

Relevance Collection #1: Persian Hamshahri.

This test attempts to measure the difference in scoring behavior for a collection where stopwords make all the difference: Persian text.

Due to some unique properties of this collection, the components of PersianAnalyzer are a no-op, only the stopwords list has an effect on relevance. These properties can be verified by using the UnicodeCCount or similar utility on the corpus: all characters are from windows-1256 (arabic) and the text is of high quality.

The best part of this test is that it can be reproduced entirely using open source software: the collection, queries, and judgements are integrated into the openrelevance project, and the lucene benchmarking suite can be used to run the evaluation.

This is a large collection with 160,000 documents. The judgements were formed by pooling.

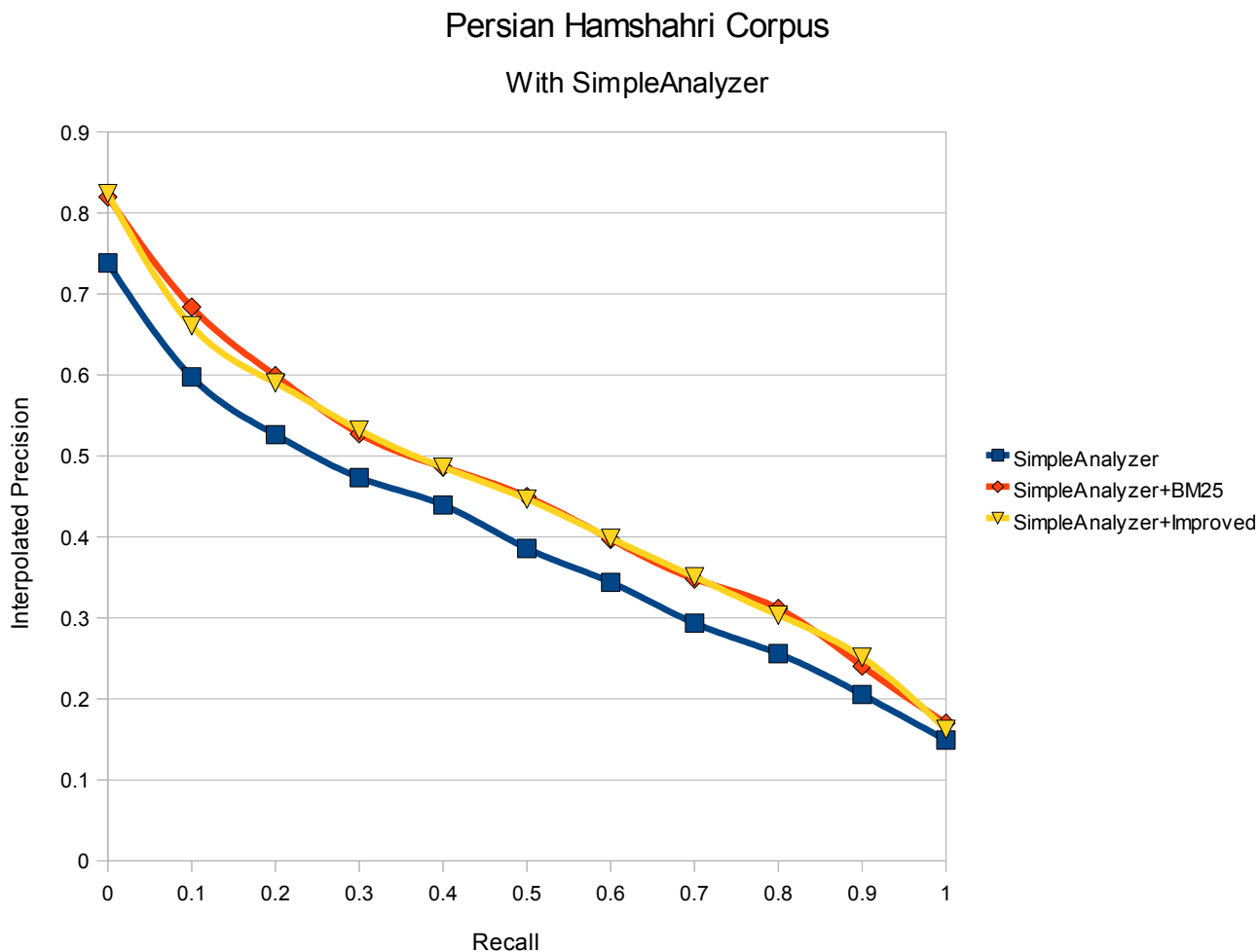
## Configuration A: Hamshahri Persian corpus with SimpleAnalyzer

In this test we compare the baseline scoring function alongside the improved scoring function and BM25, using a very simplistic language-agnostic analyzer.

For this language, the baseline with SimpleAnalyzer suffers from two problems:

1. The corpus has a wide variety of document lengths
2. SimpleAnalyzer has no knowledge of persian stopwords, especially persian morphemes that are written with additional space such as the plural **لها**

The improved scoring function performs similar to BM25 in this configuration. In this case it is a significant improvement over the default scoring algorithm.

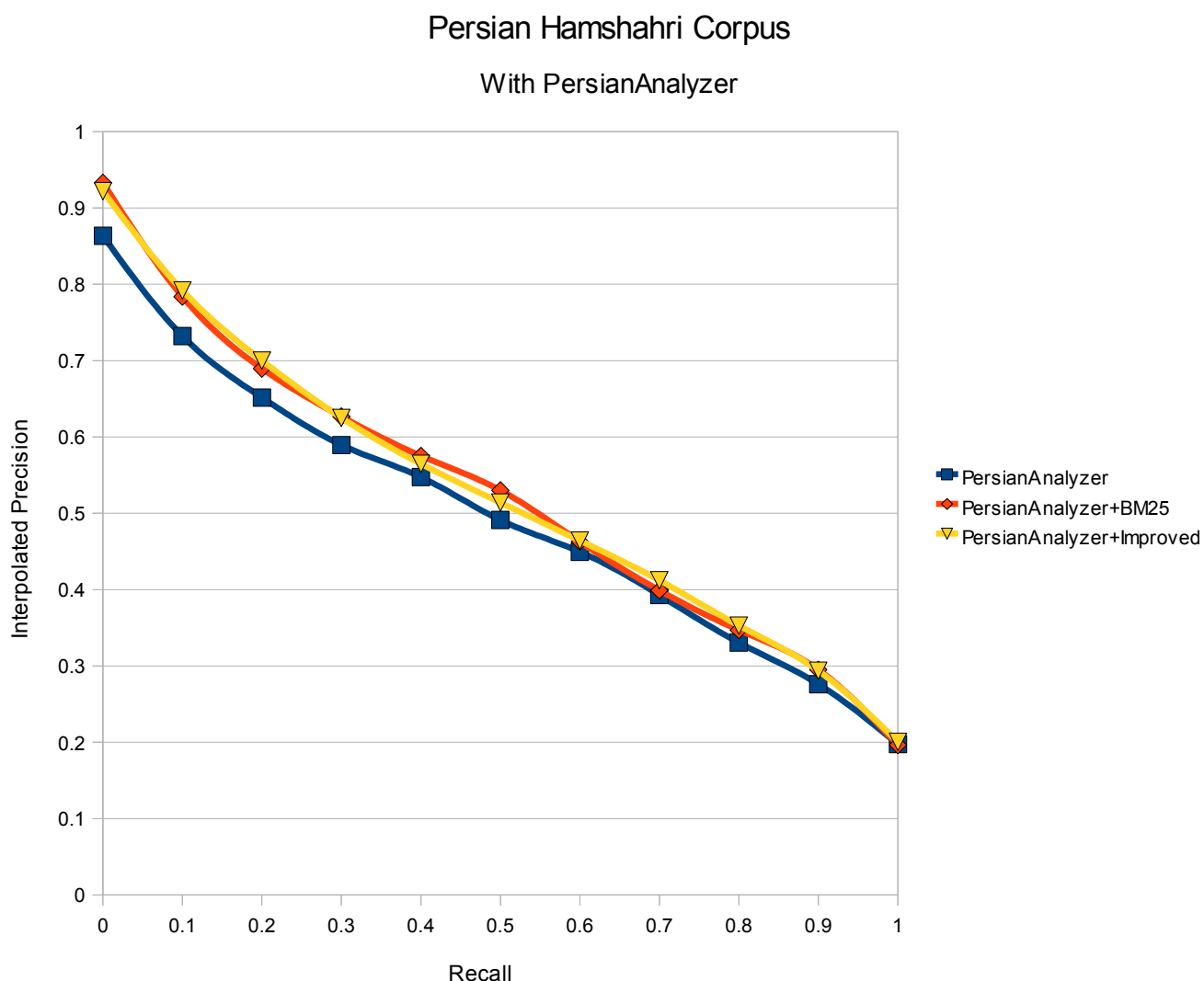


## Configuration B: Hamshahri Persian corpus with PersianAnalyzer

In this test we compare the baseline scoring function alongside the improved scoring function and BM25, using an analyzer tuned for persian text.

With this corpus, there are few unicode issues typically associated with Persian, and the text is of high quality, so there are few concatenated words: the analyzer is geared towards good quality persian text and this corpus is an excellent match for the analysis.

The improved scoring function again performs similar to BM25 in this configuration, still providing improvements over Lucene's default scoring algorithm even with good analysis.



## Relevance Collection #2: Hindi Jagran Corpus

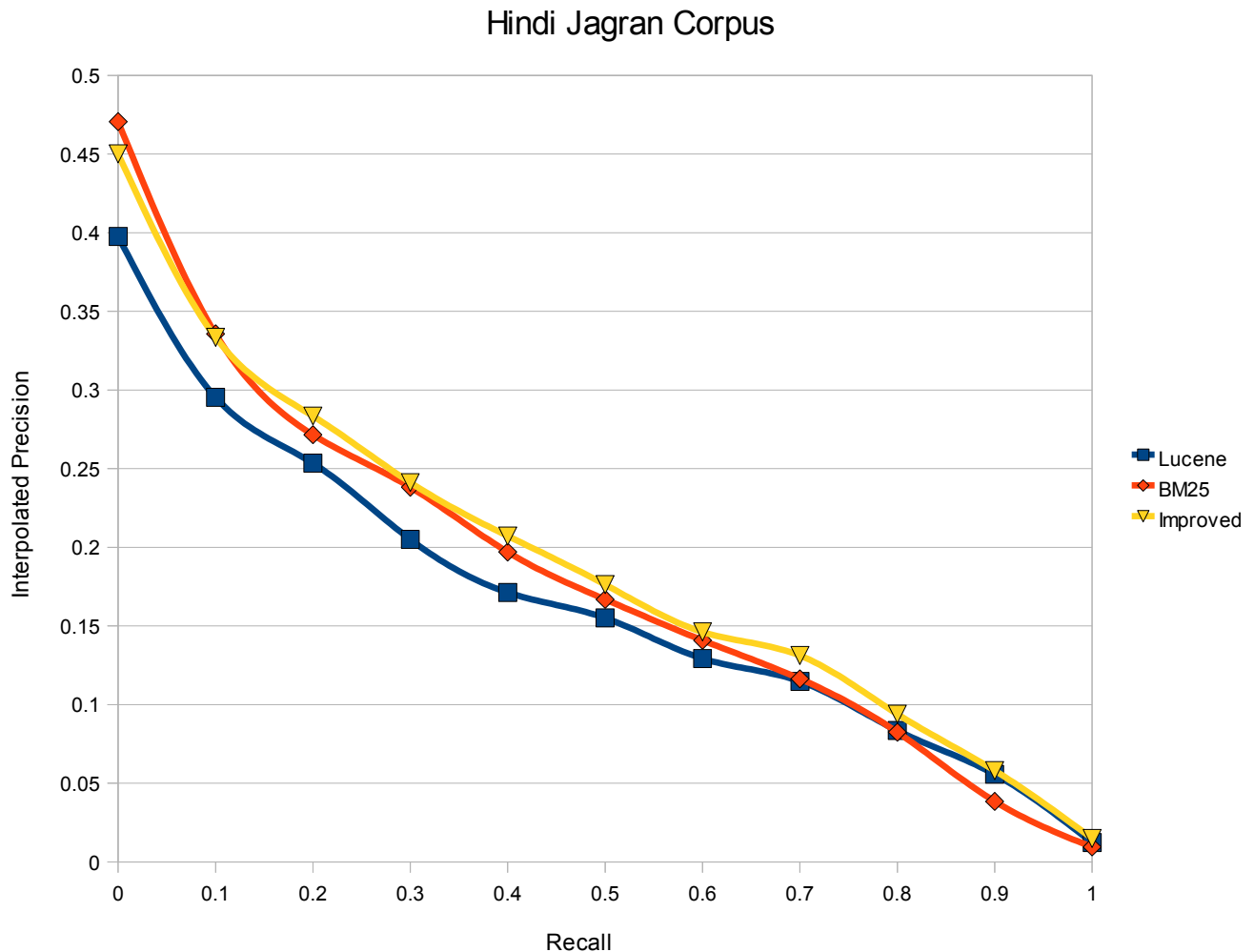
This corpus has approximately 95,000 hindi documents with pooled judgements.

In this test we compare the baseline scoring function alongside the improved scoring function and BM25, using a very simple “IndicAnalyzer”:

```
@Override
protected boolean isTokenChar(char c) {
    return super.isTokenChar(c)
        || Character.getType(c) == Character.NON_SPACING_MARK
        || Character.getType(c) == Character.COMBINING_SPACING_MARK;
}
```

followed by LowerCasing.

Technically, the new scoring function outperforms BM25, but I didn't bother to see if the changes against BM25 are significantly different, either way its comparable, and an improvement over the baseline.



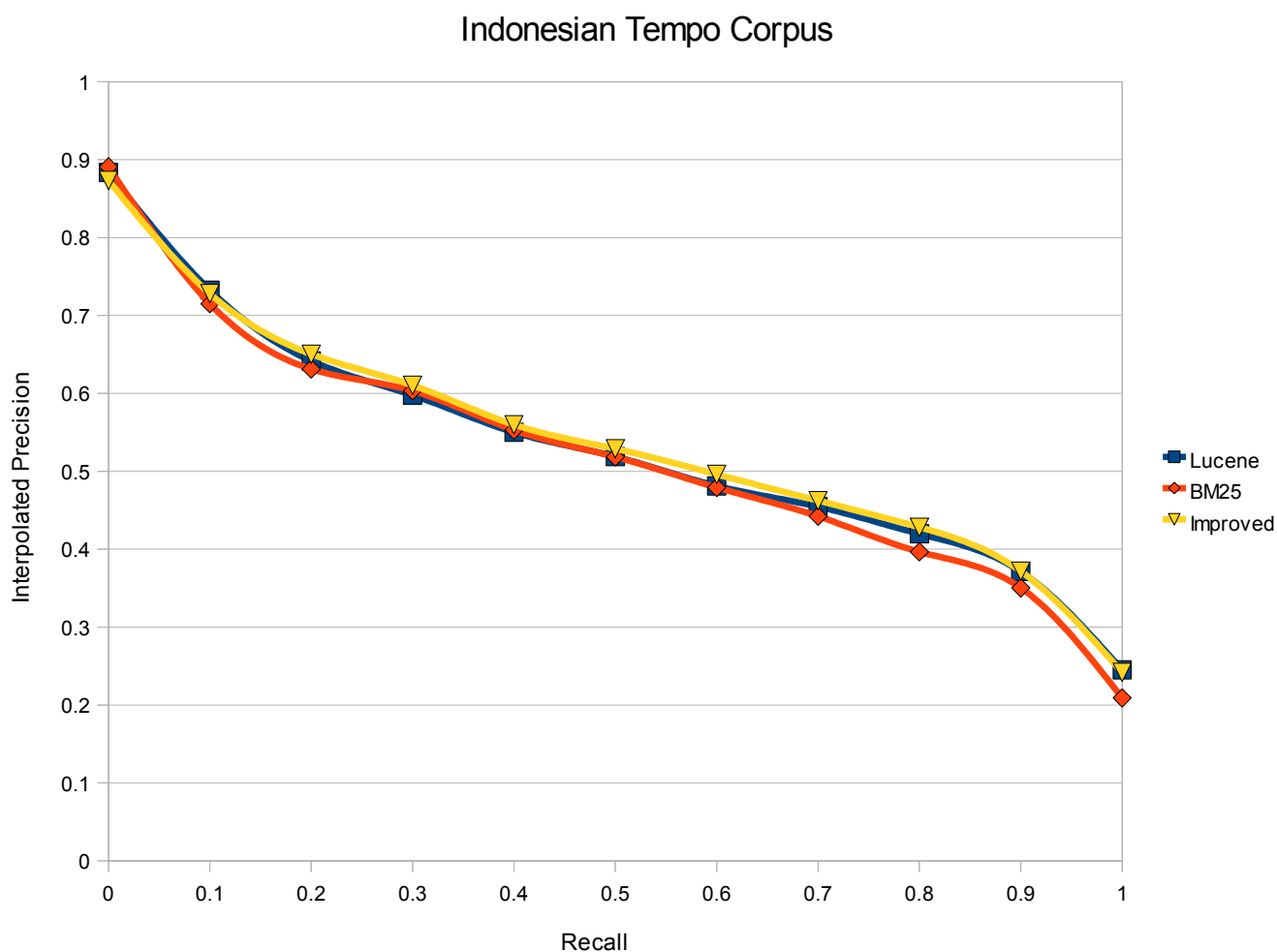
### Relevance collection #3: Indonesian Tempo Corpus: worst case

This corpus has approximately 22,000 documents, all judged manually.

This test is that it can be reproduced entirely using open source software: the collection, queries, and judgements are integrated into the openrelevance project, and the lucene benchmarking suite can be used to run the evaluation.

BM25 doesn't give any gain for this collection over the baseline lucene, perhaps because of the some of the properties of the collection (proper names, etc). The goal here is to show that in this case the new formula is not any worse.

For this collection, none of the changes are statistically significant over the baseline, technically the improved formula might look slightly better but this does not matter.



#### Relevance Collection #4: English Telegraph Corpus

This corpus has approximately 125,000 documents, with pooled judgements.

BM25 doesn't do well with this corpus, but the new scoring algorithm always does well (whether or not the differences are significant, who knows)

For reference, here are the MAP values:

StandardAnalyzer

Default Scoring: 0.3837

BM25 Scoring: 0.3580

Improved Scoring: 0.3994

StandardAnalyzer + Porter

Default Scoring: 0.4333

BM25 Scoring: 0.4131

Improved Scoring: 0.4515

StandardAnalyzer + Porter + MoreLikeThis (top 5 docs)

Default Scoring: 0.5234

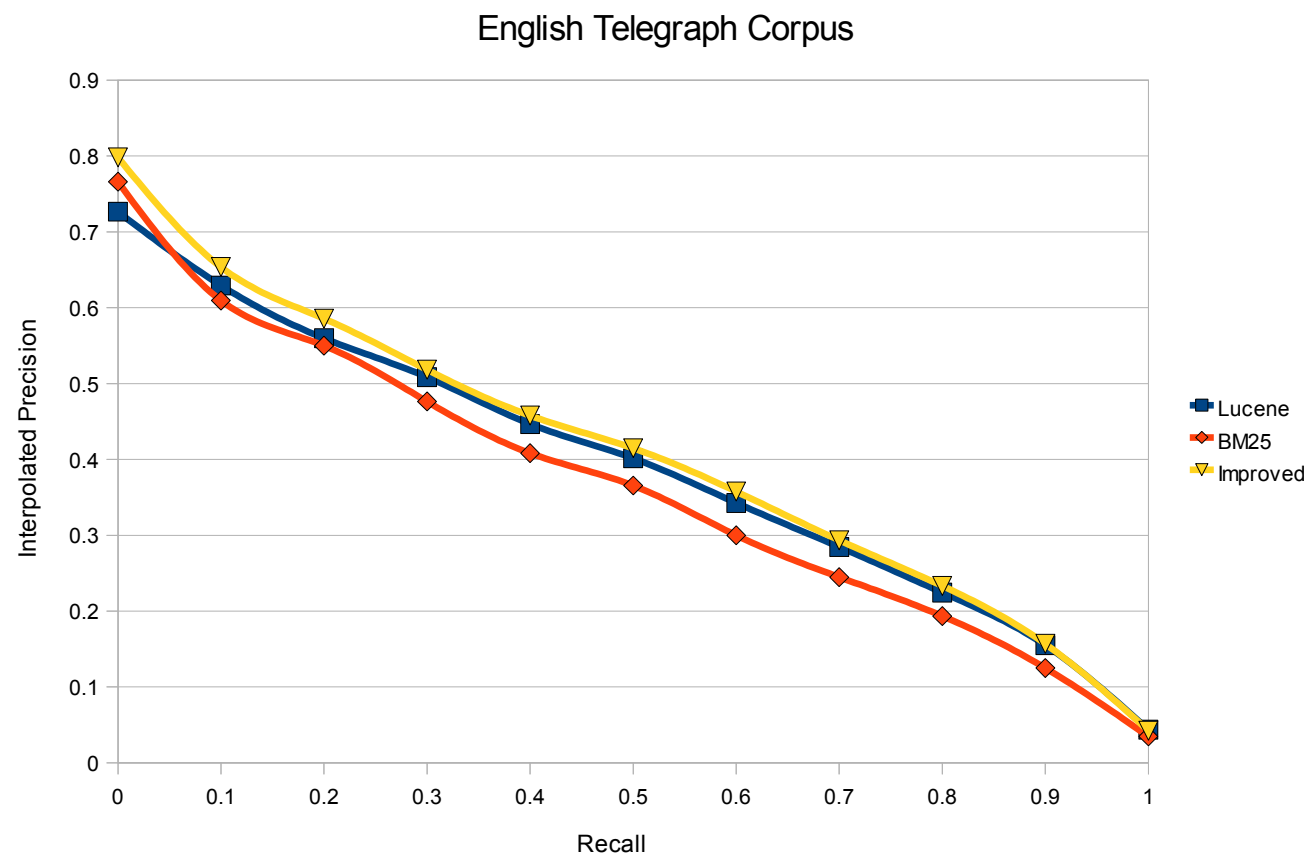
BM25 Scoring: 0.5087

Improved Scoring: 0.5474

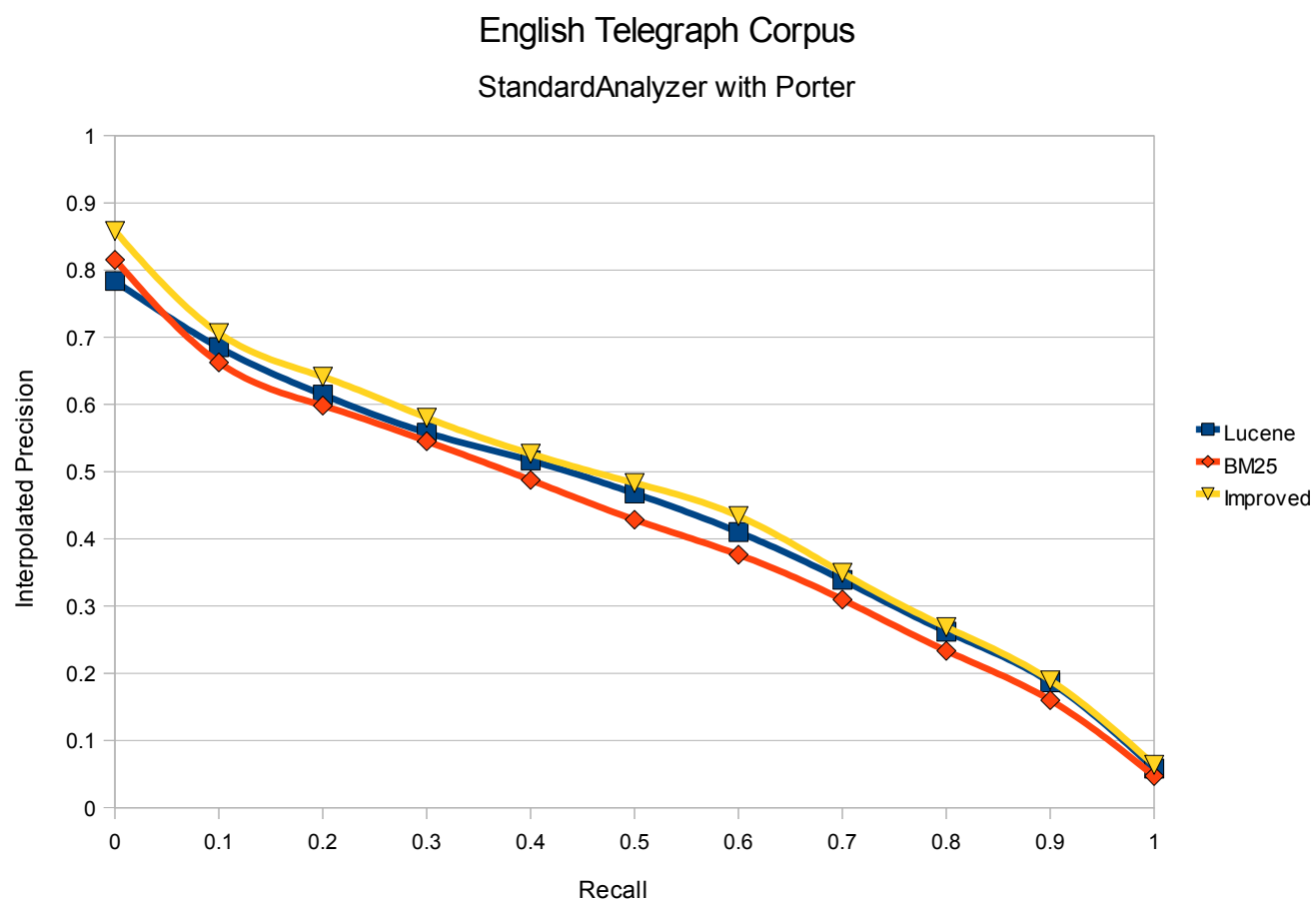
Note that 0.5572 was the highest performing MAP on this corpus (Microsoft Research) in FIRE 2008

Configuration A: StandardAnalyzer

Good old StandardAnalyzer



Configuration B: StandardAnalyzer with the PorterStemFilter from lucene core.





Configuration C: StandardAnalyzer with the PorterStemFilter from lucene core.

The Query was Or'ed with the resulting query from MoreLikeThis.like(int) for the top 5 documents for blind expansion.

A larger stopwords list from SMART was used for better MoreLikeThis results, other than this all defaults were used.

