An Agglomerative Hierarchical Clustering Algorithm for Labelling Morphs

Burcu Can
Department of Computer Engineering
Hacettepe University
Beytepe, Ankara 06800 Turkey
burcu.can@hacettepe.edu.tr

Suresh Manandhar
Department of Computer Science
University of York
Heslington, York, YO10 5GH, UK
suresh.manandhar@york.ac.uk

Abstract

In this paper, we present an agglomerative hierarchical clustering algorithm for labelling morphs. The algorithm aims to capture allomorphs and homophonous morphemes for a deeper analysis of segmentation results of a morphological segmentation system. Most morphological segmentation systems focus only on segmentation rather than labelling morphs according to their roles in words, i.e. inflectional (cases, tenses etc.) vs. derivational. Nevertheless, it is helpful to have a better understanding of the roles of morphs in a word to be able to judge the grammatical function of that word in a sentence; i.e. the syntactic category. We believe that a good morph labelling system can also help part-of-speech tagging. The proposed clustering algorithm can capture allomorphs in Turkish successfully. We obtain a recall of 86.34% for Turkish and 84.79% for English.

1 Introduction

Most morphological segmentation systems (Creutz and Lagus (2002; Creutz and Lagus (2004; Goldsmith (2001)) perform only the segmentation of words and do not label morphs according to how they function in a word. As a rule, some morphemes function as inflective, whereas some morphemes function as derivative. However, we do not aim to distinguish inflection or derivation within a word, but we aim to distinguish between various types of morphs which are either inflective or derivative, e.g. allomorphs, homophonous morphemes. Labelling morphs not only helps with analysing the segmentation of a word, but can also help other natural language problems, i.e. part-of-speech tagging.

The main purpose of this paper is to serve as a post-processing tool to label morphs that have been discovered by a morphological segmentation system. Our main aim is directed towards the Morpho Challenge competition (Mikko Kurimo (2011)), which provides a platform to compare participant morphological segmentation systems. In Morpho Challenge, morph labels in a segmented word and the respective morph labels in its gold standard are compared.

Example 1.1 *For example, the gold standard analyses of 'arrangements' and 'standardizes' in Morpho Challenge are given as:*

arrangements	*arrange_V*	*ment_s*	*+PL*
standardizes	*standard_A*	*ize_s*	*+3SG*

Although in both analyses -s occurs, their labels are different; +PL (plural) and +3SG (third person singular).

There is not much work done in morpheme labelling. Spiegler (Spiegler, 2011) presents two algorithms for morpheme labelling: one of them learns morpheme labels once morphological segmentation is completed and the other finds morpheme labels during morphological segmentation. Both algorithms work in a supervised setting in which ground truth morphemes are provided. Bernhard (Bernhard, 2008) suggests another morpheme labelling algorithm which labels morphemes as a stem, suffix, base, or prefix. Therefore, the proposed labelling method does not consider any allomorphs or homophonous morphemes.

The paper is organised as follows: section 2 gives the intuition behind this work, section 3 describes our clustering algorithm, section 4 presents our experiment results, and finally section 5 and section 6 conclude the paper with a discussion on the obtained results.

2 Intuition

Most morphological segmentation algorithms consider only segmenting words into its morphs and ignore labelling morphs. However, morph labels are not only useful for other NLP problems (e.g. PoS tagging), but also they give a better understanding on the morphological analysis of words. There are different types of morphs having different grammatical functions. The algorithm presented in this paper aims to group morphs according to their functions within a word. This grouping is accomplished by considering two types of distinction among morphemes: allomorphs and homophonous morphemes.

2.1 Allomorphs

Morphs may differ in shape but still can carry out the same function in words, such as the plural morpheme *-s* and *-ies* in English. Allomorphs are also seen quite often in some languages where vowel harmony[1] takes place, such as in Turkish, Hungarian, Finnish, etc. Some examples in Turkish are given below:

- The plural form (i.e. *-lar, -ler*): e.g. *elma**lar*** (apples), *ev**ler*** (houses).
- The possessive case (i.e. *-in, -un, -ün*): e.g. *Ali'n**in*** (Ali's), *Banu'n**un*** (Banu's), *Üstün'**ün*** (Üstün's).
- The present tense (i.e. *-ar, -ir*): e.g. *yap**ar*** (he does), *gel**ir*** (he comes).
- The prepositional case (i.e. *-de, -da*): e.g. *ev**de*** (at home), *okul**da*** (in the school).

Vowel harmony is not the only phonological change which causes allomorphs in Turkish. Furthermore, morphs that are attached to an unvoiced consonant ending word are also harmonised and the first morph letters become also an unvoiced consonant (i.e. *p*, *ç*, *t*, *k*, *s*, *ş*, and *h*):

- The ablative case (i.e. *-den, -ten*): e.g. *ülke**den*** (from the country), *sepet**ten*** (from the basket).
- The locative case (i.e. *-de, -te*): e.g. *şehir**de*** (in the city), *kent**te*** (in the town).

[1]Vowel harmony involves rules on vowels that follow each other within a word.

- The third person singular (i.e. *-dir, -tir*): e.g. *nefis**tir*** (it is delicious), *zeki**dir*** (she is clever).

Due to vowel and consonant harmonies, Turkish comprises of many examples of morphs that have the same function but that are phonological variants of each other. It would be beneficial to group the allomorphs in the same cluster by assigning the same morpheme label as described before.

2.2 Homophonous morphemes

In contrast to allomorphs, some morphemes might sound the same phonetically; however, they might function differently. These morphemes are called homophonous morphemes (i.e. homophones). Homophonous morphemes belong to different clusters, due to the difference in their meanings. Some examples of homophonous morphemes in Turkish are given below:

- *kalem**i***: *-i* might correspond to either an accusative form (e.g. *his/her pen*) or a possessive form (e.g. *give me the pen*) which can be determined from the context.
- *yap**ın*** and *kapın**ın***: *-ın* corresponds to an imperative form in the first example, whereas it corresponds to a possessive form in the latter.
- *geliyor**lar*** and *yatak**lar***: *-lar* corresponds to 3^{rd} person plural in the first example, whereas it corresponds to plural in the latter.

Although homophonous morphemes do not occur as often as allomorphs, it is crucial to determine homophony in order to be able to distinguish morphemes which have different functions and thereby meanings. Homophonous morphemes should be grouped in different clusters; however, allomorphs should be grouped in the same cluster.

3 The Algorithm for Clustering Morphemes

For morph labelling, we propose a bottom-up agglomerative hierarchical clustering algorithm in which morphs showing functional similarities are clustered together. The functional similarities of the morphs are defined by a set of features as an input to the algorithm. Therefore, a feature vector is constructed to represent each morph by a feature vector. Each feature vector consists of a sequence of features which are given below:

- Current morph to be clustered.

- Previous morph that precedes the current morph in the analysis of the same word.
- Following morph that follows the current morph in the same word.
- Stem of the word.
- The last morph of the preceding word.
- The last morph of the following word.
- Morph position in the word (i.e. if the morph comes just after the stem, then it is 0. If the morph is the last morph of the word, then it is 2, and if it is surrounded by other morphs, this value is 1.)
- Morph length in letters.

Example 3.1 *In Turkish, the morph -ıl that occurs in the analysed sentence "O+n+lar ceza+lan+dır+**ıl**+acak+lar." (i.e. they will be punished) has got the features given below:*

- *Current morph: -ıl*
- *Previous morph: -dır*
- *Following morph: -acak*
- *Stem of the word: ceza*
- *The last morph of the preceding word: -lar*
- *The last morph of the following word: -*
- *Morph position in the word: 1*
- *Morph length: 2*

Constructing the feature vector of each morph initially, morph are placed in distinct clusters. In each iteration of the clustering algorithm, the two clusters having the minimum distance are merged. The distance between two clusters is measured by Kullback-Leibler (KL) divergence through all features in their feature vectors. Recall that KL divergence is not a distance metric, since it is not symmetric:

$$KL(p \parallel q) = \sum_i p(i) log \frac{p(i)}{q(i)} \qquad (1)$$

KL divergence can be formed into a symmetric measure $D(p \parallel q)$ as follows:

$$D(p \parallel q) = KL(p \parallel q) + KL(q \parallel p) \qquad (2)$$

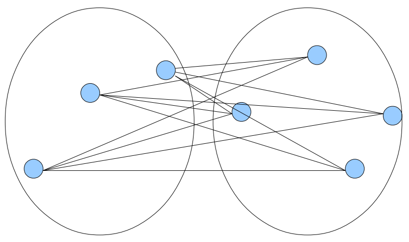

Figure 1: Average linkage clustering.

We use average linkage clustering, an instance of agglomerative clustering, for clustering morphs. In average linkage agglomerative clustering, the distance between two clusters is the average distance which is calculated through all pairs of data points in the clusters (see Figure 1):

$$D(R,S) = \frac{1}{N_R \times N_S} \sum_{i=1}^{N_R} \sum_{j=1}^{N_S} d(r_i, s_i) \qquad (3)$$

where the total distance between two clusters R and S with sizes N_R and N_S respectively is the summation of distances between each data pair in the clusters. The distance is normalised with the number of pairs. The cluster pair having the minimum distance is merged in each iteration.

In contrast to single-linkage and complete-linkage clustering, average-linkage clustering takes each data member into account; thereby leads to a more realistic measurement.

Using average linkage clustering, each cluster is defined by a feature vector which keeps all the information that comes from each morph in the cluster. For example, the previous morph in a cluster is a combination of all previous morphs that are owned by each morph in the cluster. While qualitative features are combined, quantitative features, such as morph position and morph length, are averaged for the feature vector of the cluster. Having a feature vector for each cluster, the similarity between two clusters, c_1 and c_2, is measured as follows:

$$\begin{aligned} Sim(c_1, c_2) &= \alpha D(CurMor_{c_1} \parallel CurMor_{c_2}) \\ &+ \beta D(PreMor_{c_1} \parallel PreMor_{c_2}) \\ &+ \delta D(FolMor_{c_1} \parallel FolMor_{c_2}) \\ &+ \gamma D(Stem_{c_1} \parallel Stem_{c_2}) \\ &+ \pi D(PreWMor_{c_1} \parallel PreWMor_{c_2}) \\ &+ \kappa D(FolWMor_{c_1} \parallel FolWMor_{c_2}) \\ &+ x|pos_{c_1} - pos_{c_2}| \\ &+ y|len_{c_1} - len_{c_2}| \qquad (4) \end{aligned}$$

where $CurMor_{c_1}$ denotes the set of current morphs $PreMor_{c_1}$ denotes the set of previous morphs, $FolMor_{c_1}$ denotes the set of following morphs, $Stem_{c_1}$ denotes the set of stems, $PreWMor_{c_1}$ is the set of last morphs of previous words and $FolWMor_{c_1}$ is the set of last morphs of following words in c_1. In addition to the qualitative features, quantitative features pos_{c_1} and len_{c_1} refer to the average position and the average length of the morphs belonging to the cluster c_1. Here, the quantitative features (i.e. pos_{c_i},len_{c_i}) are simply subtracted to find the distance between them. The weights of each feature are denoted by $alpha$, β, δ, γ, π, κ, x, and y.

Imagine that we have two clusters and let the current morphs be: c1: {*-i,-u*} and c2: {*-i,-ü*}. In order to compute $D(CurMor_{c_1} \parallel CurMor_{c_2})$, we use Equation 2 over each morph in the combination of two sets; c1+c2: {*-i,-u,-ü*}. We apply add-n smoothing to eliminate counts having a zero value in the vectors (e.g. the probability of *-u* would be zero for c_2 otherwise).

The algorithm starts with N morphs, each belonging to a distinct cluster. In each iteration, the two clusters with the minimum KL divergence are merged until all the morphs are merged in one cluster. The final cluster will be the root node in the hierarchical tree.

4 Experiments & Results

We used the gold standard analyses of words in Turkish and English for all of our experiments, which are provided by the Morpho Challenge (Mikko Kurimo, 2011). The word lists contain 552 English words and 783 Turkish words. Words are segmented and the morphemes are labelled in the gold standard, such that:

abacuses	*abacus*	N	PL
abstained	*abstain*	V	PAST

We modified the analyses manually, by replacing morpheme labels with actual morphs, such as:

abacuses	*abacus*	es
abstained	*abstain*	ed

As an input to the clustering algorithm, we extracted all morphs in the lists. The final lists contain 567 morphs in English and 1749 morphs in Turkish. We constructed the feature vectors of all morphs and applied the hierarchical clustering algorithm as described before. Once the trees were constructed, we cut the trees at different levels to retrieve the final clusters. Some resulting clusters in English are given in Table 1.

Morphemes	Words
-ism, -ion,	*hero**ism**, deduct**ion** etc.*
-ed, -ing	*insert**ed**, roof**ed**, leak**ed**, aris**ing**, puls**ing**, rat**ing** etc.*
-ness, -ity	*extensive**ness**, commun**ity**, earthi**ness** etc.*
-s	*town**s**man, yacht**s**, yacht**s**man etc.*
-er	*baby-sitt**er**s, plann**er**s, match-mak**er**s etc.*
-s'	*humanitie**s'**, protestant**s'**, swimmer**s'**, reduction**s'** etc.*

Table 1: Some morph clusters in English.

Since English is not a morphologically rich language, no homophonous morphemes or allomorphs could be captured. The reason for this is that morphs do not have sufficient contextual information. Nevertheless, morphs that show similar functional properties (i.e. tenses, derivative morphemes) are captured by the clustering algorithm. For example, both *-ism* and *-ion* are derivative morphemes that make the word a noun; *-ed* and *-ing* are inflectional morphemes that define the tense of a verb and *-ness* and *-ity* are derivative morphemes. There are many redundant clusters that have only one type of morpheme, such as plural morpheme *-s*, possessive morpheme *-s'* etc.

Experiments in Turkish provide a better understanding of what type of clusters are obtained from the clustering algorithm. Some resulting clusters in Turkish are given in Table 2. It is easier to see from the results that a good number of allomorphs are captured in Turkish due to the widely used vowel harmony. For example, allomorphs *-i* and *ı*; *-dır* and *-dir*, and *-nı* and *-ni* are captured. In addition to allomorphs, functionally similar morphemes *-a*, *-e*, *-i* and *-ı*, *-in* that refer to dative, accusative and genitive case respectively are also captured.

In order to evaluate our results, we again replaced the morphs in the gold standard with the obtained cluster labels, such that:

Morphemes	**Words**
-a, -e, -i, ı, -in	*faturaların**ı**, kongreler**i**, bilinmelerin**e**, bağışıklığ**ın**, mağazaların**a** etc.*
-dır, -dir	*almakta**dır**, ödeyecekler**dir**, değinilmeli**dir** etc.*
-let, -t	*iş**let**ecek, kuru**t**urken, uza**t**abilir etc.*
-lığ, -liğ, -yış	*başarısız**lığ**ı, başla**yış**ını, isteksiz**liğ**inin etc.*
-nı, -ni, -ne, -na	*bırakabileceğ**ini**, yakalandığ**ını**, düzeyler**ine**, mağazaları**na** etc.*

Table 2: Some morph clusters in Turkish.

commutation	Cluster50	*mutate*	+Cluster34
contradiction	*contradict*	+Cluster34	
decoded	Cluster50	*code*	+Cluster43
knifed	*knife*	+Cluster43	

Suffixes were inserted with a plus sign, whereas the other morphs were inserted with their labels. This provides a more comprehensive analysis on affixes and non-affixes separately.

We applied the evaluation method that Morpho Challenge (see Mikko Kurimo (2011)) follows. In the Morpho Challenge evaluation method, segmentations are evaluated through word pairs that have common morphemes. For example, in order to decide whether *book-s* is segmented correctly, another word in the results having the morph *-s* is found. Let's imagine we find *pen-s* in the results to make a word pair with *book-s*. In order to decide whether *book-s* is segmented correctly, we find the two words in the gold standard segmentations and check whether they really share a common morph. In that case, it does not matter whether the morphs or morph labels are used.

We tested our algorithm with different combinations of features. The results for Turkish by using the features, previous morph, following morph, current morph, stem and morph position are given in Table 3. The results consist of 162 clusters. The number of clusters is chosen in accordance with the highest evaluation score obtained.

Here, two types of analyses are presented: non-affixes and affixes. As mentioned above, the evaluation with non-affixes considers only non-affixes; whereas the evaluation with affixes considers the rest of the morphemes (i.e. stems and prefixes). Scores show that the algorithm is better at labelling suffixes than prefixes.

	Non-affixes	**Affixes**	**Total**
Precision	84.53	62.14	68.02
Recall	77.62	28.40	42.86
F-measure	80.93	38.98	52.58

Table 3: Evaluation results according to 162 clusters in Turkish by employing previous morph, following morph, current morph, stem and morph position as features.

	Non-affixes	**Affixes**	**Total**
Precision	87.15	57.45	65.04
Recall	79.51	31.76	45.79
F-measure	83.15	40.91	53.74

Table 4: Evaluation results according to 162 clusters in Turkish by employing previous morph, following morph, current morph, stem, morph position and morph length as features.

Results from another experiment that employs previous morph, following morph, current morph, stem, morph position and morph length are given in Table 4 for Turkish. The results are analysed according to the same number of clusters in order to investigate the impact of using different features. Here we can observe that using morph length as a feature improves the results.

The third experiment explores the impact of using the last morph of the previous word and the following word. The results of the experiment that uses previous morph, following morph, current morph, stem, the last morph of the previous word and the last morph of the following word are given in Table 5 for Turkish. The results show that using the last morph of the previous and following word does not improve the scores, but reduces contrarily.

All experiments that are presented above use equal weights for the features. We carried out another experiment by assigning weights to the features according to their importance. We set the weights manually, such that:

$$\begin{aligned} Sim(c_1, c_2) &= 0.3D(CurMor_{c_1} \parallel CurMor_{c_2}) \\ &+ 0.2D(PreMor_{c_1} \parallel PreMor_{c_2}) \\ &+ 0.2D(FolMor_{c_1} \parallel FolMor_{c_2}) \\ &+ 0.2D(Stem_{c_1} \parallel Stem_{c_2}) \\ &+ 0.1|pos_{c_1} - pos_{c_2}| \end{aligned} \quad (5)$$

The results of the weighted clustering algo-

	Non-affixes	Affixes	Total
Precision	87.93	46.95	61.06
Recall	73.05	12.03	29.96
F-measure	79.80	19.15	40.20

Table 5: Evaluation results according to 162 clusters in Turkish by employing previous morph, following morph, current morph, stem, morph position, the last morph of the previous word and following word as features.

	Non-affixes	Affixes	Total
Precision	93.82	69.64	80.23
Recall	86.34	44.08	74.41
F-measure	89.92	53.98	77.21

Table 6: Evaluation results by employing weighted features, which are previous morph, following morph, current morph, stem and morph position in Turkish.

rithm that employs the previous morph, following morph, current morph, stem and morph position are given in Table 6 for Turkish.

We also evaluated the algorithm for English by employing previous morph, following morph, current morph, stem, morph position and morph length as features. We obtained the results according to 100 clusters. The results are given in Table 7. In the experiment, the features were also weighted the same as the previous experiment.

5 Discussion

We tested the proposed clustering algorithm with various combinations of features. It should be noted that using previous and following morphs in English is not very beneficial due to the simple morphology of the language. However, we used these two features because of a number of words having more than one morph. Since Turkish is richer in morphology compared to English, previous and following morphs are more beneficial in clustering of Turkish morphs.

Another issue in Turkish morphology that needs to be noted that is the ambiguity of morphs. Words can be segmented in different ways depending on the meaning of the word, which can be discovered by looking at the context of the word. Hence, it also makes sense to employ the context of a morph in clustering. We employ the last morphs of the previous and following words to make use of the context in clustering. This makes a considerable amount of improvement in the results because Turkish grammar has noun phrases, subject-verb agreement etc.

	Non-affixes	Affixes	Total
Precision	95.60	90.72	92.93
Recall	84.79	34.46	70.59
F-measure	89.87	49.95	80.24

Table 7: Evaluation results according to 100 clusters in English by weighting features, which are previous morph, following morph, current morph, stem, morph position, the last morph of the previous word and the last morph of the following word.

In all experiments we manually assign weights to the features. Weighting features improves results since the features are not equally important in clustering. We leave the issue of estimating weights to be explored in the future.

6 Conclusion & Future Work

In this paper, an agglomerative hierarchical clustering algorithm is presented for labelling morphs. The algorithm aims to capture allomorphs and homophonous morphemes for a deeper analysis of morphological segmentation results. Most morphological segmentation systems focus only on segmentation, rather than labelling morphs. Nevertheless, it is helpful to label morphs in order to have an idea about the grammatical function of the word in a sentence; i.e. the syntactic category. We believe that a good morph labelling system will help PoS tagging, as well.

The presented algorithm can find allomorphs in Turkish by clustering them together. However, as far as we could observe from the results, it cannot show the same accuracy for homophonous morphemes.

We aim to improve the proposed approach by adopting mixture components for each morph label in a nonparametric Bayesian framework. We aim to handle the sparsity in the data with a nonparametric approach. Even with an infinite mixture model, it is possible to make the number of morph labels infinitely defined.

References

Delphine Bernhard, 2008. *Simple Morpheme Labelling in Unsupervised Morpheme Analysis*, pages 873–880. Springer-Verlag, Berlin, Heidelberg.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning - Volume 6*, MPL '02, pages 21–30, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2004. Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, SIGMorPhon '04, pages 43–51, Stroudsburg, PA, USA. Association for Computational Linguistics.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Sami Virpioja Ville Turunen Mikko Kurimo, Krista Lagus. 2011. Morpho challenge 2010. `http://research.ics.tkk.fi/events/morphochallenge2010/`, June.

Sebastian Spiegler. 2011. *Machine Learning For The Analysis Of Morphologically Complex Languages*. Ph.D. thesis, Merchant Venturers School of Engineering, University of Bristol, April.