⁷. Table 4 shows the recall/precision of the Search Engine averaged over the 22 queries. The upper row shows the result of a string matching search where no analysis was done while creating the index and submitting the queries. The three lower rows show the search results of Hemed with the different threshold parameters. String matching search achieves high precision but very low recall. In contrast, Hemed versions search for the base forms and return all appearances of their inflections.

	Recall	Precision
String Matching	0.559	0.364
Hemed $\epsilon = 0$	0.836	0.248
Hemed $\epsilon = 0.1$	0.797	0.263
Hemed $\epsilon = 0.5$	0.758	0.277

Table 4. The recall/precision of different search engines averaged over 22 queries.

The search engine integrated with the Hemed versions achieves much better results but with lower precision. Using a threshold of 0 improves the recall of the search significantly since all analyses are kept inside the index. This is also

⁷ Unfortunately, there is no forum like TREC, and no test collections like those on Ed Fox’s Virginia disk for Hebrew search systems. We therefore had to “craft” our own Hebrew test collection, and the results should only be seen as encouraging until a more formal forum is established.

the reason for the deterioration in the search precision. Enlarging the threshold parameter improves the precision of the search with a little decrease in recall.

Figure 5 shows the Recall/Precision curves of the Hebrew Search Engines, averaged over the 22 queries. The results strengthen our claim that string matching search fails for Hebrew. Its performance is reflected by the recall/precision curve which is much lower than the curves of the search engines integrated with Hemed. The graph also highlights the role of the threshold parameter for Hemed procedure. It allows the user to control the tradeoff between recall and precision by changing the threshold according to required results.

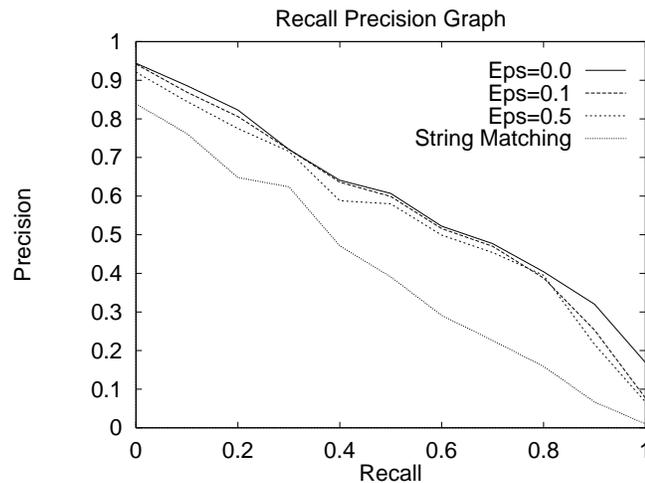


Fig. 5. The recall/precision graph of three versions of Hemed, and a string matching search engine, averaged over 22 queries.

5 Summary

In this work we describe a new approach for linguistic indexing of Hebrew text files. We showed that linguistic indexing can significantly improve the recall rate of a Hebrew search engine with low reduction in precision rate, compared to precise indexing used by a string-matching search engine. We introduced the idea of adding controllable morphological disambiguation while indexing and while analyzing users' queries. The level of disambiguation can be set dynamically to control the recall and precision of the search results.

We described Hemed, a statistical morphological disambiguator for Hebrew. Hemed exploits statistical data gathered from a large Hebrew corpus to reduce the level of morphological ambiguity. Hemed has been integrated into *BabaGuru*, a research prototype of a Hebrew search engine, and is now being integrated into

Intelligent Miner for Text (IM4T) [5], a multi-lingual search engine, to enable linguistic indexing for Hebrew.

The context-free disambiguator described in this work analyzes words without concerning their context. In many cases the “true” morphological analysis of a given word can not be determined without taking its context into consideration. Context-sensitive disambiguation function might have better results for indexing. The question how to implement such a context-sensitive disambiguator for Hebrew remains open for further research.

Acknowledgments

We are grateful to thank Victoria Skoblikov for implementing *BabaGuru*, the Hebrew search engine used in the experiments described in this work. We would also like to thank Moshe Levinger for stimulating discussions and useful suggestions concerning this work.

References

1. Esther Benture, Aviela Angle, and Danit Segev. Computerized analysis of Hebrew words. In *Hebrew Computerized Linguistics*. Isreal Ministry of Science and Technology, 1992.
2. Yaacov Choueka, A. S. Fraenkel, S. T. Klein, and E. Segal. Improved techniques for processing queries in full-text systems. In *Proceedings of ACM Conference on research and development in Information Retrieval*, pages 306 – 315. ACM press, NY, June 1987.
3. Yaacov Choueka and S. Lusignan. Disambiguation by short contexts. *Computers and Humanities*, 19:147 – 157, 1985.
4. J. L. Fagan. The effectiveness of a nonsyntactic approach to automatic phrase indexing for document retrieval. *JASIS*, 40(2):115–132, 1989.
5. Intelligent Miner for Text. <http://www.software.ibm.com/data/iminer/fortext>. IBM Corporation, 1998.
6. Moshe Levinger. Morphological disambiguation, Master thesis (in Hebrew), Technion, Israel institute of technology. 1992.
7. Moshe Levinger, Uzzi Ornan, and Alon Itai. Learning morpho-lexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics*, 2(3):383 – 404, 1993.
8. Yoelle Maarek and F. Smadja. Full text indexing based on lexical relations, an application: Software libraries. In N. Belkin and C. van Rijsbergen, editors, *Proceedings of SIGIR89*, pages 198 – 206. Cambridge MA, ACM press, 1989.
9. Uzzi Ornan. Phonemic script: A central vehicle for processing NL – the case of hebrew, IBM, scientific center, haifa. Technical Report 88-181, 1986.
10. Uzzi Ornan. Theoretical gemination in Israeli Hebrew. In Allan S. Kaye and Otto Harrassowitz, editors, *Semitic Studies in Honor of Wolf Leslau*, pages 1158 – 1168. 1991.
11. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130 – 137, 1980.
12. Responsa project. <http://www.biu.ac.il/JH/Responsa/>. Bar-Ilan univrtsity, 1998.

13. G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. Computer Series. McGraw-Hill, New York, 1983.
14. Ellen M. Voorhees and Donna Harman. Overview of the Sixth Text REtrival conference (TREC-6). In *Proceedings of the Sixth Text REtrieval Conference*. National Institute of Standards and Technology, August 1997.