Identifying the most suitable stemmer for the CHiC multilingual ad-hoc task

Thomas Wilhelm-Stein, Benjamin Schürer, and Maximilian Eibl

Technische Universität Chemnitz, 09107 Chemnitz, Germany,
{wilt, schben, eibl}@hrz.tu-chemnitz.de

Abstract. Because the 2013 Cultural Heritage in CLEF (CHiC) lab focused on multilingual retrieval, our goals were the integration of Apache Solr in our Xtrieval framework and the evaluation of different stemmers available for most of the relevant languages. As there were thirteen languages to cover, we tried to find a generic stemmer which works with all languages. We experimented with four setups, where one setup was without any stemmer, two setups used mainly rule-based stemmers and the last setup used a dictionary-based stemmer. For the dictionary-based stemmer we employed the HunSpell stemmer, which works with the same dictionaries as OpenOffice.

Keywords: stemmer, evaluation, dictionary-based stemmer, rule-based stemmer, cultural heritage

1 Introduction

In 2013 the Cultural Heritage in CLEF (CHiC) lab focused on multilingual retrieval, i.e. searching over all available languages. For this task topics and relevance assessments were provided for a total of thirteen languages. This set allowed us to perform a large scale evaluation of stemming algorithms for all available languages.

Our goals for this year were the integration of Apache Solr[1] in our Xtrieval framework [1] and the evaluation of the different stemmers available within Solr for a broad use on multilingual corpora. Hence we focused on the multilanguage ad-hoc task.

2 System overview and setup

As usual we used our Xtrieval framework to carry out all our experiments. But this year we added a new retrieval engine: Apache Solr. It is a very popular open source enterprise search, [2] which is built on top of Apache Lucene[2], and provides several web service interfaces to conduct the different tasks necessary to perform large scale searches.

[1] http://lucene.apache.org/solr
[2] http://lucene.apache.org/java

Although we have already been using Apache Lucene for our experiments since 2006 [3], this new approach has changed the way we have to configure them. A considerable amount of configuration is now done in Solr using various XML files. Within these configuration files one can define fields with associated types where the processing steps are declared.

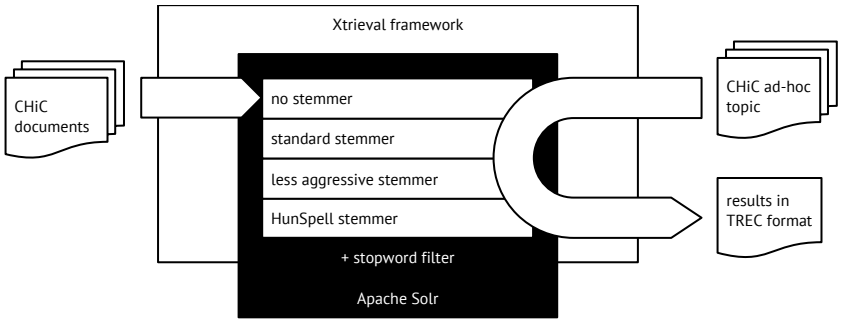

Fig. 1. Xtrieval framework using Apache Solr for our stemming experiments

As shown in figure 1, the Xtrieval framework was still an essential part of our experiments. We used it to read and parse the document collection and fed it into the Solr web service. At this point there are still some pre-processing steps left to the Xtrieval framework, which cannot be configured in Solr.

After the indexing, we used our framework to transform the topics into search queries for the Solr web service and gathered the results in the TREC format in order to submit them.

This year we focused on the different stemming approaches, which are offered by Solr. As there were thirteen languages available, we tried to find a generic stemmer which covers all these languages. We experimented with four settings: one was without any stemmer, two setups used mainly rule-based stemmers and the last setup used a dictionary-based stemmer. For the dictionary-based stemmer we employed the HunSpell stemmer [4], which works with the same dictionaries as OpenOffice and LibreOffice. Because of the open source nature of these applications there are large numbers of dictionaries available for almost every language.

Finally, we set up these stemming experiments:

- No stemmer
- The standard stemmer configured in Solr: In most cases this is the snowball stemmer or another rule-based stemmer. Where no other stemming algorithm was available the HunSpell stemmer was used.
- A less aggressive stemmer: This was used instead of the standard stemmer for every language where a less aggressive stemmer was available.
- The HunSpell stemmer: Only languages with HunSpell dictionaries which had a reasonable performance used this stemmer. There were performance

problems with complex languages, like for example French, where probably many connections exist between dictionary entries. The bad performance for this language sometimes resulted in a processing time 1000 times longer than with the standard stemmer. Therefore we used no stemmer for these languages.

Furthermore we applied a language detection to determine fields with wrongly labeled languages and assigned the detected language. We also removed stop words for each language. Each language was indexed into the same index, but using separate fields. When processing the topics for each available language a query was issued to the whole index, i.e. every language in the index.

The mapping of the XML data to the fields of the Solr/Lucene index was the same we used for our experiments last year. [5]

3 Results

Table 1 shows the results of our experiments. We compared four different measures: mean average precision (map), geometric mean average precision (gmap), binary preference (bpref), and precision at R (r-precision). Each line represents a stemmer configuration as described in the system overview and setup. For each measure the best score is highlighted.

Table 1. Results of our stemming experiments

stemmer	**map**	**gmap**	**bpref**	**r-precision**
standard	0.2336	0.1423	0.3115	0.3130
less aggressive	0.2338	0.1421	0.3108	0.3140
hunspell	0.1739	0.0928	0.2468	0.2517
no stemmer	0.1534	0.0575	0.2239	0.2117

The difference between standard and less aggressive stemming is marginal and cannot be rated as significant. Without any stemmer, the results are the lowest and the HunSpell stemmer performed slightly better than no stemming at all.

4 Conclusions and future work

It is evident that stemming improves the results, as the experiment with no stemming scored below every other stemming approach. Despite HunSpell stemming scoring higher than no stemming, it should not be considered a beneficial approach, because the results do not correspond to the processing time that must be dedicated to the stemming. However, there are languages without any other stemming algorithms, which could benefit from the HunSpell stemming.

This should be investigated further, especially in matters of processing time for these languages.

Another benefit of our participation was the addition of the Apache Solr interface. Now the Xtrieval framework is able to use Solr for indexing and retrieving documents. Furthermore existing retrieval interfaces like AJAX Solr[3] can be used to inspect the index and the documents in a more interactive way than before.

References

[1] Kürsten, J., Wilhelm, T.: Extensible retrieval and evaluation framework: Xtrieval. In Baumeister, J., Atzmüller, M., eds.: LWA. Volume 448 of Technical Report., Department of Computer Science, University of Würzburg, Germany (2008) 107–110

[2] Smiley, D., Pugh, E.: Apache solr 3 enterprise search server (2011)

[3] Kürsten, J., Eibl, M.: Monolingual retrieval experiments with a domain-specific document corpus at the chemnitz university of technology. [6] 178–185

[4] Halácsy, P., Trón, V.: Benefits of resource-based stemming in hungarian information retrieval. [6] 99–106

[5] Kürsten, J., Wilhelm, T., Richter, D., Eibl, M.: Chemnitz at the chic evaluation lab 2012: Creating an xtrieval module for semantic enrichment. In Forner, P., Karlgren, J., Womser-Hacker, C., eds.: CLEF (Online Working Notes/Labs/Workshop). (2012)

[6] Peters, C., Clough, P., Gey, F.C., Karlgren, J., Magnini, B., Oard, D.W., de Rijke, M., Stempfhuber, M., eds.: Evaluation of Multilingual and Multi-modal Information Retrieval, 7th Workshop of the Cross-Language Evaluation Forum, CLEF 2006, Alicante, Spain, September 20-22, 2006, Revised Selected Papers. In Peters, C., Clough, P., Gey, F.C., Karlgren, J., Magnini, B., Oard, D.W., de Rijke, M., Stempfhuber, M., eds.: CLEF. Volume 4730 of Lecture Notes in Computer Science., Springer (2007)

[3] https://github.com/evolvingweb/ajax-solr/