Corpus based evaluation of stemmers

István Endrédy

Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
MTA-PPKE Hungarian Language Technology Research Group
50/a Práter Street, 1083 Budapest, Hungary
endredy.istvan@itk.ppke.hu

Abstract

There are many available stemmer modules, and the typical questions are usually: which one should be used, which will help our system and what is its numerical benefit? This article is based on the idea that any corpora with lemmas can be used for stemmer evaluation, not only for lemma quality measurement but for information retrieval quality as well. The sentences of the corpus act as documents (hit items of the retrieval), and its words are the testing queries. This article describes this evaluation method, and gives an up-to-date overview about 10 stemmers for 3 languages with different levels of inflectional morphology. An open source, language independent python tool and a Lucene based stemmer evaluator in java are also provided which can be used for evaluating further languages and stemmers.

1. Introduction

Information retrieval (IR) systems have a common critical point. It is when the root form of an inflected word form is determined. This is called stemming. There are two types of the common base word form: lemma or stem. The lemma is the base or dictionary form of a word, which is the grammatically correct, morphological root of the word. The stem, however, is a common root form of a word, which is made usually by cutting typical endings from the end of the word. The stem need not be a full word. For instance, the word form *spies* has the lemma *spy* and its stem might be just *sp*. Stemmers are thus algorithms that remove certain endings from word forms. Lemmatizers, however, are special stemmers that produce the dictionary form of a word. Lemmatizers are able to deal with irregular word forms, they perform dictionary and morphological operations on the input. They usually use more resources than stemmers. Therefore, every lemmatizer is a high quality and more resource-intensive stemmer.

The advantage of the lemmatizers is that they exactly know the words, and they have good precision on known words. But they usually can not handle unknown words. On the contrary, there are no unknown words for algorithmic stemmers. They are much simpler, smaller and usually faster than lemmatizers, and for many applications their results are good enough. For instance, the final result of an IR does not require to define exact lemma, it is enough to assign the same stem to the related word forms.

One of the famous algorithmic stemmers is the Porter stemmer for English (Porter, 1980). This logic was expanded, and a programming language, Snowball was created for defining stemming algorithms (Porter and Boulton, 2001).

Stemming has strong impact not only on the precision and recall of searching, but it can also reduce the size of the index. In the case of agglutinative languages (such as Finnish or Hungarian) where a single word might have thousands of word forms, stemming is essential.

2. Related work

Stemmers can be evaluated in several ways. First, their impact on IR performance is measurable by test collections. The earlier stemmer evaluations (Hull, 1996; Tordai and De Rijke, 2006; Halácsy and Trón, 2007) were based on pre-defined queries and their expected document sets. These sets were called experimental collections. A few hundreds of queries were constructed by hand and evaluated on thousands of documents for relevance with respect to these queries. This dataset had to be made very carefully: the selection had great impact on the final results. Stemmers were evaluated against this dataset. Ranking was also the part of the evaluation.

These evaluations might show the usefulness of a stemmer, but a stemmer developer does not get feedback about the typical errors of the module. Thus error metrics (Paice, 1994) were defined which might give an insight into the comparison of stemmers and can also give detailed information about errors. There are basically two types of stemming error: overstemming and understemming. Overstemming index (OI) shows how many times a stemmer wrongly cuts too many letters from a word form. The higher is the OI, the more unrelated words are connected to the same stem in an IR, and precision is decreased. Understemming index (UI) counts how many times a stemmer misses to remove a suffix. This type of error results in the separation of related word forms. In an IR it causes lower recall. The ideal case is when OI and UI are both zero. The rate of OI / UI means the weight or strength of a stemmer: a light stemmer strips only fewer affixes from words, a heavy one eats them with greater appetite. This weight does not represent the quality of a stemmer, it only characterizes its suffix stripping mode. Another rate was defined to compare stemmers: error rate relative to truncation (ERRT) which shows the distance of the (UI, OI) series from the ideal one.

There are several available open source IR systems, for instance Lucene (Hatcher et al., 2004) and systems based on it: Apache Solr (Smiley et al., 2015), Elasticsearch (Gormley and Tong, 2015). They are shipped with more stemmers. However, it is not easy to decide which stemmer to choose for a given system. Previous evaluations are older than these systems, and there was no available evaluation about them.

This article would like to give an up-to-date overview about stemmers, and also to provide a simple method and a language-independent python and java tool [1], which can be used to compare and evaluate any stemmers for further languages.

3. Metrics on the direct output of the stemmers

Stemmers are evaluated by six metrics. First, a corpus with lemmas is needed, whose lemmas will be the gold standard. Second, each word of the corpus is stemmed with various stemming modules. Their results can be compared to the gold standard. Some languages may have more lemmas for the same word form. Most of the leading search tools are able to handle this phenomenon (Lucene, Elastic search, MS Indexing Service, MSSQL Fulltext search, etc), others not (e.g. Tovek tools). A stemmer module typically has one input word without its context, so it has no chance to disambiguate its stem alternatives. It might provide only a heuristic ranking. There can be more strategies: use the first one, use the most probable one or use all of them.

The tool counts lemma accuracy first: how many times the first lemma is correct. Secondly, another 3 metrics are applied which mark the false alternatives in a strong and a weak way. The latter one is called *average precision at maximum recall* (Lindén, 2009), for *n* lemmas before the correct one, we get a precision of 1/(*n*+1). The benefit of this metric that it evaluates the lemma ranking of the stemmer.

Other metric assumes that every lemma will be stored, so each incorrect lemma means false positive (fp), correct lemmas mean true positive (tp), and missing one means false negative (fn).

Third metric concentrates on the first stem: correct means tp, incorrect is fn, and fp is the case when the correct stem exists in the alternatives (but not at the first place).

Fourth metric takes into consideration all correct lemmas occured in the corpus compared to the output of a stemmer. Each word form in the corpus gets its all lemmas, and these sets were compared to the output of a stemmer. Their intersection is tp, lemmas only in corpus are fn, and lemmas only from stemmer are fp.

In addition, Paice metrics described in Section 2. are also counted: UI, IO, ERRT by NLTK library (Bird, 2006). It is based on list of words with their lemmas and stems, which is available. In each metric F-score is the harmonic mean of the recall and the average precision.

To sum it up, five metrics are applied to the output of the stemmers, which measure their lemma accuracy, their lemma ranking algorithm, their incorrect lemmas, their lemmas compared to all possible correct lemmas and Paice metrics as well. No doubt, the stemmers have disadvantage in this evaluation (they do not produce a lemma by design), but this metrics might give an overview about the stemmer and the language.

4. Metrics on the IR quality of the stemmers

The main idea of this article is that every corpus with lemmas can be used as an IR evaluation data set. In the tool presented here, every sentence in the corpus will be the result item of an IR (hit), and its words (in their original forms) are the queries. These word-sentence connections will be used as a gold standard, and each stemmer will be tested against this: calculation of precision and recall is based on the sets of sentences determined by stemmers and by the gold standard (illustrated on Figure 1).

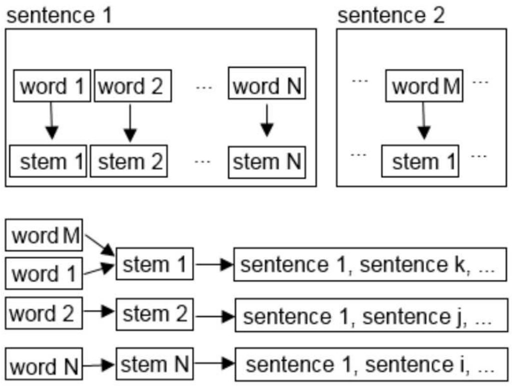

Figure 1: IR quality evaluation of a stemmer, based on a corpus with lemmas: sentences=documents and words=queries, sentence sets are compared to the gold standard

On the one hand, the tool contains a Lucene evaluation java code as well: sentences of the corpora (as separate documents) are indexed by stemmers, and each word (as a query) gets a result set which is compared to the gold standard. This is a classical collection set based evaluation, but the collection is big (millions of doc) and it is made automatically from the corpus. This java code evaluates in two ways: evaluates all result items, and evaluates only the first *n* results. This latter option reflects the mode when a human verifies the results: only the first *n* items matter.

4.1. Experimental evaluation method

On the other hand, there is another approach without IR system. Every word of the corpus has a sentence set, and gold standard sets are defined by the hand annotated lemmas of the corpus, while test sets come from the stemmers. The following F-score calculation makes it possible to evaluate the usefulness of a stemmer without the evaluation of lemma precision or even using an IR system. This means that during a test process each word of the corpus is stemmed by the tested modules, and our evaluation tool stores the original word form, its stem(s) and the sentence identifier (SID). Stopwords are skipped. At the end of the process, the SID of every input word is compared to the gold standard. A standard metric was used for measuring performance. If a SID of an input word from a stemmer is in the gold standard, it means true positive, if not, it means

[1] source code of the tool is available at `https://github.com/endredy/stemmerEval`

false positive. Finally, if a SID of a word exists in the gold standard but it is missing from the same word SID set of a stemmer, it means a false negative result.

The F-score of this evaluation method correlated with the native IR evaluation made by Lucene (Section 4.), if all result items were evaluated (not only the first *n*). It was verified by Pearson correlation.

The false positive errors are usually caused by an overstemmed word: for instance *occasions* gets the stem *occas*, so it might get hits with *occasionally, occasionalism* or even worse. These false positive hits reduces precision. However, a real IR system has ranking, which might hide the noisy hits of overstemming: these hits are ranked towards the end of the result list, thus these false results do not cause problems in practice. How can we simulate ranking in this test? We assume that the true positive hits are top-ranked by the IR, therefore we apply a logarithmic metrics on the false positive results. This metric reflects a ranked hit list, where the items have decreasing importance on the list. (This is similar to the case, when a human usually uses only the first n result pages of a search. Less accurate hits are acceptable on later pages, as the searcher probably never visits them.)

This method evaluates the quality of a stemmer or lemmatizer alone assuming that the original word form is not added to the index. It just measures how exactly a stemmer clusters different inflected word forms.

5. The evaluation tool

The previously described methods were implemented. Gold lemma extraction from corpora and metrics described in Section 3. and 4.1. were implemented in a Python, Lucene based IR evaluation (method of Section 4.) was implemented in java. These tools were used in the evaluation of 10 stemmer modules for 3 languages. First, the test inputs and their gold standards were generated from a corpus with lemmas. Each corpus has its own format, therefore the extraction script needed customization to have the following, uniform (word, lemma) tuples in a file, each tuple in a new line. Sentences were delimited by an empty line. Second, the stemmers were run on the input words, and their outputs were stored in separate files. These files and the gold standard were the inputs of our tool: it checked every word of the corpus, and executes the methods described in Sections 3. and 4.

The native IR evaluation starts with extracting the sentences from corpora, each sentence gets an ID, and all unique word forms get a sentence ID list: the given word occurs in those sentences. This will be the gold standard. Then, each sentence was indexed as a separate document by every stemmer, and all unique words of the corpus were the queries. The result sets were compared to the gold standard. These codes are available.

6. Tested stemmer modules

One of the most popular open source stemmer is **Hunspell**, which is widely used, mainly in open source projects. At the time of writing this article, more than 30 applications use it, including LibreOffice, OpenOffice, Firefox, Thunderbird and Google Chrome. Basically it is used for spell checking, but it can lemmatize as well. Hunspell lexicons were created for several languages.

The **Snowball** stemmer is an algorithmical one, it can handle many languages, and it is available in several applications. During this evaluation it was used from NLTK sdk (Bird, 2006), but it is also available from the Lucene engine. Snowball was adapted to Hungarian, too (Tordai and De Rijke, 2006). A morphological analyzer **Humor** (High-speed Unification MORphology) (Prószéky and Kis, 1999) has a lemmatizer module as well, which was also evaluated. It was developed by the author. This product can be found in several applications (Microsoft products, web applications), and it is not open source. It has lexicons for more than 20 languages. **Hunmorph**(Trón et al., 2005) and **Hunmorph-foma** (Hulden, 2009) were also tested. Apache Lucene (Hatcher et al., 2004) is a popular, high-performance search engine library with several stemming modules. Some of them were tested: **KStem**, **Porter** (Porter, 1980) and **EnglishMinimal** for English, and **Stempel**, **Morfologik** for Polish. (Lucene also has a Hunspell java implementation, but the original, c++ version was used.)

7. Corpora used in evaluation

Evaluation was done in three languages, which represent different levels of inflectional morphology: English (low), Polish (medium) and Hungarian (high). First, **British National Corpus** (Clear, 1993) was used, which is a 100 million word snapshot of British English (with lemmas). The corpus contains more than 2,000,000 sentences from the widest possible range of linguistic productions, from several domains. Evaluation was made with the 3rd XML edition of the BNC.

A 1-million word subcorpus (Degórski and Przepiórkowski, 2012) of the **National Corpus of Polish** is available with 64,000 sentences, which was used for Polish tests. It is also a well balanced large corpus, with several domains: it contains classic literature, daily newspapers, specialist periodicals and journals, transcripts of conversations, and a variety of short-lived and internet texts.

Hungarian tests were run on the **Szeged Treebank** (Csendes et al., 2005), the biggest Hungarian manually annotated corpus which contains about 80,000 sentences with 1,200,000 words annotated with lemmas.

8. Results

The size of this article do not allow to show every evaluation table (6+2 per languages), therefore two aspects were selected: first lemma evaluation (3rd metric described in Section 3.) and IR evaluation by Lucene (Section 4.). The results of the English stemmers are interesting. Recall of Kstem, Porter, En-minimal and Snowball are high but their precisions are very low. It is caused by millions of fp hits. (On the contrary, without a stemmer the precision is high but recall is low.) One reason for these results is that the stemmers above split words with a hyphen into more words (for instance *low-print-run* will be *low, print* and *run*), which causes low precision. Another type of error is overstemming: they cluster too many different

domain	tokens	no stem		Kstem		Porter		En-minimal		Snowball		Hunspell		Humor	
		F	oov	F	oov	F	oov	F	oov	F	oov	F	oov	F	oov
ACPROSE	15.7M	78.4	0	92.6	0	67.1	0	91.6	0	71.4	0	87.1	9.8	**93.3**	4.8
CONVRSN	4.2M	73.6	0	**93.6**	0	82.6	0	91.6	0	87.0	0	90.5	5.3	88.5	4.9
FICTION	16.1M	72.8	0	90.3	0	75.8	0	86.4	0	79.5	0	86.6	6.1	**91.4**	3.9
NEWS	9.4M	69.8	0	**92.1**	0	74.7	0	90.0	0	79.4	0	86.1	11.9	91.4	7.4
NONAC	24.1M	74.4	0	92.1	0	70.1	0	90.6	0	74.5	0	86.7	9.9	**92.6**	5.3
OTHERPUB	17.9M	74.2	0	**92.3**	0	73.4	0	91.3	0	77.6	0	87.5	9.8	92.3	6.0
OTHERSP	6.1M	78.6	0	**94.3**	0	77.9	0	92.8	0	82.0	0	90.6	5.1	90.6	5.6
UNPUB	4.4M	72.2	0	**92.7**	0	72.8	0	91.3	0	76.6	0	87.7	9.1	92.6	5.5
total	98.3M	74.4	0	**92.2**	0	72.7	0	90.4	0	77.0	0	87.3	9.0	92.1	5.4

Table 1: First lemma evaluation of English stemmers on gold standard lemmas of the BNC (Section 3.)

words together, which words only have the same prefix. This decreases precision drastically. Nevertheless, overstemming and splitting words by hyphens cause larger index and lower precision (Table 1, 2). The stemmers differ significantly in method 2 in Section 4. (according to ttest), except Porter and Kstem.

Polish results also show that stemming quality has great impact on the F-score of the IR. A stemmer can improve the F-score of an IR system at least 2 or even 2.5 times (see Table 4), the best scores belong to Morfologik stemmer in both testing methods (Table 3 and 4).

The results of the Hungarian stemmers can be seen in Table 5, 6 and 7. Since this language is heavily agglutinative, stemming improves the F-score significantly, even with the weakest stemmer (more than 50% difference comparing to the results without any stemmer). The best scores belong to the Humor stemmer in both methods. Algorithmic stemming is not appropriate for this language. According to ttest, Hunmorph and Hunmorph-foma stemming results have no significant differences, the others have.

Another important property is the speed of a stemmer: how many tokens can be stemmed in a second by the given module. It correlates with the time of reindexing, which can be critical in production environment. These performances can be seen on Table 8. Stemmers proved to be 4-5 times faster than lemmatizers.

stemmer	tokens/second
En-minimal	**86 165**
Kstem	85 353
Porter	84 032
Snowball	47 651
Stemfel	29 173
Morfologik	19 862
Hunmorph	275
Hunmorph-foma	27 850
Hunspell	13 105
Humor	5 629

Table 8: Speed of the stemmers, stemmed tokens/second (on a 8-core 1.1GHz CPU, 74GB memory, 64 bit ubuntu server)

9. Conclusion

In agglutinative languages the characteristic of the two evaluation methods are similar: the better the quality of the lemmatization, the better the quality of the IR will be. In the case of English, this connection is not verified. The corpus based evaluation makes it possible to create automatically gold standard for IR evaluation of stemmers. An experimental IR evaluation method was defined which works without a real IR system, and its result correlated with the real IR evaluation made by Lucene (Pearson correlation).

When the IR evaluation was made only on the first *n* results, it was designed to simulate a human verification. However, it is able to also evaluate the ranking algorithm of the IR system itself.

The defined six metrics on the direct output of the stemmers make it possible not only to compare stemmers but to give feedback about typical stemmer errors, which can be useful at developing stemmers.

This evaluation gives an up-to-date overview about available stemmers, and it provides also a tool which can be used for evaluating other languages and stemmers.

Acknowledgments.

I am grateful to Dr Nóra Wenszky, Dr Gábor Prószéky for their help and encouragement, to Attila Novák for thinking together, and to the anonym reviewers for their valuable suggestions.

10. References

Bird, Steven, 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL*. ACL.

Clear, Jeremy H., 1993. The digital word. chapter The British National Corpus. Cambridge, MA, USA: MIT Press, pages 163–187.

Csendes, D., Csirik J., Gyimóthy T, and Kocsor A, 2005. The Szeged Treebank. In *Text, Speech and Dialogue*. Springer.

Degórski, Łukasz and Adam Przepiórkowski, 2012. *Recznie znakowany milionowy podkorpus NKJP*. Wydawnictwo Naukowe PWN, pages 51–58.

Gormley, Clinton and Zachary Tong, 2015. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc.

Halácsy, Péter and Viktor Trón, 2007. Benefits of resource-based stemming in Hungarian information re-

domain	**no stemmer**	**Kstem**	**Porter**	**En-minimal**	**Snowball**	**Hunspell**	**Humor**
academic prose	42,5	**58,8**	51,7	55,6	49,8	53,4	48,7
conversation	52,9	72,8	75,8	67,4	74,3	65,7	**77,4**
fiction and verse	46,0	**68,8**	68,6	60,1	64,5	62,5	67,1
newspapers	52,3	73,3	69,4	67,4	67,4	65,9	**78,4**
non-academic prose	48,3	**66,3**	58,6	63,2	55,8	59,4	54,7
other published materials	46,9	**66,5**	62,0	62,3	60,2	59,9	55,9
any other spoken text	52,4	74,3	70,7	68,7	69,2	64,9	**78,0**
unpublished materials	48,4	**68,9**	64,4	62,7	63,5	61,5	60,0
total	47,2	**66,5**	61,4	61,8	59,0	59,9	59,0

Table 2: IR quality evaluation of English stemmers on the sentences of the BNC by Lucene (Section 4.)

domain	**tokens**	**no stemmer**		**Stempfel**		**Morfologik**		**Hunspell**		**Humor**	
		F	oov	F	oov	F	oov	F	oov	F	oov
literature	54 205	53.80	0	78.36	2.7	**89.34**	2.7	84.45	5.6	88.71	7.8
information and informative	56 779	54.08	0	77.31	3.9	**89.09**	3.9	84.38	7.7	87.87	5.8
conversations	59 024	68.26	0	72.60	9.5	**83.62**	9.5	78.58	11.3	79.63	12.8
fiction	169 270	55.81	0	77.48	4.1	**88.98**	4.1	84.38	5.9	87.84	4.2
spoken radio	23 303	64.52	0	73.94	6.4	**86.11**	6.4	82.12	8.1	83.43	10.2
research and teaching	20 229	50.23	0	79.56	3.6	**89.64**	3.6	85.90	7.1	89.38	5.8
internet	72 273	55.27	0	77.15	3.2	**88.78**	3.2	83.48	8.0	86.02	7.8
journalistic	506 214	51.78	0	79.23	2.8	**90.03**	2.8	86.10	6.1	89.16	4.6
written-Parliament	66 315	51.24	0	78.77	4.3	89.85	4.3	85.48	7.4	**89.86**	2.7
utilities	30 998	52.22	0	80.82	1.2	90.65	1.2	87.48	7.5	**91.38**	2.3
not classified	10 140	54.38	0	78.72	2.0	**90.11**	2.0	85.55	3.9	88.52	3.9
total	1 028 671	54.21	0	78.20	3.6	**89.25**	3.6	85.02	6.7	88.08	5.4

Table 3: First lemma evaluation of Polish stemmers on gold standard lemmas of the 1-million words PNC (Section 3.)

trieval. In *Evaluation of Multilingual and Multi-modal Information Retrieval*. Springer, pages 99–106.

Hatcher, Erik, Otis Gospodnetic, and Michael McCandless, 2004. Lucene in action.

Hulden, Mans, 2009. Foma: a finite-state compiler and library. In *EACL*. ACL.

Hull, David A., 1996. Stemming algorithms - a case study for detailed evaluation. *Journal of the American Society for Information Science*, 47:70–84.

Lindén, Krister, 2009. Entry generation by analogyencoding new words for morphological lexicons. *Northern European Journal of Language Technology*, 1(1):1–25.

Paice, Chris D, 1994. An evaluation method for stemming algorithms. In *Proceedings of the 17th ACM SIGIR conference on Research and development in information retrieval*. Springer.

Porter, Martin F, 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Porter, MF and Richard Boulton, 2001. Snowball stemmer.

Prószéky, Gábor and Balázs Kis, 1999. A Unification-based Approach to Morpho-syntactic Parsing of Agglutinative and Other (Highly) Inflectional Languages. In R. Dale and K. Church (eds.), *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics.

Smiley, David, Eric Pugh, Kranti Parisa, and Matt Mitchell, 2015. *Apache Solr Enterprise Search Server*. Packt Publishing Ltd.

Tordai, Anna and Maarten De Rijke, 2006. *Four stemmers and a funeral: Stemming in Hungarian at CLEF 2005.*

Trón, Viktor, András Kornai, György Gyepesi, László Németh, Péter Halácsy, and Dániel Varga, 2005. Hunmorph: open source word analysis. In *Proceedings of the Workshop on Software*. ACL.

domain	no stemmer	Morfologik	Stempel	Humor	Hunspell
literature	44.6	**85.9**	75.8	80.6	81.5
information and informative	43.7	**86.3**	75.2	80.5	81.9
conversations	39.0	**83.2**	70.6	67.7	80.2
fiction	32.8	**85.9**	72.5	77.2	80.8
spoken-Radio	49.0	**85.3**	76.9	76.7	82.0
research and teaching	53.7	**90.4**	82.2	86.8	87.0
internet	40.1	**81.3**	70.9	72.8	77.8
journalistic	26.3	**85.4**	72.3	78.6	81.2
written-Parliament	42.8	**92.6**	77.3	87.4	90.6
utilities	42.8	**92.1**	81.5	87.9	89.5
not classified	63.0	**88.8**	81.8	85.2	84.4
total	31,3	**85,7**	73,0	78,6	81,6

Table 4: IR quality evaluation of Polish stemmers on the sentences of the 1-million words PNC by Lucene (Section 4.)

domain	tokens	no stemmer		Hunspell		Hunmorph-foma		Hunmorph		Snowball		Humor	
		F	oov	F	oov	F	oov	F	oov	F	oov	F	oov
literature	185 436	53.1	0	90.7	6.1	83.8	11.9	85.9	1.8	65.0	0	**93.5**	2.5
student	278 497	50.3	0	92.3	2.6	83.5	11.3	88.8	1.2	61.5	0	**93.1**	0.7
newspaper	182 172	59.9	0	88.2	7.8	84.4	14.2	80.1	4.0	72.8	0	**94.6**	1.8
IT	175 991	59.7	0	85.1	13.5	82.9	18.1	80.3	6.5	76.4	0	**93.5**	4.5
juristic	220 069	66.2	0	85.4	11.1	81.5	22.2	81.4	8.6	81.0	0	**91.5**	3.2
business	186 030	61.7	0	83.6	11.6	84.4	17.3	77.1	7.8	73.6	0	**95.1**	2.1
total	1 228 195	58.6	0	87.8	8.5	83.4	15.8	82.4	5.1	71.9	0	**93.5**	2.4

Table 5: First lemma evaluation of Hungarian stemmers on Szeged TreeBank (Section 3.)

domain	no stemmer	Hunspell	Hunmorph-foma	Hunmorph	Snowball	Humor
literature	27.5	91.8	92.8	86.7	68.7	**96.2**
student	20.2	95.1	93.5	91.2	68.7	**97.6**
newspaper	41.4	86.4	90.4	76.0	78.1	**95.4**
IT	25.1	89.9	91.1	85.0	81.9	**96.7**
juristic	19.7	91.1	89.6	88.4	87.2	**95.8**
business	25.5	90.1	93.3	90.3	82.3	**97.8**
total	24.5	91.4	91.8	87.5	78.3	**96.7**

Table 6: Experimantal IR quality evaluation of Hungarian stemmers on the sentences of Szeged TreeBank (Section 4.1.)

domain	no stemmer	Hu-light	Snowball	Hunspell	Humor
literature	19.9	62.4	62.6	80.3	**83.0**
student	14.6	62.0	62.0	80.2	**83.8**
newspaper	26.5	69.4	72.8	74.7	**84.6**
IT	21.3	76.2	78.1	79.0	**87.1**
juristic	16.5	80.6	82.0	77.5	**87.1**
business	25.5	78.4	78.4	46.4	**90.8**
total	19.4	71.1	72.2	72.2	**85.7**

Table 7: IR quality evaluation of Hungarian stemmers on the sentences of Szeged TreeBank by Lucene (Section 4.)