Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 3253 – 3260

doi:10.3906/elk-1605-216

Unsupervised learning of allomorphs in Turkish

Burcu CAN*
Department of Computer Engineering, Faculty of Engineering, Hacettepe University, Ankara, Turkey

Received: 20.05.2016 • **Accepted/Published Online:** 30.12.2016 • **Final Version:** 30.07.2017

Abstract: One morpheme may have several surface forms that correspond to allomorphs. In English, *ed* and *d* are surface forms of the past tense morpheme, and *s*, *es*, and *ies* are surface forms of the plural or present tense morpheme. Turkish has a large number of allomorphs due to its morphophonemic processes. One morpheme can have tens of different surface forms in Turkish. This leads to a sparsity problem in natural language processing tasks in Turkish. Detection of allomorphs has not been studied much because of its difficulty. For example, *tü* and *di* are Turkish allomorphs (i.e. past tense morpheme), but all of their letters are different. This paper presents an unsupervised model to extract the allomorphs in Turkish. We are able to obtain an F-measure of 73.71% in the detection of allomorphs, and our model outperforms previous unsupervised models on morpheme clustering.

Key words: Natural language processing, morphology, allomorphs, clustering, unsupervised learning, nonparametric Bayesian learning

1. Introduction

Morphological segmentation is an essential task in many natural language processing (NLP) applications such as question answering, information retrieval, and sentiment analysis. Due to a large number of different word forms, a sparsity problem emerges for morphologically rich languages during such NLP tasks. For example, the number of word forms in Turkish is theoretically infinite because of heavy inflection and derivation during morphological generation. Hankamer [1] argued that listing every word form in an agglutinative language is impossible. Therefore, words are morphologically segmented into their smallest units, called morphemes. For example, the Turkish word *Türkçeleştiremediklerimizden* (which means 'it is the one we could not translate into Turkish') is split into the following morphemes: *Türkçe*, *leş*, *tir*, *e*, *me*, *dik*, *ler*, *imiz*, and *den.*

Many morphological segmentation systems [2–5] only split words into their surface morphs rather than finding lexical morphemes. Morphs are distinct realizations that belong to the same type of morpheme. For example, *s* and *es* are two different morphs belonging to the same morpheme type (i.e. plural/present tense) in English. However, a complete morphological analysis also requires finding the underlying realizations of morphs and their morphological tags (e.g., plural, past tense, case, and person).

To our knowledge, unsupervised work that provides morphological analysis with morphological tags does not exist. This process also requires finding allomorphs. There are rule-based systems that provide labeled morphological segmentation [6,7]. However, these are supervised and require manual annotation of all suffixes, roots, and morphotactic rules. Cotterell et al. [8] introduced a semi-Markov model for labeled morphological segmentation that is semisupervised.

*Correspondence: burcucan@cs.hacettepe.edu.tr

Labeled morphological segmentation, and thereby finding allomorphs, not only mitigates the sparsity in NLP tasks, but it is also essential for some NLP tasks. For example, in sentiment analysis it is essential to distinguish the negation morpheme from the noun derivation morpheme in Turkish. Both are written *ma* or *me*, depending on the orthographic features of the stem.

Allomorphs are very common in Turkish because of two morphophonemic processes, i.e. vowel harmony and consonant harmony. Different surface forms of the same morpheme are selected based on the vowels and consonants in the surrounding segments, thereby forcing all the letters (both consonants and vowels) to be harmonized with each other. This may lead up to 16 different allomorphs of the same morpheme in Turkish. For example, *cık*, *cik*, *cük*, *cuk*, *çık*, *çik*, *çuk*, *çük*, *cığ*, *ciğ*, *cuğ*, *cüğ*, *çığ*, *çiğ*, *çuğ*, and *çüğ* care are all allomorphs (i.e. derivational suffixes that give the meaning of 'small' to a noun; *kitap* means 'book', whereas *kitapçık* means 'brochure' or 'leaflet').

The detection of allomorphs has not been studied much in natural language processing. Spiegler [9] introduced two algorithms for morpheme labeling, which are both supervised. Virpioja et al. [10] extracted allomorphs by detecting the mutations (substitution or deletion) between morphs. However, Virpioja et al. [10] could only find 1.9% of the mutations in Turkish, which is a very small set of allomorphs in Turkish and is not sufficient for morphological analysis.

This article is organized as follows: Section 2 describes the phenomenon of allomorphs with Turkish examples, Section 3 describes the mathematical model and the algorithm for extracting allomorphs in Turkish, Section 4 presents the experimental results by comparing them with other models and, finally, Section 5 concludes the paper along with mention of potential future work.

2. The model and algorithm

In our model, we cluster allomorphs by using the distributional similarities of their neighborhoods. For example, the morphs that follow the allomorphs *ir*, *ur*, *ür*, and *ır* are very similar when they are devowelized (Table 1), or the previous morphs of the allomorphs *lik* and *liğ* are very similar when they are devowelized (Table 2). We utilize both the following and previous morph distributions of allomorphs in order to cluster them.

Table 1. The following morphs of the allomorphs *ir*, *ır*, *ür*, and *ur*.

ir	d(i), l(e)r, s(e), m(i)ş, etc.
ır	d(ı), l(a)r, s(a), m(ı)ş, etc.
ur	d(u), l(a)r, s(a), m(u)ş, etc.
ür	d(ü), l(e)r, s(e), m(ü)ş, etc.

Table 2. Previous morphs of the allomorphs lik and liğ.

lik	c(i), l(i), (i)ş, s(i)z, (i)c(i), etc.
liğ	c(i), l(i), (i)c(i), etc.

2.1. Clustering vowel allophones

In order to cluster vowel allophones, we exploit the distribution of the following morphs that are devowelized (e.g., the devowelized form of *lar* is *l_r* and *ler* is *l_r*). We define a multinomial-Dirichlet distribution over

the devowelized following morphs. The multinomial distribution is defined on a set of devowelized morphs $\mathbf{M_{fol}}=\{\mathbf{m_1 m_2},\ldots,\mathbf{m_N}$ with parameters θ whereas prior probability is defined for the parameters of the multinomial distribution in a Dirichlet distribution form with hyperparameters β:

$$m_i|\theta \sim Multinomial(\theta) \tag{1}$$

$$\theta|\beta \sim Dirichlet(\beta) \tag{2}$$

The definition of the Dirichlet distribution follows the form:

$$\frac{1}{B(\beta)}\prod_{k=1}^{K}\theta^{\beta_k-1} \tag{3}$$

where $\mathbf{B}(\beta)$ is a normalizing constant in beta function form, and $\mathbf{K}$ represents the number of allomorph clusters.

The multinomial distribution is defined over the outcomes $\mathbf{M_{fol}}=\{\mathbf{m_1 m_2}\ldots,\mathbf{m_k}\}$ as follows:

$$p(M_{fol}|\theta)=\frac{N!}{\prod_{k=1}^{K} n\left(m_k\right)!}\prod_{k=1}^{K}\theta_k^{n(m_k)} \tag{4}$$

where $\mathbf{N}$ denotes the total number of morph tokens belonging to one of the possible allomorph clusters; that is, $\mathbf{N}=\prod_{\mathbf{k=1}}^{\mathbf{K}}\mathbf{n}\left(\mathbf{m_k}\right)$, where $\mathbf{n}\left(\mathbf{m_k}\right)$ is the number of morph tokens that are allomorphs with $\mathbf{m_k}$.

The first factor in Eq. (4) provides the exchangeability over the morph tokens; the second factor computes the probability of observing each morph token. We integrate out θ to obtain the joint distribution over all morphs: $\mathbf{M_{fol}}=\{\mathbf{m_1 m_2},\ldots,\mathbf{m_N}\}$:

$$p(M_{fol}|\beta)=\frac{\Gamma(B)}{\Gamma(N+B)}\prod_{i=1}^{K}\frac{\Gamma(n\left(m_k\right)+\beta_k)}{\Gamma(\beta_k)} \tag{5}$$

where $\mathbf{B}=\sum_{\mathbf{k}}\beta_{\mathbf{k}}$. We use symmetric hyperparameters for the clusters because morph clusters are a priori uniform.

We only model allomorph clusters based on vowel allophones here, such as *lar* and *ler* (*a* and *e* are allophones); *tır*, *tir*, *tur*, and *tür* (ı, i, u, and ü are allophones); and so on. In other words, we extract allomorphs that differ from each other. Noting that *a* and *e* are allophones, all morphs that involve *a* or *e* are forced to be allomorphs (*lar* and *ler*, *dan* and *den*, *tan* and *ten*, and *sa* and *se* become allomorphs).

2.2. Clustering consonant allophones

In order to cluster allomorphs further that differ from each other based on consonants such as *lik* and *liğ* or *cik* and *ciğ*, we use previous morphs that are devowelized. The following morphs do not give enough information on this type of allomorphs because the last consonants of morphs may change depending on the following morphs because of consonant harmony. For example, *lik* and *liğ* have completely different following morphs. *k* transforms into *ğ* because of the following morph. However, the previous morphs of allomorphs with consonant allophones are similar to each other (Table 2).

This leads us to use the similarity in the devowelized previous morphs in order to discover the allomorphs that involve consonant allophones. We analogously define a multinomial-Dirichlet distribution for clustering allomorphs with consonant allophones. Let the devowelized previous morphemes be M_{pre}=$\{s_1 s_2, \ldots, s_t\}$. The multinomial-Dirichlet distribution is then defined for the previous morphs as follows:

$$s_i|\gamma \sim \ Multinomial(\gamma) \tag{6}$$

$$\gamma|\alpha \sim \ Dirichlet(\alpha) \tag{7}$$

with multinomial parameters γ and Dirichlet hyperparameters α. The joint distribution over the devowelized previous morphs of each morph is defined as follows:

$$p(M_{pre}|\alpha) = \frac{\Gamma(A)}{\Gamma(T+A)} \prod_{i=1}^{K} \frac{\Gamma(n\,(s_k)+\alpha_k)}{\Gamma(\alpha_k)} \tag{8}$$

where K is the number of allomorph clusters, T is the total frequency of allomorphs, and $A = \sum_k \alpha_k$. We also use symmetric hyperparameters for each allomorph cluster here.

3. Algorithm

The clustering algorithm involves two steps:

a. clustering allomorphs based on vowel allophones,

b. clustering allomorphs based on consonant allophones by using the allomorph clusters obtained in Step a.

In both steps, we use the Metropolis–Hastings algorithm [11] for the inference. In order to cluster vowel allophones, we begin with one cluster for each single vowel. We gradually replace the vowels either by creating a new cluster or inserting the selected vowel in one of the existing clusters uniformly. According to this replacement, we determine which morphs will gather in the same allomorph cluster and which will fall into different clusters. For example, if a and e are in the same allophone cluster, all morphs with a and e (provided that all other characters in the morphs are either the same or also allophones of each other) are grouped in the same allomorph cluster. We postulate that all morph tokens of the same type will be in the same cluster. Homophonous morphs are not in the scope of this paper. We either accept or reject the new clustering with the given probability:

$$P_{Acc}=\frac{P_{new}}{P_{old}} \tag{9}$$

where P_{new} and P_{old} are the new and old joint probabilities (see Eq. (5)). If $P_{Acc} \geq 1$ we accept the new sample. We still accept the new sample with P_{Acc} to randomize the search in order to find the global maximum.

We cluster the consonants and the corresponding allomorph clusters based on the clusters that are found in the previous step analogously by again applying the Metropolis–Hastings algorithm. We use the joint probability for consonants, given in Eq. (8).

4. Experiments and results

4.1. Data

We use the Turkish word list provided by Morpho Challenge 2010 [12], which consists of 617,298 words that are not morphologically segmented and only involves the frequency of each word. Morphological segmentation of words is assumed to be known a priori in our model. We use an open source morphological analyzer, Zemberek [13], to parse the word list into its morphs. We discard the stems that are not seen with any suffix in the corpus. Therefore, our final word list involves 604,091 words, 813,832 morph tokens, and 369 morph types.

Simulated annealing is applied with an initial temperature $t = 2.0$, and the system is cooled down to 0.1 with decrements of 0.1 in each iteration. The settings are the same for both inference steps (learning vowel and consonant allophones) in the algorithm. We did several experiments for various values of α and β in order to empirically set their values.

4.2. Results

Some of the allomorph clusters obtained from the model are given in Table 3. There are not any clusters in the results that consist of morphs that are not allomorphs of each other. Some of the allomorphs are instead scattered over two different clusters, rather than gathering into one cluster. For example, *ti*, *tu*, *tı*, *tü* and *di*, *du*, *dı*, and *dü* fall into two different clusters, whereas they must be gathered in a single cluster. Many of the allophones are correctly found. For example, $\{a, e\}$, $\{ğ,\ k\}$, and $\{ı,\ i,\ u,\ ü\}$ are correctly found, and they are the most common allophones in Turkish. Therefore, all the corresponding allomorphs that involve these allophones are correctly learned.

Table 3. Some of the allomorph clusters.

iyor, ıyor, üyor, uyor
ici, ücü, ucu, ıcı
ımız, ümüz, imiz, umuz
cik, cuğ, cık, cük, cüğ
luk, lık, liğ, lüğ, lığ, lik, lük, luğ
dükçe, dikçe, dıkça, dukça
tüğ, tuk, tik, tuğ, tığ, tiğ, tük, tık
dığ, düğ, dık, diğ, dük, duk, duğ, dik

In order to evaluate the allomorph clusters, we use the purity and F-measure. We compute purity as follows:

$$Purity = \frac{1}{N} \sum_{i} \max |c_i \cap t_i| \tag{10}$$

where N is the total number of morphs, c_i is the number of morphs in result cluster c_i, and t_i is the number of morphs in gold cluster t_i. We obtain a purity of 0.84% when $\alpha = \beta = 0.0001$.

In order to calculate the F-measure, we use the same evaluation method used by Morpho Challenge [12]. In the Morpho Challenge evaluation method, word pairs are compared to check whether they share a common segment or not. For example, in order to evaluate the segmentation *year + s*, another word that involves *- s* is found in the results. Whether two words share a common segment in the gold segmentations is checked. To apply the same evaluation method, each allomorph cluster is given a unique ID and morphs are replaced with their cluster IDs in the wordlist. Since the Morpho Challenge evaluation method computes the scores based

on common morphs between word pairs, it does not make a difference if there are actual morphs or tags. For example, the gold analyses of *üniversite + ler + in* (of the universities) and *iste + se + ler* (if they want) are as follows in the Morpho Challenge gold analyses:

üniversitelerin	üniversite	+ PL	+ GEN
isteseler	iste	+TNS_sa	+PER3P

Although the morpheme *ler* has the same surface form in both words, the labels are different (PL and PER3P) since the first one is a plural morpheme and the second morpheme refers to the third person plural. We replace the Zemberek-segmented words here with their cluster IDs in order to calculate the F-measure:

erteledi Gimiz	ertele	Cluster1	Cluster42
kongreleri	kongre	Cluster26	Cluster13

We obtain an F-measure of 73.71% when $\alpha = \beta = 0.0001$. All results for different values of α and β are given in Table 4.

Table 4. Purity and F-measure scores of clusters obtained from the multinomial-Dirichlet model for different values of α and β.

α, β	Tag size	Purity	F-Measure (%)
0.5, 0.5	301	0.42	64.77
0.1, 0.1	307	0.59	70.79
0.01, 0.01	168	0.58	69.99
0.001, 0.001	122	0.63	72.92
0.0001, 0.0001	130	0.84	73.71

There not many studies on morpheme tag classification. We compare our model with the agglomerative hierarchical clustering algorithm by Can and Manandhar [14] and the semi-Markov model by Cotterell et al. [8]. The agglomerative hierarchical clustering algorithm of Can and Manandhar [14] clusters morphs according to their meanings within words, thereby finding the allomorphs and homophonous morphemes. The semi-Markov model does labeled morphological segmentation that assigns labels to each morph based on different granularity levels (from a coarse-grained level that only has prefix, root, and suffix to a fine-grained level that involves case, person, etc.). The results are given in Table 5. However, it should be noted that the semi-Markov model used by Cotterell et al. is semisupervised whereas the other two models are unsupervised. Our model outperforms the agglomerative hierarchical clustering model.

Table 5. F-measure scores of the multinomial-Dirichlet model, the agglomerative hierarchical clustering algorithm [14], and the semi-Markov model [8].

Model	Tag size	F-measure (%)
Semi-Markov model [8]	50	85.07
Multinomial-Dirichlet	130	73.71
Agglomerative clustering [14]	162	53.74

For an extrinsic evaluation, we use our final allomorphs for the morphological generation task. We build finite state automata by assigning each state an allomorph cluster (Figure). States are linked to each other if

any morph between two allomorph clusters are seen together within the same word in the corpus. The same also applies for links between stems and allomorphs. We used the first 5000 sentences in the text corpus provided by Morpho Challenge 2009 [12] that involve 3049 unique words. We generated 237,219 words in total. Zemberek is used for checking whether a generated word form is a valid Turkish word form or not. We obtained an accuracy rate of 76.79% for this task.

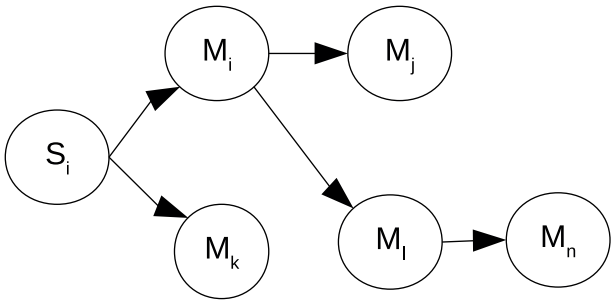

Figure. A finite state automaton where S_i corresponds to a stem category and M_i, M_j, M_k, and M_l correspond to allomorph clusters.

5. Conclusion and future work

Allomorphs are very common in Turkish and lead to sparsity in natural language processing tasks. This has been one of the prominent problems in Turkish natural language processing because of the sparsity that it introduces in any natural language processing task. Finding allomorphs will be beneficial for natural language processing tasks by reducing sparsity. Therefore, allomorphs can be treated similarly in such tasks instead of treating a single morpheme on its own.

We introduce an unsupervised method for clustering allomorphs in Turkish. Our method is Bayesian and exploits the contextual distributions of morphemes in order to capture the distributional similarity between allomorphs. Multinomial-Dirichlet distribution is used for modeling the contextual distributions in this article. Results show that our model outperforms previous unsupervised work based on agglomerative hierarchical clustering [14]. Extrinsic evaluation scores obtained from morphological generation also show that the generation task performs with 76.79% accuracy by using the allomorphs obtained from our model. Hence, the model can capture allomorphs well in an unsupervised framework.

When the final allomorph clusters are investigated, it can be observed that allomorphs that diverge from each other in the last consonant (e.g., *k* vs. *ğ*) or any vowels in the middle can be captured by the model. However, allomorphs that diverge from each other in the first consonant (e.g., *c* and *ç*) cannot be captured by the model due to consonant mutation.

In this article, we do not learn homophonous morphemes that are phonologically the same but different in meaning (such as plural *s* and present tense *s*). Therefore, one of the drawbacks of the model is that it assumes that all morpheme tokens belonging to the same morpheme type are assigned only one allomorph cluster. Finding homophonous morphs will be kept for future work.

References

[1] Hankamer J. Morphological parsing and the lexicon. In: Marslen-Wilson WD, editor. Lexical Representation and Process. Cambridge, MA, USA: MIT Press; 1989. pp. 392-408.

[2] Goldsmith J. Unsupervised learning of the morphology of a natural language. Comput Linguist 2001; 27: 153-198.

[3] Creutz M, Lagus M. Unsupervised discovery of morphemes. In: Workshop on Morphological and Phonological Learning; 2002; Morristown, NJ, USA. pp. 21-30.

[4] Can B, Manandhar S. Probabilistic hierarchical clustering of morphological paradigms. In: 13th Conference of the European Chapter of the Association for Computational Linguistics; 23–27 April 2012; Avignon, France. pp. 654-663.

[5] Poon H, Cherry C, Toutanova K. Unsupervised morphological segmentation with log-linear models. In: Annual Conference of the North American Chapter of the Association for Computational Linguistics; 31 May–5 June 2009; Boulder, CO, USA. pp. 209-217.

[6] Eryiğit G, Adalı E. An affix stripping morphological analyzer for Turkish. In: International Conference on Artificial Intelligence and Applications; 16–18 February 2004; Innsbruck, Austria. pp. 299-304.

[7] Çöltekin Ç. A freely available morphological analyzer for Turkish. In: 7th International conference on Language Resources and Evaluation; 17–23 May, 2010; Valetta, Malta. pp. 820-827.

[8] Cotterell R, Müller T, Fraser AM, Schütze H. Labeled morphological segmentation with semi-Markov models. In: 19th Conference on Computational Natural Language Learning; 30–31 July 2015; Beijing, China. pp. 164-174.

[9] Spiegler S. Machine learning for the analysis of morphologically complex languages. PhD, University of Bristol, Bristol, UK, 2011.

[10] Virpioja S, Kohonen O, Lagus K. Unsupervised morpheme analysis with Allomorfessor. In: Peters C, Di Nunzio GM, Kurimo M, Mandl T, Mostefa D, Penas A, Roda G, editors. Multilingual Information Access Evaluation I - Text Retrieval Experiments. Berlin, Germany: Springer; 2010. pp. 609-616.

[11] Hastings WK. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 1970; 57: 97-109.

[12] Kurimo M, Virpioja S, Turunen V. Morpho Challenge 2009. Available online at http://research.ics.aalto.fi/events/morphochallenge2009/.

[13] Akın AA, Akın MD. Zemberek, an open source NLP framework for Turkic languages. Structure 2007; 10: 1-5.

[14] Can B, Manandhar S. An agglomerative hierarchical clustering algorithm for labelling morphs. In: Recent Advances in Natural Language Processing; 7–13 September 2013; Hissar, Bulgaria. pp. 129-135.