String Distance-Based Stemming of the Highly Inflected Croatian Language

Jan Šnajder and Bojana Dalbelo Bašić
Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
{*jan.snajder, bojana.dalbelo*}@*fer.hr*

Abstract

Stemming refers to the grouping of morphologically related words into so-called *stem classes* for the purpose of improving information retrieval performance. Traditional approaches to stemming are language-specific and require a substantial amount of linguistic knowledge. A viable alternative is string distance-based stemming, in which stem classes are obtained by clustering word-forms from a corpus. In this paper, we apply string distance-based stemming to the highly inflected Croatian language using a number of string distance measures proposed in the literature. We focus on evaluating the stemming performance at both inflectional and derivational level, and investigate how this performance relates to the choice of the distance threshold value. Although our focus is on the Croatian language, we believe our results transfer well to languages of similar morphological complexity.

Keywords

Stemming, morphology, string distance,Croatian language

1 Introduction

Most information retrieval (IR) systems represent documents simply as a collection of words. The performance of such systems is negatively affected by the fact that words in texts appear in various morphological forms, either as the result of *inflection* (transformation of a word into various word-forms) or *derivation* (transformation of a word into new, but semantically related words). To reduce morphological variation, IR systems typically rely upon some sort of *morphological normalization* to conflate the various morphological forms into a single representative form. Numerous studies have shown morphological normalization to be beneficial for IR; this has been shown for English [4], as well as for other, more morphologically complex languages [13, 15].

The most common morphological normalization technique is *stemming*. Stemming refers to the removal of affixes from word-forms, yielding a stem common to all word-forms. The well-known Porter algorithm [11] is an example of such a rule-based approach to stemming. More generally, *stemming* refers to the process of grouping morphologically related word-forms into the so-called *stem classes*. Traditional rule- and dictionary-based stemming requires significant linguistic expertise and resources, which is why, for resource-poor languages, language-independent approaches are gaining popularity. Among others, string distance-based stemming, in which stem classes are derived by clustering word-forms based on their character structure, has been shown to be a viable alternative. String distance-based clustering was first proposed by Adamson and Boreham [1] for the English language, and similar approaches were later employed for Arabic [12] and Turkish [3]. More recently, Majumder et al. [9] proposed string distance measures for stemming in Bengali, as well as in Hungarian and Czech [8]. The performance of their stemming procedure has been shown to be comparable to traditional rule-based approaches.

In this paper, we investigate the applicability of string distance-based stemming to the Croatian language. The Croatian language, much like other Slavic languages, is morphologically complex, especially in the form of inflection. Previous approaches to the stemming in Croatian are rule-based [5, 7] or dictionary-based [14, 16] and are restricted to inflectional morphology. To our knowledge, the work reported here is the first application of a language-independent stemming technique to the Croatian language. The main focus of our work is a detailed evaluation of stemming performance, performed at both inflectional and derivational levels. Although our focus is on the Croatian language, we believe our results transfer well to languages of similar morphological complexity.

The rest of the paper is structured as follows. The next section describes the details of our approach: the string distance measures used, the clustering algorithm, and the evaluation methodology. Section 3 presents and discusses the experimental results. Section 4 concludes the paper and outlines future work.

2 Methodology

2.1 String distance measures

A variety of string distance measures for the clustering of morphologically related word-forms have been proposed in the literature. In [1], a Dice coefficient based on character bigrams was used as a measure of string similarity. We generalize this approach to a distance measure based on arbitrary-length n-grams:

$$Dice_n(X,Y) = 1 - \frac{2c}{x+y}, \qquad (1)$$

where x and y are the total number of n-gram tokens in words X and Y, respectively, and c is the number of n-gram tokens common to X and Y. The intuition behind this measure is that, because morphologically related words have a number of morphemes in common, they will also have a number of n-grams in common.

In [9], string distance is computed by considering character matches up to the first mismatch and penalizing all subsequent character positions. Of the four measures proposed in [9], our preliminary experiments indicated that the following two measures are most promising:

$$D_3(X,Y) = \frac{n-m+1}{m} \sum_{i=m}^{n} \frac{1}{2^{i-m}}, \qquad (2)$$

$$D_4(X,Y) = \frac{n-m+1}{n+1} \sum_{i=m}^{n} \frac{1}{2^{i-m}}, \qquad (3)$$

where m is the position of left-most character mismatch, and $n+1$ is the length of the longer of the two strings. Intuitively, measure D_4 penalizes long non-matching suffixes, whereas measure D_3 also rewards long matching prefixes.

One widely-known string distance measure is the Levenshtein distance [6], also called the *edit distance*. Edit distance between two strings is the minimal number of insertion, deletion, and substitution operations needed to transform one string into another. Of the three measures listed, edit distance is least morphologically sensitive.

2.2 Clustering

Partitioning algorithms like the k-means algorithm are widely used for clustering due to their effectiveness and simplicity. Such algorithms are not directly applicable to string distance-based clustering because they require a vector space-based measure in order to compute the cluster centroids. Thus, similar to [9] and [1], we cluster the word-forms using a hierarchical agglomerative algorithm [2]. The algorithm starts by assigning word-forms to singleton clusters and proceeds by merging at each level the two least distant clusters until a single cluster remains. From the resulting cluster tree (the *dendrogram*), clustering at specific distance levels can be obtained. The distance between two clusters is typically computed as the maximum, minimum, or average distance between elements of the two clusters, referred to as *complete-linkage*, *single-linkage*, and *average-linkage* algorithm, respectively. Complete-linkage results in small and compact clusters, single-linkage results in elongated clusters, whereas the result of average linkage is somewhere in between. In our experiments, we use average-linkage clustering. Note that, although we use a hierarchical agglomerative algorithm, we make no use of the derived hierarchical structure.

The main drawback of hierarchical agglomerative clustering is its computational inefficiency. Typical implementation makes use of an $n \times n$ distance matrix, where n is the number of elements. Thus, the space complexity of the algorithm is $\mathcal{O}(n^2)$. To construct the complete dendrogram, the distance matrix is searched for the least distant cluster pair within each of the n iterations, yielding a time complexity of $\mathcal{O}(n^3)$. Because the number of distinct word-forms in a corpus is on the order of hundreds of thousands, this problem must be addressed somehow.

Our approach is to cluster in two consecutive steps: a divisive and an agglomerative step. The idea is to use the divisive step to partition the set of word-forms into pre-clusters and then to perform agglomerative clustering on each of the pre-clusters separately. Pre-clusters must be coarse-grained so that morphologically related word-forms are assigned to the same pre-cluster, otherwise they cannot be merged in the agglomerative step. Partitioning n word-forms into pre-clusters of size m reduces the space complexity to $\mathcal{O}(m^2)$ and the time complexity to $\mathcal{O}(nm^2)$.

A straightforward approach to pre-clustering is to compute the equivalence classes of word-forms sharing a common prefix of a specified length l. We will denote this partition by $P(l)$. The size of pre-clusters is inversely proportional to l, but so is the quality of pre-clustering. If we consider longer prefixes, more morphologically related word-forms will end up being assigned to distinct pre-clusters. This problem suggests that the procedure can be further improved by taking into account the size of the obtained pre-clusters. The idea is to increase the specified prefix length and recursively partition only those clusters whose size is above the specified threshold. This procedure results in size-bounded pre-clusters of maximal quality. We will denote this partition by $M(s)$, where s is the maximum size of the pre-clusters.

2.3 Evaluation

Stemming algorithms are traditionally evaluated extrinsically, i.e., by considering their effect on the performance of IR systems. Such task-specific evaluation makes it impossible to distinguish between the case where the stemmer makes faulty conflations and the case of correct conflation not being beneficial for the task at hand. To address this, we use an intrinsic, task-independent evaluation first proposed by Paice [10]. This method evaluates a stemmer by counting the actual understemming and overstemming errors that the stemmer commits. The under- and overstemming errors are counted on a manually constructed word sample in which the words are grouped accordingly. The *understemming index UI* is computed as the proportion of pairs from the sample that are not conflated even though they belong to the same group, whereas the *overstemming index OI* is computed as the proportion of pairs that belong to different groups among those that are conflated to the same stem. The *stemming weight SW* is defined as the ratio *OI*/*UI*.

The word sample we used consisted of 10,000 distinct word-forms (nouns, verbs, and adjectives) from the Croatian newspaper "Vjesnik".[1] In order to make separate evaluation of both inflectional and derivational stemming performance possible, we grouped the

[1] `http://www.vjesnik.hr`

Table 1: *Examples of word groups from the sample*

`{{arheolog}}`
`{{arhitekt, arhitekta},` `{arhitekturi, arhitekture, arhitekturama},` `{arhitektonski, arhitektonskih}}`
`{{arhiva, arhivima, arhivu},` `{arhivske, arhivskim, arhivskoj}}`
`{{arija, arije, ariju}}`

word-forms manually at two distinct levels. *Inflectional groups* are comprised of inflectional word-forms and have clear-cut semantic boundaries. *Derivational groups* are comprised of word-forms from morphologically and semantically related inflectional groups. More precisely, two inflectional groups are joined together if the corresponding word-forms are derivationally as well as semantically related (in the case of polysemy, it suffices if some of their senses are related). A derivational group is then obtained by a transitive closure of this pairwise relation. The semantic relations between members of such groups are less clear and are often context dependent. Our sample consists of $5,508$ inflectional and $3,833$ derivational groups. Table 1 shows an excerpt from the sample in which 17 word-forms are grouped into seven inflectional and four derivational groups.

3 Experiments and discussion

The experiments were performed on a corpus comprised of $92,465$ articles from the newspaper "Vjesnik", amounting to over 23 million word-form tokens and 560,137 word-form types.

3.1 Pre-clustering

As discussed above, the purpose of the divisive clustering step is to decrease the complexity of agglomerative clustering. Because divisive clustering results in understemming, we aim at keeping understemming errors as low as possible, while at the same time obtaining small-sized pre-clusters.

The results for both aforementioned pre-clustering partitions are depicted in Table 2. For each partition, we give the number of pre-clusters, the size of the largest pre-cluster, and the inflectional (iUI) and derivational (dUI) understemming indices. The understemming indices reflect how many errors the algorithm makes by assigning inflectionally or derivationally related word-forms to distinct pre-clusters, while the size of the largest class determines the upper bound of the algorithmic complexity. The problem of pre-clustering with a common fixed-length prefix is that, in order to obtain pre-clusters of manageable sizes, the prefix length must be at least 5. This splits apart many inflectional groups, as indicated by the high understemming values. Size-bounded partitioning, on the other hand, can be used to obtain pre-clusters of manageable sizes, while at the same time committing much less understemming errors. In particular, partition $M(500)$ seems like a reasonable trade-off between

Table 2: *Size and understemming indices for partitions $P(l)$ and $M(s)$*

	Pre-clusters		Understemming	
Partition	Number	Largest	iUI %	dUI %
$P(1)$	32	72108	4.21	2.49
$P(2)$	808	29716	4.79	3.76
$P(3)$	9365	10774	7.45	7.09
$P(4)$	45794	2029	16.36	20.87
$P(5)$	113532	988	28.78	38.08
$M(5000)$	1501	4932	5.62	4.70
$M(2500)$	2873	2464	6.92	6.34
$M(1000)$	5732	1000	8.17	9.29
M(500)	**10316**	**498**	**10.80**	**13.48**
$M(250)$	18108	250	15.78	21.56
$M(100)$	37155	100	29.78	43.47

computational efficiency and stemming performance. Thus, for the divisive step, we use $M(500)$ and pre-cluster 560,137 word-forms into 10,316 pre-clusters.

3.2 Clustering

After partitioning the corpus into pre-clusters of a manageable size, we applied hierarchical agglomerative clustering on each pre-cluster separately. We used string distance measures $Dice_2$ and $Dice_3$, as defined by (1), measures D_3 and D_4, as defined by (2) and (3), respectively, and the edit distance. As a baseline, we used the partitioning method $P(l)$, which is equivalent to simply truncating a word-form to the first l characters.

The UI-OI plot on Fig. 1 shows the inflectional stemming performance of the five distance measures. As the value of the distance threshold increases, the understemming decreases and the overstemming increases. The plot reveals that all five measures perform far better than simple truncation. As expected, the edit distance performs worse than other measures. Measures D_3 and D_4 are of comparable performance and consistently outperform measures $Dice_2$ and $Dice_3$. The UI-OI plot for derivational stemming performance is shown on Fig. 2. Except for the edit distance, which performs considerably worse than the truncational baseline, the other string distance measures yield modest improvement over the baseline. Measures D_3, D_4, and $Dice_3$ perform at comparable levels, whereas measure $Dice_2$ performs slightly less well.

3.3 Optimal measure

A good string distance measure should commit as few under- and overstemming errors as possible. The trade-off between these two types of errors is similar to the trade-off between precision and recall in IR. As in IR, we define a composite measure of stemming performance, the *stemming quality*, as a harmonic mean between $1 - UI$ and $1 - OI$, as follows:

$$SQ = \frac{2(1 - UI)(1 - OI)}{2 - UI - OI}. \tag{4}$$

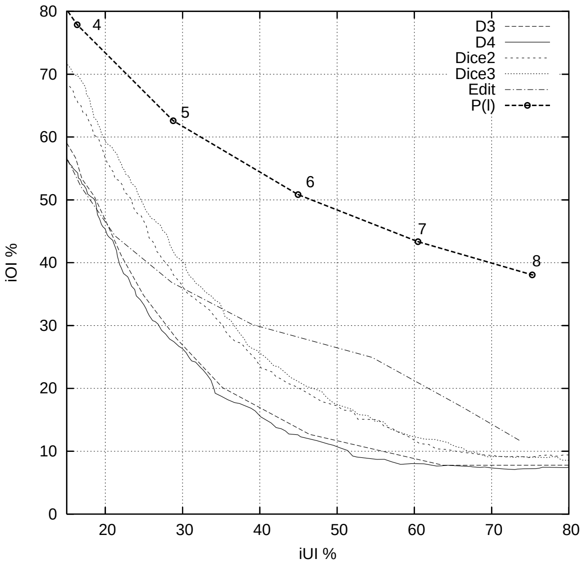

Fig. 1: *UI-OI plot for inflectional grouping*

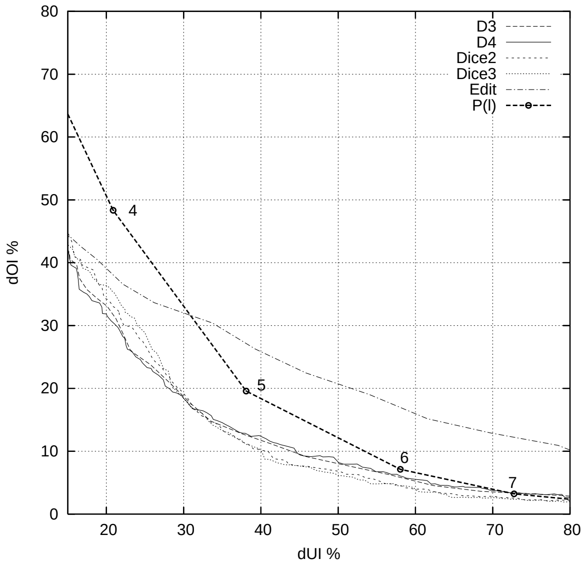

Fig. 2: *UI-OI plot for derivational grouping*

This assumes that under- and overstemming errors are equally important, though this may not be always the case. We use *iSQ* and *dSQ* to denote inflectional and derivational stemming quality, respectively; these tell us how well the string distance measure approximates the inflectional and derivational groupings.

Table 3 lists the optimal inflectional and derivational stemming quality of the five string distance measures and the corresponding threshold value t. Measure D_4 result in the best stemming quality. We can see that the stemming quality of all measures is worse on inflectional levels than on derivational levels, and that in most cases the understemming errors are more pronounced.

3.4 Optimal threshold value

The choice of the distance threshold is obviously crucial for stemming performance. A lower threshold value yields "light" and predominantly inflectional stemming, whereas a higher threshold value yields "heavy" and predominantly derivational stemming. The difficulty in choosing the optimal threshold value derives from the semantic relationships between derivationally related words being to a certain degree arbitrary. It is therefore difficult to decide how much derivational stemming is appropriate. What is certain, however, is that stemming should occur at least at the inflectional level, but should not extend beyond the derivational level. To account for this, we only have to consider inflectional understemming and derivational overstemming errors, and redefine the stemming quality SQ given by (4) in terms of indices iUI and dOI. Fig. 3 shows the so-defined stemming quality SQ of measure D_4, along with the inflectional and derivational stemming qualities iSQ and dSQ. Measure D_4 achieves optimal inflectional stemming quality for $t = 0.537$ and optimal derivational stemming quality

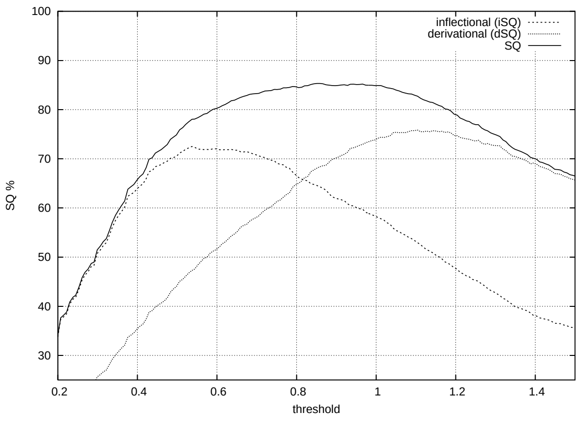

Fig. 3: *Stemming quality of measure D_4*

for $t = 1.104$ (cf. Table 3). Between these two extremes, the optimal stemming quality $SQ = 85.32\%$ is reached for $t = 0.859$; this is where measure D_4 makes the least number of inflectional understemming and derivational overstemming errors, 19.38% and 9.39%, respectively.

3.5 Discussion

Among the considered measures, stemming quality is best for measure D_4; this measure seem to capture best the inflectional and derivational morphology of the Croatian language, which is mostly suffixational. Apart from this, we can make two interesting observations. Firstly, inflectional stemming quality is consistently worse than derivational stemming quality, and improvement over the simple truncational baseline is much greater for inflection than for derivation. These results are probably due to the fact that, at the level of character structure, inflectional relations are less read-

Table 3: *Optimal inflectional and derivational stemming performance of five string distance measures*

	Inflection					Derivation				
Measure	t	iUI %	iOI %	iSQ %	iSW	t	dUI %	dOI %	dSQ %	dSW
D_3	0.813	35.17	20.13	71.56	0.57	3.047	23.15	25.95	75.42	1.12
D_4	0.537	34.22	**19.23**	**72.51**	0.56	1.104	27.61	20.36	**75.84**	0.74
$Dice_2$	0.332	36.18	28.15	67.60	0.78	0.560	31.35	**16.61**	75.31	0.53
$Dice_3$	0.376	38.41	26.87	66.86	0.70	0.668	30.97	16.84	75.44	0.54
Edit	3.075	**28.59**	36.86	67.02	1.29	5.711	**22.12**	36.58	69.91	1.65

ily discernible than derivational relations. Secondly, for both inflection and derivation, understemming errors are more pronounced than overstemming errors (i.e., $SW < 1$). This difference in errors can probably be attributed to pre-clustering, which introduces additional understemming errors.

With an appropriate threshold, the stemming quality of measure D_4 can reach to over 85%. This should be contrasted with stemming quality of simple truncation, which (on the same sample) reaches to $SQ = 75.55\%$, and lexicon-based inflectional normalization [16], reaching to $SQ = 95.07\%$. This result is in favor of string distance-based stemming as a language-independent approach to stemming. A more conclusive comparison will have to be done in an extrinsic, task-specific setting.

4 Conclusion

String distance-based stemming is an alternative to the traditional language-specific stemming approaches. We have applied string distance-based stemming to the morphologically complex Croatian language, using a number of string distance measures proposed in the literature. For reasons of computational efficiency, the clustering algorithm we used combines divisive pre-clustering with agglomerative clustering.

Intrinsic evaluation of stemming performance on the given sample has shown that certain string distance measures are more adequate than others for capturing Croatian inflectional and derivational morphology. By choosing an appropriate distance threshold, stemming quality considerably outperforms the truncational baseline, and stemming errors can be kept in the range of 10–20%. While this is likely to be acceptable for many IR applications, it remains to be proven on actual IR tasks, and we leave this for future work.

Additionally, in our future work, we intend to complement the string distance-based approach with some language-specific knowledge and investigate how this refinement improves stemming quality.

Acknowledgments

The authors would like to thank the reviewers for helpful comments and suggestions on an earlier draft of this paper. This work has been supported by the Ministry of Science, Education and Sports, Republic of Croatia, under the Grant 036-1300646-1986 and the Government of Flanders under the Grant KRO/009/06 (CADIAL).

References

[1] G. Adamson and J. Boreham. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Processing and Management*, 10(7/8):253–260, 1974.

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* John Wiley & Sons, New York, 2001.

[3] F. C. Ekmekcioglu, M. F. Lynch, and P. Willett. Stemming and n-gram matching for term conflation in Turkish texts. *Information Research News*, 7(1):2–6, 1996.

[4] D. A. Hull. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society of Information Science*, 47(1):70–84, 1996.

[5] D. Lauc, T. Lauc, D. Boras, and S. Ristov. Developing text retrieval system using robust morphological parsing. In V. H.-D. Damir Kalpić, editor, *Proceedings of 20th International Conference on Information Technology Interfaces (ITI'98)*, pages 61–65. SRCE, Zagreb, 1998.

[6] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[7] N. Ljubešić, D. Boras, and O. Kubelka. Retrieving information in Croatian: Building a simple and efficient rule-based stemmer. In *Digital information and heritage*, pages 313–320, Zagreb, 2007. Odsjek za informacijske znanosti Filozofskog fakulteta u Zagrebu.

[8] P. Majumder, M. Mitra, and D. Pal. Hungarian and Czech stemming using YASS. In *Working Notes for the CLEF 2007 Workshop*, 2007.

[9] P. Majumder, M. Mitra, S. K. Parui, G. Kole, P. Mitra, and K. Datta. YASS: Yet another suffix stripper. *ACM Transactions on Information Systems*, 25(4):18:1–18:20, 2007.

[10] C. D. Paice. Method for evaluation of stemming algorithms based on error counting. *Journal of the American Society for Information Science*, 47(8):632–649, 1996.

[11] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[12] A. D. Roeck and W. Al-Fares. A morphologically sensitive clustering algorithm for identifying Arabic roots, 2000.

[13] J. Savoy. Light stemming approaches for the French, Portuguese, German and Hungarian languages. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1031–1035, New York, NY, USA, 2006. ACM Press.

[14] M. Tadić and B. Bekavac. Inflectionally sensitive web search in Croatian using Croatian lemmatization server. In V. Lužar-Stiffler and V. H. Dobrić, editors, *Proceedings of 26th International Conference on Information Technology Interfaces (ITI'06)*, pages 481–486. SRCE, Zagreb, 2006.

[15] S. Tomlinson. Lexical and algorithmic stemming compared for 9 European languages with Hummingbird SearchServer at CLEF 2003. In *CLEF*, pages 286–300, 2003.

[16] J. Šnajder, B. Dalbelo Bašić, and M. Tadić. Automatic acquisition of inflectional lexica for morphological normalisation. *Information Processing and Management*, 44(5):1720–1731, 2008.