Practical Hebrew search

Itamar Syn-Hershko
@synhershko
http://code972.com

/Me

- Itamar Syn-Hershko
- Hibernating Rhinos
  - Data Access champs
  - ORM Profilers
  - RavenDB
- Lucene, CLucene
- HebMorph
- More @ http://code972.com

Dealing with data explosion

- Manual tagging is too much work
- Scanning texts takes too long
- Inverted index: faster, flexible, relevance
- Measuring TR engine: precision, recall
- There is no perfect search engine: language, users, corpora dependent

Search 101

- The indexing process: given a corpus, produce an inverted index
- Querying: based on a user question, build the best query possible that is understood by the search engine
- Performing the actual search: read the index (in a method dictated by the query), and make relevance calculations as fast as possible

Search 101: the Inverted Index

1	The old night keeper keeps the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep
4	Where the old night keeper never did sleep.
5	The night keeper keeps the keep in the night
6	And keeps in the dark and sleeps in the light.

6 documents to index

The index:

Dictionary and posting lists

Term	Positions
and	<6>
big	<2> <3>
dark	<6>
did	<4>
gown	<2>
had	<3>
house	<2> <3>
in	<1> <2> <3> <5> <6>
keep	<1> <3> <5>
keeper	<1> <4> <5>
keeps	<1> <5> <6>
light	<6>
never	<4>
night	<1> <4> <5>
old	<1> <2> <3> <4>
sleep	<4>
sleeps	<6>
the	<1> <2> <3> <4> <5> <6>
town	<1> <3>
where	<4>

Example from:
Justin Zobel , Alistair Moffat,
Inverted files for text search engines,
ACM Computing Surveys (CSUR)
v.38 n.2, p.6-es, 2006

Search 101: the Inverted Index

User queries for “Keeper”

1	The old night **keeper** keeps the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep
4	Where the old night **keeper** never did sleep.
5	The night **keeper** keeps the keep in the night
6	And keeps in the dark and sleeps in the light.

6 documents to index

The index:

Dictionary and posting lists

Term	Positions
and	<6>
big	<2> <3>
dark	<6>
did	<4>
gown	<2>
had	<3>
house	<2> <3>
in	<1> <2> <3> <5> <6>
keep	<1> <3> <5>
keeper	<1> <4> <5>
keeps	<1> <5> <6>
light	<6>
never	<4>
night	<1> <4> <5>
old	<1> <2> <3> <4>
sleep	<4>
sleeps	<6>
the	<1> <2> <3> <4> <5> <6>
town	<1> <3>
where	<4>

Search 101: Term Normalization

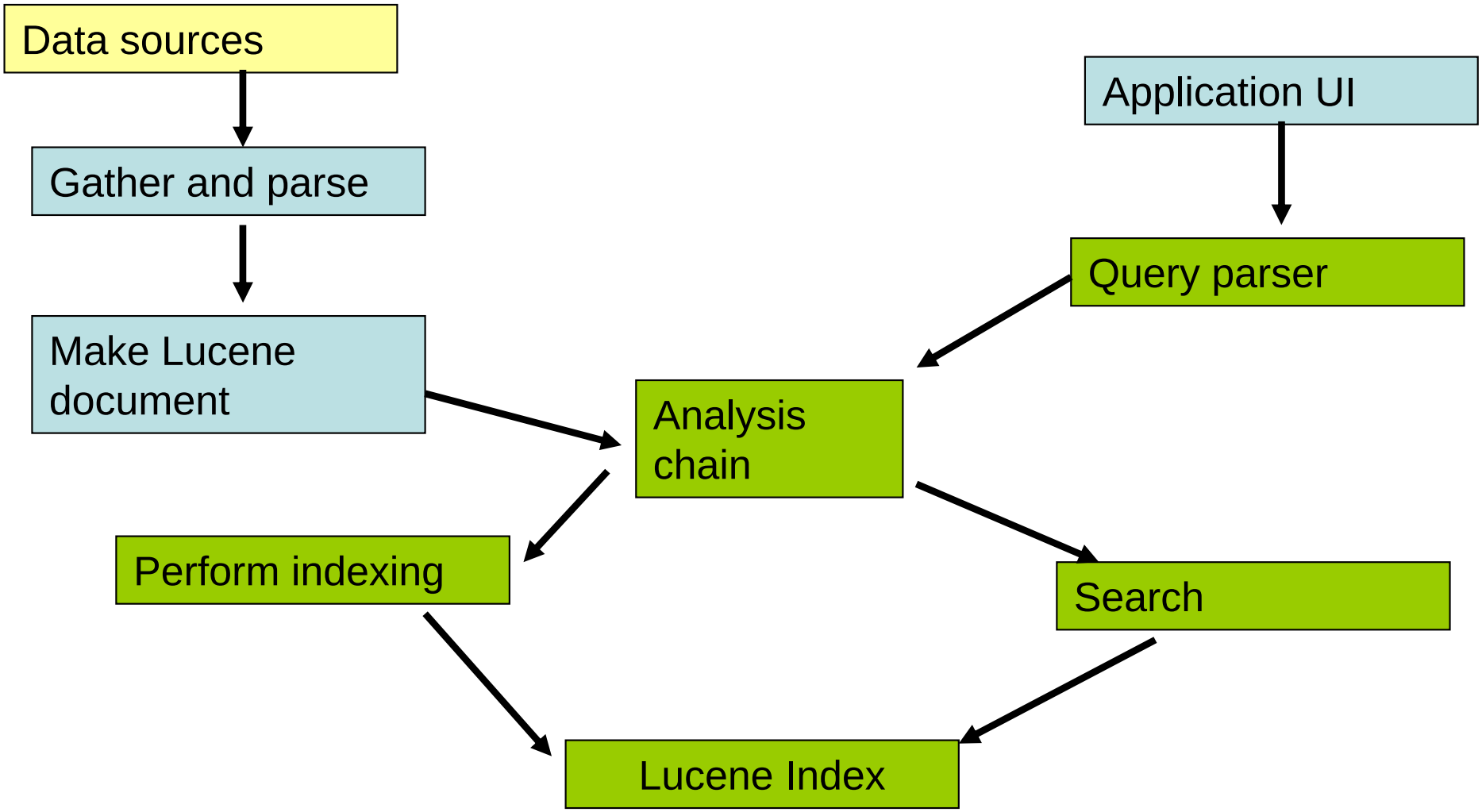
- Stop words (grey)
- Stemming
  - Porter stemmer
  - s-stemmer

Term	Positions
and	<6>
big	<2> <3>
dark	<6>
did	<4>
gown	<2>
had	<3>
house	<2> <3>
in	<1> <2> <3> <5> <6>
keep	<1> <3> <5>
keeper	<1> <4> <5>
keeps	<1> <5> <6>
light	<6>
never	<4>
night	<1> <4> <5>
old	<1> <2> <3> <4>
sleep	<4>
sleeps	<6>
the	<1> <2> <3> <4> <5> <6>
town	<1> <3>
where	<4>

Meet Lucene

- Mature, state of the art IR library
- Provides API for adding indexing and search capabilities to applications
- Written in Java, with ports also to .NET, C++
- Fast, efficient, constantly evolving
- Many extension points, Contribs
- Document has Fields, each Field holds Terms
- The analysis chain

Meet Lucene


Data sources
Gather and parse
Make Lucene document
Application UI
Query parser
Analysis chain
Perform indexing
Search
Lucene Index

Using Lucene: Indexing

```
1  import org.apache.lucene.analysis.SimpleAnalyzer;
2  import org.apache.lucene.document.Document;
3  import org.apache.lucene.document.Field;
4  import org.apache.lucene.index.IndexWriter;
5  import org.apache.lucene.store.FSDirectory;
6
7  // ...
8
9          IndexWriter indexWriter = new IndexWriter(
10                 FSDirectory.open(path), // or RAMDirectory, or other Dir
11                 new SimpleAnalyzer(), // Analyzer of choice
12                 true, // create if does not exist
13                 IndexWriter.MaxFieldLength.LIMITED);
14
15         // Add a new index document
16         Document doc = new Document();
17         doc.add(new Field("contents", new FileReader(textFile)));
18         doc.add(new Field("filename", textFile.getCanonicalPath(),
19            Field.Store.YES, Field.Index.ANALYZED));
20
21         indexWriter.addDocument(doc); // commit the doc to the index
22
23         indexWriter.optimize();
24         indexWriter.close();
```

Using Lucene: Search

```java
import org.apache.lucene.analysis.SimpleAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.queryParser.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.util.Version;

// ...

        Directory directory = FSDirectory.open(indexDir);
        IndexSearcher searcher = new IndexSearcher(directory);

        // Analyzer used for parsing the query has to match the one used for indexing!
        QueryParser parser = new QueryParser(Version.LUCENE_30, "contents", new SimpleAnalyzer());
        Query query = parser.parse("+my +file");

        TopDocs topDocs = searcher.search(query, maxHits);

        ScoreDoc[] hits = topDocs.scoreDocs;
        for (int i = 0; i < hits.length; i++) {
            int docId = hits[i].doc;
            Document d = searcher.doc(docId);
            System.out.println(d.get("filename")); // only works on STORED fields
        }
        System.out.println(hits.length + " results found");
```

Using Lucene: Analyzers

The quick brown fox jumped over the lazy dogs, bob@hotmail.com 123432.

StandardAnalyzer:

[quick] [brown] [fox] [jumped] [over] [lazy] [dog] [bob@hotmail.com] [123432]

StopAnalyzer:

[quick] [brown] [fox] [jumped] [over] [lazy] [dogs] [bob] [hotmail] [com]

SimpleAnalyzer:

[the] [quick] [brown] [fox] [jumped] [over] [the] [lazy] [dogs] [bob] [hotmail] [com]

WhitespaceAnalyzer:

[The] [quick] [brown] [fox] [jumped] [over] [the] [lazy] [dogs,] [bob@hotmail.com] [123432.]

KeywordAnalyzer:

[The quick brown fox jumped over the lazy dogs, bob@hotmail.com 123432.]

Using Lucene: There's a lot more

- Highlighting and extraction of best fragments
- MoreLikeThis
- "Did you mean … ?"
- Faceted search
- Similarity (BM25)
- Real-time search
- Cloud Directory implementations
- And much more…

Challenges with Hebrew IR

Particles and inverted index

Term
איש
אנשים
ביקשתי
בלבן
דובים
הדוב
החי
הלבן
הסתיו
לטייל
לשלושה
מאיש
...
קראתי
שלושה
שלישיות
שלש

1	שלושה דובים יצאו לטייל
2	קראתי לשלושה אנשים לבוא ולעזור
3	שלושה משפטים עם שלישיות זה קצת מעצבן להמציא
4	הדוב הלבן, החי בצפון כדור הארץ משמין עם בוא הסתיו
5	ביקשתי ממנו לצבוע את קירות בית המשפט בלבן
6	קיבלנו מאיש מסתורי שלש חוברות מתנה

Challenges with Hebrew IR

- Tokens ambiguity with niqqud-less spelling, which is the most common

English: Look, Luke; Wine, Whine; Stack, Stuck.

Hebrew: שָׁנִי, שֵׁנִי, שֶׁנֵי, שֹׁנִי, שְׁנִי

Niqqud-less spelling: שני, שני, שני, שני, שני…

Challenges with Hebrew IR

- Hebrew word uses particles for context
- Without removing suffixes, relevant words might be skipped (for example: חבלה)
- Without removing prefixes, relevant words will not be looked up at all
- Ambiguity makes affixes removal impossible in many cases

בית -> הבית, בבית, שבבית, לבית, והבית...
הרכבת -> רותי פספסה את הרכבת
הרכבת המוצר מסובכת להפליא
כלבי -> ?
שבתו –> ?

Challenges with Hebrew IR

- No spelling rules:
  - כתיב חסר / מלא ("אימא")
  - Loanwords and names

דוגמה או דוגמא?
אחשורוש או אחשוורוש?
שבדיה או שוודיה?
טורקיה או תורכיה?
פריס או פריז? או אולי פאריז?

Challenges with Hebrew IR

- Stop words ambiguity

אשר, כדי, אף...

- Stop words as collations

על ידי, אי פעם, אף על פי, שום דבר...

- Collations where a meaning of a single word is changed

פי התהום

Challenges with Hebrew IR

- Tokenization:
  - Hebrew acronyms use double-quotes character, which is usually considered as punctuation character by most tokenizers
  - Same with Geresh, which is used for abbrevations
  - Geresh is also used for חצ"ץ ג"ז
  - … and ambiguity again: אינצ'

Ways of resolution

- Deciding on an “indexing unit” is the cornerstone of any good performing search engine
- For Hebrew we have:
  - The original term (and possibly using wildcards?)
  - Hebrew triliteral root
  - Lemma (דלתותינו ← דלת)
  - Psuedo-lemma, Stem
  - Other non word-based approaches (n-grams)?
- Considerations

Hebrew NLP methods

- In order to extract a correct lemma, the word has to be evaluated within its original context
- Dictionary based or algorithmic
- Both require a lot of work, and are still prone to errors
- Even with the most advanced tools, ambiguity will remain:

"המראה של מטוסים ריקים [...]"
"ראש הממשלה בבון"
"ללכת לנגב"

Food for thought

- Researches have shown 4-grams and light stemmers ("light-10") to work better than morphologic lemmatizers for Arabic IR
- Apparently, good relevance can be achieved without 'knowing' the language
- Search: Computers vs Humans
- Lemmatization and disambiguation processes do make mistakes
- Contextual processing can fail for short queries, producing incorrect searches
- Currently there is no way of knowing if common Web search engines really produce quality results for your Hebrew searches!

HebMorph

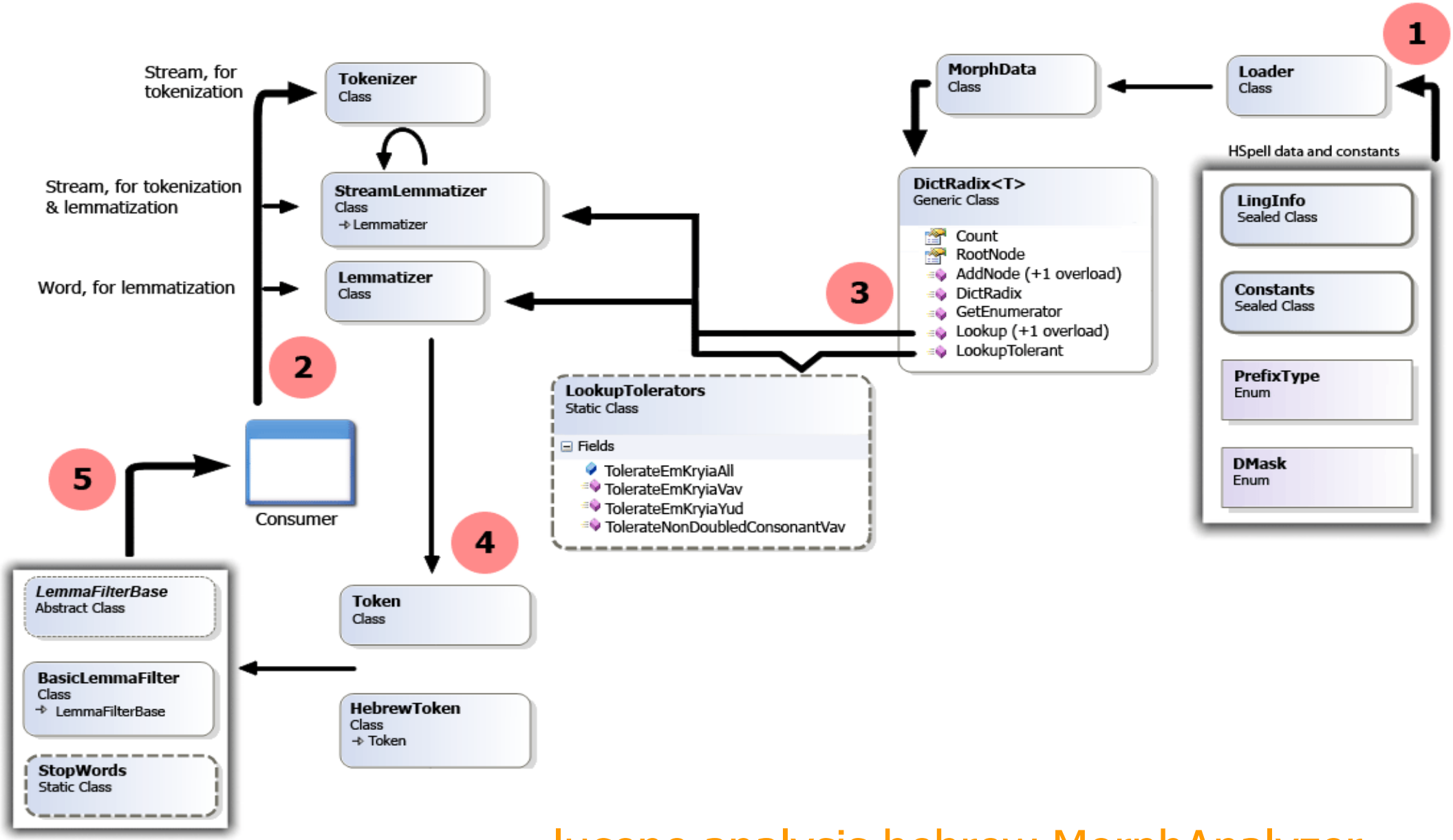
… is a free, open-source effort for making Hebrew properly searchable by various IR software libraries, while maintaining decent recall, precision and relevance in retrievals.

- 2 goals
- Testing and evaluation are done on top of Lucene
- Available in .NET and Java, C++ underway
- MorphAnalyzer, Hebrew.SimpleAnalyzer (+ duality)
- OpenRelevance

What do we have now?

- MorphAnalyzer
  - hspell is loaded into a Radix
  - Hebrew Tokenizer
  - Token is looked up in the Radix
  - If not found, tolerating lookup is tried
  - Token filters (score, stop words)
- Hebrew.SimpleAnalyzer
  - Hebrew Tokenizer
  - Optional duality with MorphAnalyzer (suffix)
- QueryParser

Practical Hebrew Search



lucene.analysis.hebrew.MorphAnalyzer

Using HebMorph

- Keep MorphAnalyzer around, don't recreate
- Take advantage of boosts, LemmaFilters, BinaryCoordSimilarity

```
// Create a MorphAnalyzer instance
var morphAnalyzer = new MorphAnalyzer(HSpellDataFilesPath);

// Plug in a standard HebMorph lemma filter, to remove low scored results (tolerated lemmas)
var lemmaFilter = new HebMorph.LemmaFilters.ChainedLemmaFilter();
lemmaFilter.Filters.AddLast(new HebMorph.LemmaFilters.BasicLemmaFilter());
morphAnalyzer.lemmaFilter = lemmaFilter;

// To enable duality search with Hebrew.SimpleAnalyzer, set only before indexing
// More about this here: http://www.code972.com/blog/2010/07/more-flexible-hebrew-indexing-hebmorph/
morphAnalyzer.alwaysSaveMarkedOriginal = true; // to allow for non-morphologic searches too

// Parse queries with a Hebrew compatible QueryParser
var qp = new HebrewQueryParser(Lucene.Net.Util.Version.LUCENE_29, "field", morphAnalyzer);
indexSearcher.Search(qp.Parse("query"), tsdc);
```

On-Line Demo

Hebrew Wikipedia searchable by HebMorph

Try it live yourself:
http://hebmorph.code972.com

Full source available from
http://github.com/synhershko/HebMorph.CorpusSearcher
(AGPLv3)

HebMorph: The road ahead

- Hebrew judgments for OpenRelevance with Orev
- Comparing various possible approaches to Hebrew IR (n-grams, Viterbi, …)
- Tokenizer improvements
- MorphAnalyzer:
  - hspell improvements (coverage, lemma probabilities, prefix probabilities)
  - Better Toleration mechanism
  - Smarter OOV handling
  - Better stop words handling
- Other uses (NLP, OCR, you name it)

Orev – OpenRelevance Viewer

- Work in progress
- Relevance judgments
- Corpus, Topic, Judgment
- Muli-lingual, multi-corpus application for corpus-topic judging

Questions ?

Our mailing list: https://lists.sourceforge.net/lists/listinfo/hebmorph-thinktank

Code repository (AGPLv3):
http://github.com/synhershko/HebMorph

Activity updates and more information:
http://hebmorph.code972.com/