Turkish Journal of Electrical Engineering & Computer Sciences

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

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 1012 – 1027

doi:10.3906/elk-1804-10

Turkish lexicon expansion by using finite state automata

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

Received: 03.04.2018 • **Accepted/Published Online:** 10.12.2018 • **Final Version:** 22.03.2019

Abstract: Turkish is an agglutinative language with rich morphology. A Turkish verb can have thousands of different word forms. Therefore, sparsity becomes an issue in many Turkish natural language processing (NLP) applications. This article presents a model for Turkish lexicon expansion. We aimed to expand the lexicon by using a morphological segmentation system by reversing the segmentation task into a generation task. Our model uses finite-state automata (FSA) to incorporate orthographic features and morphotactic rules. We extracted orthographic features by capturing phonological operations that are applied to words whenever a suffix is added. Each FSA state corresponds to either a stem or a suffix category. Stems are clustered based on their parts-of-speech (i.e. noun, verb, or adjective) and suffixes are clustered based on their allomorphic features. We generated approximately 1 million word forms by using only a few thousand Turkish stems with an accuracy of 82.36%, which will help to reduce the out-of-vocabulary size in other NLP applications. Although our experiments are performed on Turkish language, the same model is also applicable to other agglutinative languages such as Hungarian and Finnish.

Key words: Morphology, lexicon expansion, morphological generation, finite-state automata

1. Introduction

Morphological segmentation is the task of segmenting a word into its smallest meaning-bearing units called morphemes, whereas morphological generation is the reverse task of morphological segmentation, which is the task of generating various word forms from a given stem. For example, the word *disgraceful* can be split into *dis*, *grace*, and *ful*, which is called morphological segmentation[1] and morphological generation is the task of generating *disgrace*, *disgraceful*, *disgracefully*, *ungraceful*, *ungracefully* and many others by using the stem *grace*. However, it is also the task of not generating invalid word forms such as *gracelyful* or *disungraceful*. This brings the challenge in morphological generation. Thus, morphological generation (i.e. word generation, lexicon expansion)[2] is usually inseparable from morphological segmentation, although the tasks are reverse of each other.

In this article, we mainly investigate morphological generation, which plays a significant role in many natural language applications such as machine translation, question answering, language generation, dialog systems. For example, translation of a phrase with a rich morphology requires the right word form to be

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

[1]The task is called morphological analysis when morphosyntactic labels of each morpheme are also learned. For example, the morphological analysis of the word *tables* is *table[Noun]+s[A3Pl]* that tells *table* is a noun and *s* is a suffix that makes the noun plural. Here, we only consider morphological segmentation. Other than that, given a morphological analysis, the task of generating the correct word form is called *morphological reinflection*.

[2]We use morphological generation, lexicon expansion, and word generation interchangeably in this article.

generated for a specific sentence. This is only possible through morphological generation because it is impossible to build a language model with an infinite dictionary that includes all possible word forms in that language. The task is even more implausible for the agglutinative languages. Instead of assuming a fixed-size dictionary in any model, the correct word form should be generated when needed.

Morphological generation in Turkish involves orthography and morphotactics. Orthography corresponds to changes in morphemes due to the spelling rules, where phonological operations applied while morphemes are glued together. Morphotactics is concerned with how and in what order morphemes are glued together in order to generate new word forms.

One of the methods for morphological segmentation/generation is the finite-state transducers (FSTs). FSTs are good at handling both morphotactics and orthography. Because of this, they have been numerously applied to morphological analysis. Supervised methods have usually been applied to morphological generation. However, supervised learning requires a large amount of annotated data. Although Turkish has a regular morphology, all the orthographic features and the morphological rules that correspond to morphotactics have to be provided. Moreover, language is dynamic, and persistent need of update on the annotated data is a limitation. For these reasons, in this article, we introduce a system based on minimally supervised learning for morphological generation by using only morphologically segmented data, and no other information. Our model only uses the output of any segmentation system (i.e. segmented data) and does not involve a segmentation mechanism itself unlike the other morphological analyzer/generators. Therefore, the only goal is morphological generation. In the first setting, we use gold segmentations that are produced by a supervised morphological segmentation system. However, the results show that our method works also well without using any gold segmentation but only using the output of an unsupervised morphological segmentation system in a fully unsupervised setting.

2. Related work

Turkish is an agglutinative language with a very rich but also regular morphology [1, 2]. The language is based on both rich inflectional and derivational morphology.

Turkish morphological analysis/segmentation has been well studied. Most of the earlier work uses finite-state transducers (FSTs) for Turkish morphological analysis in a supervised setting. When the two-level morphology was proposed by Koskenniemi [3], it was adapted for the Turkish language by Oflazer [4]. The setting was supervised and required all the morphotactics and orthographic rules to be involved manually. Sak, Güngör, and Saraçlar [5] exploited the two-level rules of Oflazer [4] in a stochastic finite-state morphological parser. Both methods [4, 5] involve morphological generation implicitly (through the inverse of the FSTs), although the methods do only aim for morphological analysis and not generation. Eryiğit and Adalı [6] proposed a morphological analyzer for Turkish. Morphotactic rules are used to define the finite-state machines. Similar to ours, suffixes are grouped into several classes. For each suffix class, a finite-state machine is defined that describes the concatenation rules of the suffixes. Çöltekin [7] introduced a Turkish morphological analyzer that is also two-level similar to that of Oflazer [4]. Zemberek [8] is another morphological analyzer tool, which is also based on finite-state automata where the morphotactic and orthographic rules are defined manually.

Morphological generation has been well studied on Arabic language. Arabic is a Semitic language with a nonconcatenative morphology that makes morphological analysis and generation challenging. In Arabic, stems are generated with a root and a vowel melody, which forms a pattern [9, 10]. For example, the Arabic stem *katab* (means *he wrote*) is formed by the stem *ktb* and the vowel melody *a-a* using the pattern CVCVCV (C:

consonant, V: vowel) [10]. Habash [11] reverses the Buckwalter Arabic morphological analyzer [12] by using its database and converting it into a large-scale lexeme-based Arabic morphological generator. However, morpheme features, such as number, gender, and case inflection, are given as input to the system in a supervised setting. Another study on Arabic is by again Rasooli et al. [13]; it presents a morphological analyzer and generator for Arabic dialects. The authors combine morphemes from different dialects with separate morphological and orthographic representations. They implement the multitape approach of Kiraz [14] with an additional tape for phonology and orthography that builds the system called *MAGEAD*.

All the works mentioned above are supervised. Rasooli et al. [15] introduced an unsupervised vocabulary generator for low-resource languages in order to reduce the out-of-vocabulary word rate in natural language processing applications, such as automatic speech recognition or optical character recognition. The model was tested on seven languages: Assamese, Bengali, Pashto, Persian, Tagalog, Turkish, and Zulu. Rasooli et al. [15] used Morfessor CatMAP [16] for unsupervised morphological segmentation to obtain the morphemes as input into their system. They implemented weighted finite-state transducers (WFST) for the lexicon expansion model using the learned morphemes. Finally, the generated word forms are reranked to filter overgenerated word forms by using different techniques, such as reweighting the WFSTs with a letter trigraph language model or instead of reweighting the WFSTs, by getting the n-best words generated by the WFSTs by reranking them according to the trigraph probabilities. For example, the Turkish word *çiçeklridn* (the true word form is *çiçeklerinden*) will have a low generation probability since the probability of seeing trigraphs *klr* and *idn* will be low.

Our work is mostly similar to that of Rasooli et al. [15]. We developed our model both in minimally supervised and unsupervised settings. We also used morpheme categories similar to Köprü and Miller [17], and also Eryiğit and Adalı [6]. We were motivated by both for clustering the stems and suffixes for generalizing the rules in order to apply similar rules to stems that are syntactically similar.

We induced morphotactics and orthographic rules by using finite-state automata (FSAs) to build a morphological generator. Although morphological analyzer FSTs can be converted into morphological generators, to our knowledge, there is no specific work that focuses only on Turkish morphological generation by aiming to find all possible word forms as a lexicon expansion problem in a minimally supervised or unsupervised learning framework apart from the model proposed by Rasooli et al. [15]. However, our algorithm deviates from their algorithm since we cluster both stems and suffixes to build the FSAs, whereas they learn the language model using the actual stems and suffixes without using their classes. Additionally, Tantuğ and Eryiğit [18] proposed probabilistic word root generation in Turkish by using the bigram letter probabilities. However, their method only aims for generating roots and does not focus on inflectional or derivational morphology.

3. Allomorphs

Some morphs carry out the same function corresponding to the same meaning although they consist of different letters and they are pronounced differently. These morphs are called *allomorphs*. Plural morphs *-s* and *-ies* are examples to allomorphs in English.

Allomorphs are very common in some languages due to vowel harmony. Vowel harmony is the adaptation of vowels in a word. Turkish is a language that applies vowel harmony intensely leading to severe allomorphy in the language. For example, the accusative case of a word can be seen in different forms due to vowel harmony: [*ı*], [*i*], [*u*], [*ü*]. Depending on the last vowel in the word, one of the morphemes in the allomorph set is selected. The examples are given as follows:

Example 3.1

ı	*baraj*	*dam*	-	*baraj-ı*	*the dam*
i	*kent*	*town*	-	*kent-i*	*the town*
u	*koyun*	*sheep*	-	*koyun-u*	*the sheep*
ü	*üzüm*	*grape*	-	*üzüm-ü*	*the grape*

Vowel harmony is not the only phonological process in Turkish. Unvoiced consonants (i.e. *p*, *ç*, *t*, *k*) at the end of words mutate into voiced consonants (i.e. *p:b*, ç:*c*, t:*d*, k:*ğ*) if the first letter of the following morpheme is a vowel. This process is called consonant mutation. The examples are given below:

Example 3.2

The accusative case: kitap-lık-ı becomes kitap-lığ-ı in the surface form (means the bookcase)
The accusative case: ağaç-ı becomes ağac-ı in the surface form (means the tree)
The dative case: şarap-a becomes şarab-a in the surface form (means to the wine)
The dative case: kağıt-a becomes kağıd-a in the surface form (means to the paper)

Consonant mutation and consonant harmony also produce allomorphs. For example, *dır* in *para-dır* (means *it is money*) becomes *tır* in *kağıt-tır* (means *it is a paper*). Thus, *dır* and *tır* become allomorphs of each other. Clustering allomorphs can be a good way to mitigate sparsity. Indeed, some Turkish morphological analyzers use the capital letter notation for allomorphs. For example, *A* may correspond to *a* or *e* in any given word in Oflazer, Göçmen and Bozsahin [19] which is a convention to represent allomorphs (e.g. *lAr* for *lar* and *ler*; *DA* for *da*, *de*, *ta*, and *te*). We also use this notation in one of our experiments to mitigate sparsity (see Section 4).

4. The lexicon expansion model and the algorithm

4.1. Data

We used the raw Turkish text corpus provided by Morpho Challenge 2009[3] as the training data. The text corpus consists of 1 million sentences. We used the first 5000 sentences from this corpus in order to generate various word forms. The first 5000 sentences involve 5630 unique words.

We obtained the morphological segmentation of words from an open-source morphological analyzer called *Zemberek* [8]; this is regarded as the gold segmentations. Stems with no suffix in the corpus were discarded in order to reduce the noise. Eventually, we obtained 3049 unique stems for training. Since the task is lexicon expansion (in a minimally supervised setting), one of the goals is to use a small training set in order to generalize the set as much as possible by generating all possible word forms.

4.2. Finding stem categories

We cluster stems so that stems in the same category are treated likewise when selecting a suffix to attach to the end of the stem in order to generate new word forms. We use the distributional characteristics of stems for clustering; i.e. two stems are categorically similar if they are attached with similar suffixes [20]. We used the similarity metric given by Baek et al. [21]:

$$Sim(s_1; s_2) = \frac{\sum_{m_i \in M_1 \cup M_2} min(MI(s_1, m_i), MI(s_2, m_i))}{\sum_{m_i \in M_1 \cup M_2} max(MI(s_1, m_i), MI(s_2, m_i))} \quad (1)$$

[3] http://research.ics.aalto.fi/events/morphochallenge2009/datasets.shtml/

where s_1 and s_2 are the two stems. M_1 is the suffix set with which s_1 is seen in the corpus and M_2 is the suffix set with which s_2 is seen in the corpus. MI is the mutual information between the given stem and suffix, which is a measure that tells the mutual dependence between two variables. In other words, if the mutual information between two random variables is high, it means that there is a natural dependence between these two variables. Here, how similar two stems are is measured based on the mutual information of the two stems with the other morphemes. Thus, the similarity metric measures the similarity between two mutual information values. For example, there would be a high mutual information between *walk* and *-ed, -ing, -s*, the same also applies for *order* and *-ed, -ing, -s*. Therefore, the two stems would be similar based on this suffix set.

We implemented an agglomerative hierarchical clustering algorithm. We gradually merged the cluster pairs having the highest similarity based on Eq. 1. Merging was repeated until the similarity between cluster pairs is above 0. Eventually, we obtained 152 stem clusters.

We merged the stem clusters further to reduce the number of clusters to have stem categories as compact as possible. Here, the ideal situation is to have only three categories: noun, adjective, and verb. We used the Jensen–Shannon divergence that is the symmetric version of Kullback–Leibler (KL) divergence [22]:

$$JSD(S_1||S_2) = \frac{1}{2}D_{KL}(S_1||S_2) + \frac{1}{2}D_{KL}(S_2||S_1) \tag{2}$$

The KL divergence between the two stem clusters S_1 and S_2 is computed as follows:

$$D_{KL}(S_1||S_2) = \sum_{m_i \in M} p_{S_1}(m_i)\frac{p_{S_1}(m_i)}{p_{S_2}(m_i)}. \tag{3}$$

Here, $M = M_1 \cup M_2$, where M_1 is the suffix set that the stems in S_1 are seen with and M_2 is the suffix set that the stems in S_2 are seen with. $p_{S_1}(m_i)$ is the probability of the suffix m_i being attached to stems in S_1. Laplace smoothing is applied for the zero probabilities with a smoothing parameter 10^{-4}. We assigned the value of the smoothing parameter empirically as a result of several experiments.

Finally, we obtained 89 stem clusters. Some of the clusters are given in Table 1. Verb, noun, and adjective stems fall into different clusters. For example, Cluster 4 consists of only verb stems, whereas Cluster 1 and Cluster 3 consist of only noun stems, and Cluster 5 consists of adjective stems. However, stems can diverge due to allomorphy. For example, noun stems that can take different realizations of the plural morpheme (i.e. *lar* and *ler*) diverge from each other (e.g. miras vs küre).

Table 1. Examples from some of the final stem categories.

Cluster 1: bilim, fizik, küre, cin, gelenek, birey , evren, istatistik, matematik, zihin, beden
Cluster 2: görüş, yürüt, öngör, öldür, sök, bük, güt, büyült, öv, götür, sürdür, düşür
Cluster 3: kaçak, ırk, dans, kalıp, sanat, miras, balık
Cluster 4: kur, dur, sor, uy, kor, yor, ok, otur, kaybol, sok, koştur, kop, uyuştur, vur, korkut, kavur, dokun, duyur, durdur, duy, doldur, boz, unut, bulundur, kon, tut, bulun, sun
Cluster 5: uygun, boş, bozuk, uzun, mutlu, yoğun, memnun, buz, imparator, durgun, yolcu, tutuk, yolsuz, olumsuz

4.3. Clustering allomorphs

We clustered allomorphs in two steps based on the algorithm proposed by Can [23]. In the first step, we clustered allomorphs that diverge from each other with a vowel (e.g., *da* and *de*), in the second step we clustered allomorphs that diverge with a consonant (e.g., *ta* and *da*). Here, we call these letters (that vary in the allomorphs) *allophones*. Some of the final allomorph clusters are given in Table 2.

Table 2. Some of the resulting allomorph clusters.

ler, lar
cik, cIğ, cık, cük, cüğ
luk, lık, liğ, lüğ, lığ, lik, lük, luğ
dükçe, dikçe, dıkça, dukça
tüğ, tuk, tik, tuğ, tığ, tiğ, tük, tık
dIğ, düğ, dık, diğ, dük, duk, duğ, dik

4.4. Building FSA

We build FSA with the stem and allomorph clusters obtained in the previous steps. Each state in an FSA corresponds to either a stem category S_i or a suffix category M_i (where $M = (C \cup V)*$; C for consonat, V for vowel). We build one FSA for each stem category by assigning the stem state as a start state. Each stem state is linked to various suffix states if one of the stems in that category is seen with any other suffix in the suffix state. A finite-state automaton is defined by a 5-tuple $X = (A, I, f, S_0, F)$ as follows (see Figure 1):

- A is the set of states, where A belongs to either a stem category S_i or a suffix category M_i (i.e. allomorph cluster).
- I is the alphabet, where $I = M \cup S$ (i.e. M: all suffixes, S: all stems)
- f is a transition function that links a state S_i to another state M_j where $\exists w' = s_i + m_j + \ldots, w' \in W, s_i \in S_i, m_j \in M_j$, or links M_i to another state M_j where $\exists w' = s_i + m_i + m_j + \ldots, w' \in W, m_i \in M_i, m_j \in M_j$, where W denotes the corpus.
- S_0 is an item of A, which is called the start state, where $\exists w' = s_0 + \ldots, w' \in W, s_0 \in S_0$.
- F is a subset of A that are the final states, where F is either a stem category S_i or a suffix category M_i and $\exists w' = s_i, w' \in W$ or $\exists w' = s_i + m_i, w' \in W$.

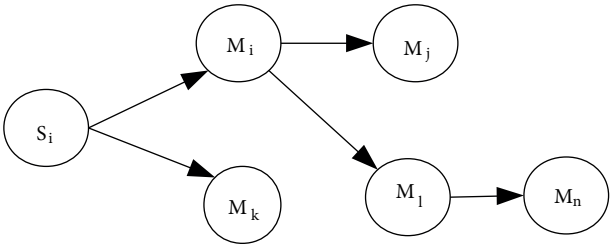

Figure 1. An FSA where S_i corresponds to a stem category and M_i, M_j, M_k, and M_l correspond to suffix categories (i.e. each of them is an allomorph cluster).

A portion of a noun FSA is given in Figure 2 and a portion of a verb FSA is given in Figure 3. We obtain 89 FSA in total for verbs and nouns that corresponds to 89 stem categories.

4.5. Word generation through the FSA

Words were generated through the FSA with a depth-first search. Since each state consists of a set of morphemes, we followed a selection methodology based on the letter bigrams in order to decide which morpheme to adopt from each state for generating the current word. Orthographic rules for the phonological operations were also generated for applying any consonant change between the stem and the selected morpheme.

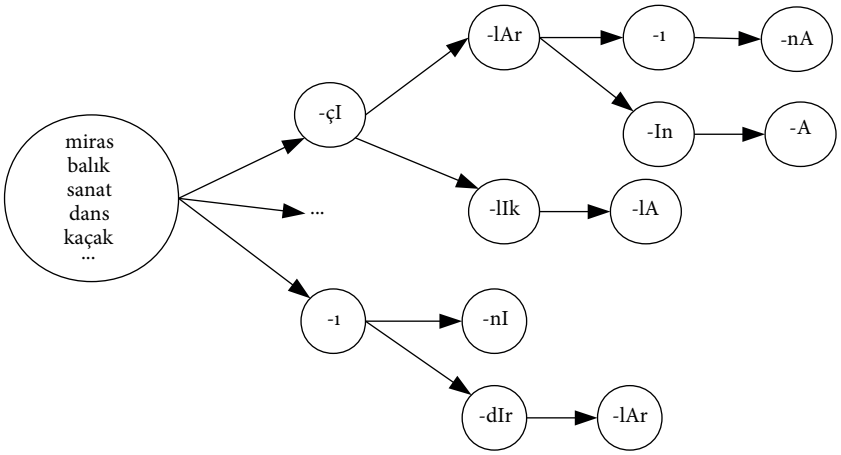

Figure 2. A portion of a noun FSA. *A* corresponds to either *a* or *e* and *I* corresponds to either *i* or *ı* [19].

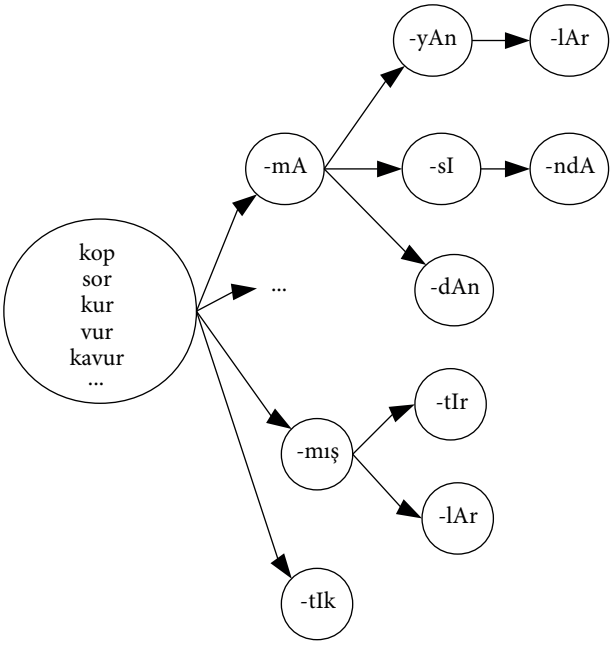

Figure 3. A portion of a verb FSA.

4.5.1. Selecting morphemes with letter bigrams

Letter bigrams were generated to apply consonant and vowel harmony while generating new word forms. We extracted two types of bigrams: 1. the last letter of the stem or the last suffix in the word, the first consonant

of the new suffix (CC or VC) 2. the last vowel of the stem or the last suffix in the word, the first vowel of the suffix (VV). The first rule deals with consonant harmony, whereas the second rule deals with vowel harmony. An example is given below:

Example 4.1
*kita**p** + **t**a + **d**ır − > "p:t" (CC) and "a:d" (VC) (means it is in the book)*
*mas**a** + d**a**n − > "a:a" (VV) (means from the table)*
*kal**e**m + l**e**r + **i**n − > "e:e" (VV) and "e:i" (VV) (means of the pen or your pens)*

We extracted 306 letter bigrams out of which 281 were bigrams for the consonant changes and 25 bigrams for the vowel harmony. Some of the bigrams and their probabilities are given in Table 3.

Table 3. Some of the letter bigrams and their probabilities computed by the relative frequency.

p:t - 0.00117	p:d - 0
s:t - 0.000584	s:d - 0
a:d - 0.00878	a:t - 0.0018
e:k - 0.00369	e:ğ - 0
e:i - 0.12	a:i - 0.0053
u:u - 0.05	u:ü - 0.000113
a:a - 0.075	a:e - 0.0022

4.5.2. Generating orthographic rules

We generated orthographic rules to apply consonant changes once the words are generated with the selection method based on the bigram probabilities but still require a consonant change. These rules deal with consonant mutation. Rules are defined analogously to Oflazer et al.'s [24] as follows:

$$f - > t || \text{ Left Context } \_ + \text{ Right Context}$$

Here, f transforms into t if the right context occurs just after the given left context and $+$ denotes a morpheme boundary. First, candidate rules were generated based on the analyses of the words (using the output of *Zemberek* [8]). For example, we generated the candidate rules from the surface form *kitabı* matching with the lexical form *kitap+ı* as given below:

Example 4.2
$p - > b \,||\, a \_ + ı$
$p - > b \,||\, ta \_ + ı$
$p - > b \,||\, a \_ + ın$
$p - > b \,||\, ta \_ + ın$

We defined a threshold of two letters from the end of the stem to the left context but we did not define any threshold for the number of letters in the suffix while generating the rules. Once all the candidate rules were generated, we counted how many times each rule was generated, that would be the rank of the rule. Rules that are substrings of each other were categorized and only the one with the highest rank was selected from each category.

Once the rules with the highest rank were selected, they were generalized by aggregating based on their context. For example, rules that diverge only with their contexts were aggregated if the union of the context letters contains all of the vowels. The generalized rule set is given in Table 4.

Table 4. Final orthographic rules.

t $->$ d || Vowel _ + Vowel
k $->$ ğ || Vowel _ + Vowel
ç $->$ c || Vowel _ + Vowel
p $->$ b || Vowel _ + Vowel
a $->$ ∅ || Vowel _ + Uyor
e $->$ ∅ || Vowel _ + iyor

4.5.3. Word generation

Once the FSA were built and the rules were extracted, FSA were traversed for the word generation process. Each pass through a stem state leads to a single stem, whereas each pass through a suffix state produces a stem + suffix concatenation (i.e. a word form).

Depth-first search is applied for each FSA. Each pass through a suffix state selects a group of suffixes. Suffix selection is performed based on the letter bigrams. The selection involves two steps.

In the first selection, depending on the last letter of the word, suffixes that give the highest probability with their first letters are selected (CC and VC). If there is more than one suffix that has the highest probability, then all of them are selected. For example, if the generated word is *kitap* (means *the book*) and the suffix state consists of *da, ta, de, da*, only *ta* and *te* win in this step (with letter bigram *p-t* which has a higher probability compared to *p-d*), and the other two suffixes (i.e. *da* and *de*) are ruled out because of the consonant harmony. If all suffixes in the state give zero letter bigram probability for the current word, search stops for this word and it continues from the previous state through the other paths in the FSA recursively.

If all suffixes in the state start with a vowel, the second selection is applied without applying the first selection (VV). This selection corresponds to the case of vowel harmony, thereby the selection uses only vowel bigrams. A single suffix or a set of suffixes (having the highest VV bigram probabilities) remain at the end of the second selection. If all of the bigrams give zero probability, the search continues from the previous state without generating a word form. For example, if the word is *kitap* (means *the book*) and the suffix state consists of suffixes such as *ı*, *i*, *u*, and *ü*, the letter bigram *a-ı* is selected (because of having the highest vowel bigram probability) leading to a word form *kitapı*. Finally, only a single suffix wins from the suffix state to be attached to the word. However, this is still not the final word form and orthographic rules will be applied on the word if needed.

Here, we did not consider the situation where the word ends with a vowel and all the suffixes in the suffix state begin with a vowel. An extra letter is added in Turkish for this situation and it was ignored in this study (e.g. *masa+**y**+a*) (means *to the table*). Nevertheless, the buffer letters such as *s*, *y*, and *n* are already a part of the suffix as a result of the morphological segmentation. One of the suffixes in the FSA is chosen based on the letter bigram probabilities. Therefore, we never generated a word form such as *masaa* but instead we generated word forms such as *masaya*, *masana*, and *masasa*. Since choosing the right buffer letter requires more analysis on the words, we leave this as a future work and our model may generate incorrect word form such as *masasa*.

Once the words are generated, orthographic rules are applied for the consonant changes, if needed. For example, for the word *kitapı* the following rule:
p $->$ b || "p" _ Vowel
is applied and the final word form becomes *kitabı* (means *the book* in the accusative case).

In Turkish, words do not end with *ğ*. We define end-of-word bigrams to prevent generating words ending with *ğ*. For example, if the final state consists of morphemes ending with *ğ* and *k*, only the suffixes ending with *k* are selected because *k:$* has got a higher probability when compared to *ğ:$* (i.e. *$* denotes the end of the word). This is applied for the final states.

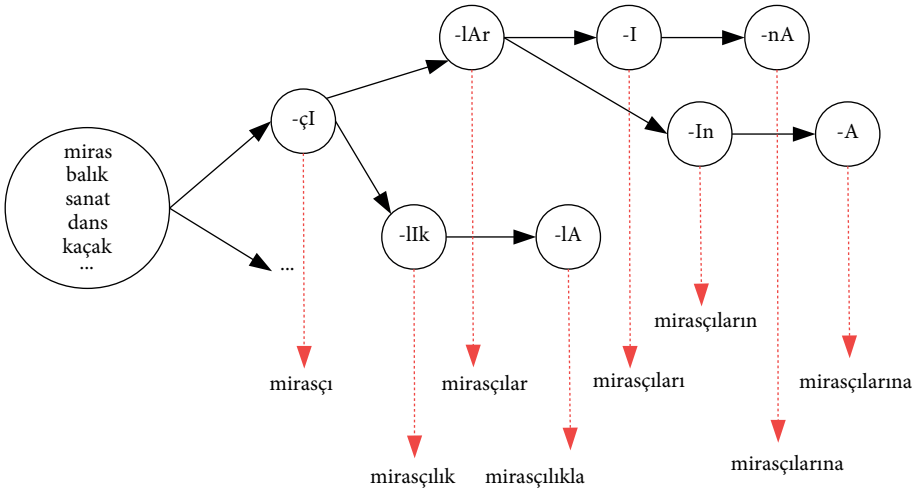

Figure 4. The generation of different word forms from the stem *miras* by using the FSA given in Figure 2.

An example word generation is given in Figure 4 and the generation algorithm is given in Algorithm 1.

5. Experiments and results

5.1. Experiments in a minimally supervised setting

We obtained 152 stem categories by using the similarity measure defined in Eq. 1. We obtained 89 stem categories by clustering the stems further with the Jensen–Shannon divergence. We evaluated the stem clusters based on purity:

$$\sum purity(S;C) = \frac{1}{N}\sum_{k=1}^{|S|} max_{j\in C}|S_k \cap C_j|, \tag{4}$$

where S denotes the resulting stem clusters, C denotes the gold stem clusters obtained from *Zemberek*. Here, N is the total number of stems. We used three main stem clusters that correspond to the part-of-speech tag of the stem, which are noun, adjective, and verb. Therefore, each resulting cluster was matched with the gold cluster that involves the most common stems with the resulting cluster. We used only noun, verb, and adjective stems since they are more salient compared to other stem categories. For 89 stem categories, the purity was 0.87, whereas for 152 stem categories the purity was 0.89. A higher purity means that there is a high overlap between the resulting clusters and the gold clusters. When more stem categories are merged, the purity decreases because stems belonging to different syntactic categories are merged.

Algorithm 1 Word generation algorithm

```
1: function GenerateWords
Input: A set of FSA X = {X^1, X^2, ...}, letter bigrams, orthographic rules
2:     for each: X^i in X do
3:         for each: s_j ∈ S_0^i do
4:             w ← s_j
5:             Add w in W'
6:             k ← 0
7:             A_k^i ← S_0^i
8:             for each: A_{k+1}^i do
9:                 A_k^i ← A_{k+1}^i
10:                if a suffix m_i is selected from A_k^i then
11:                    w ← w + m_i
12:                    Apply orthographic rules for w if needed
13:                    Add w in W'
14:                    A_k^i ← A_{k+1}^i
15:                else
16:                    A_k^i ← A_{k-1}^i
17:                    break
18:                end if
19:            end for
20:        end for
21:    end for
22: end function
```

We extracted 108 suffix categories as a result of the suffix clustering. We generated 120 candidate orthographic rules from which we generalized to 8 orthographic rules by aggregating the rules having the same context.

Since Turkish has a very productive morphology, the accuracy of our system can only be measured with the number of words generated correctly. There is not such a gold set/dictionary for the evaluation of the generated words in Turkish (and in any other agglutinative language since the number of possible word forms is theoretically infinite). For this reason, we used Zemberek [8] to decide whether a generated word was valid or not. If no analysis was given by Zemberek for a given word, then the word was counted as invalid. The evaluation was based on accuracy:

$$Accuracy = \frac{|V|}{|W'|}, \tag{5}$$

where V denotes the valid words and W' denotes all the words that are generated by the model.

We did two experiments. In the first set of experiments, we used the 152 stem categories that we obtained by using the similarity measure proposed by Baek et al. [21] (Eq. 1). We generated words by using different sizes of stems ranging from 100 to 3049. The results are given in Table 5. Even using 100 stems led to the generation of 7491 word forms with an accuracy of 76.03%. With the complete dataset of stems, that is 3049 stems, we obtained an accuracy of 76.79%. Therefore, the higher the purity of the stem categories is, the higher the accuracy is for the word generation.

In the second set of experiments, we used the 89 stem categories that were obtained by using the Jensen–Shannon divergence (see Eq. 2). The results for different sizes of stems are given in Table 6. By using 3049 unique stems, we obtained nearly 1 million words with an accuracy of 74.02%. The accuracy drops when

compared to the first set of experiments, but a larger set of word forms were generated by using fewer number of stem categories.

Table 5. Word generation accuracy scores obtained from the stem categories that were clustered according to the mutual-information-based similarity measure defined in Eq. 1.

Stem count	Number of stem categories	Number of generated words	Accuracy (%)
100	30	7.491	76.03
200	45	13.093	72.67
500	75	37.041	78.59
1000	95	80.569	76.14
2000	114	172.881	77.65
3049	152	237.219	76.79

Table 6. Word generation accuracy scores obtained from the stem categories that were further clustered by using the Jensen–Shannon divergence defined in Eq. 2.

Stem count	Number of stem categories	Number of generated words	Accuracy (%)
100	24	10.737	76.07
200	30	26.017	73.11
500	44	104.745	77.47
1000	53	243.097	76.14
2000	68	527.170	75.50
3049	89	915.288	74.02

We further clustered the stems by aggregating the allomorphs, such that *lar* and *ler* become *lAr*; *cı* and *çı* become *Cı*, and so on (see the notation in [19]). Aggregating allomorphs mitigate sparsity, therefore decrease the number of stem categories. We obtained 33 stem categories by using Jensen-Shannon divergence on the complete set of stems by using the aggregated allomorphs. The purity of the stem clusters becomes 0.88 for the aggregated allomorphs. Using 33 stem categories, the system generated 1,516,769 word forms with an accuracy of 73.19%.

We discovered that some suffixes were taken by stems only once or twice. We ruled out infrequent suffixes (i.e. with a frequency 1 or 2) from each suffix category by terminating the generation process for the infrequent suffixes. The results of this experiment are given in Table 7 for the frequency thresholds 1 and 2 separately. We obtained an accuracy of 82.36% when the frequency threshold was set to 2.

Table 7. Word generation accuracy scores when the infrequent suffixes were ruled out from the suffix states.

Frequency threshold	Number of generated words	Accuracy (%)
1	845.467	80.11
2	744.061	82.36

Our model is based on a pipeline process (stem clustering, suffix clustering, applying the rules based on bigram probabilities, applying orthographic rules, etc.). Therefore, any error emerging in one of the steps ends

up with an invalid word form. This is one of the weaknesses of pipeline approaches. When the results were analyzed, we observed that wrongly clustered stems and allomorphs caused most of the invalid words forms.

Without using any syntactic information, it is hard to cluster stems by using only the distributional information of the attached suffixes. We believe that the results will improve if any syntactic knowledge is incorporated in the model. Results are still promising for such a model by using a small set of training data without using any manually defined rules or other information apart from the word segmentations.

Our model will be the first attempt for minimally supervised word generation in Turkish.

5.2. Experiments in an unsupervised setting

In the unsupervised setting, we used the unsupervised morphological segmentation system, Morfessor CatMAP [16] to obtain the morphologically segmented words as training data. Stem categories, suffix categories, bigrams, orthographic rules, and FSA were learned from the training data without using any other knowledge.

We used the Turkish word list provided by Morpho Challenge 2010[4] to generate the training data. We selected the words having a frequency higher than 50 and discarded the rest to avoid the noisy words in the word list. We obtained 26,230 word types eventually.

We obtained 965 unique stems from 26,230 words that were split by Morfessor CatMAP. We generated 53 stem categories, 548 suffix categories, and 381 unique bigrams with a total frequency of 44,294. We could not produce any orthographic rules since Morfessor CatMAP does not provide any information about the phonological changes. We used the two-level morphological analyzer by Oflazer [4] to decide whether the given word form was valid or not. Our model generated 36,648 unique word forms with an accuracy of 68.9%. We also analyzed the results using Zemberek in order to be able to compare the unsupervised results to the results obtained from the minimally supervised setting. We obtained an accuracy of 31.58% using Zemberek [8].

It is still not fair to compare the results obtained from the minimally supervised and unsupervised setting since the number of generated word forms are different in both settings. Nevertheless, the results show that the accuracy of the morphological generation model harshly drops in the unsupervised setting.

Additionally, we compared our model with the unsupervised lexicon expansion model called *BabelGum* [15] that also uses Morfessor CatMAP [16] for the morphological segmentation. We used the same 26,230 words to run BabelGum. We split the word list into two and used one half to train the system and the other half as the development data. In the simple model of *BabelGum*, all affixes are concatenated. For example, the word *kalem+ler+im* (means *my pens*) is given as *kalem + lerim* (in the input data). In the bigram model, a bigram language model is used without concatenating the affixes in the input data. Eventually, their model generated around 1 million word forms for both models with an accuracy of around 10%, whereas we obtained an accuracy of around 30% from our model by generating around only 36 thousand word forms. We believe that the results are not compatible since the number of the generated word forms is quite different and it will not be fair to compare both systems based on accuracy. Nevertheless, we included the results to give a comparative idea about our model.

6. Discussion

We introduced a lexicon expansion model for Turkish in this article. Our system consists of several steps which affect the final accuracy of the generated words. Morphological segmentation step is one of the primary steps in the system that affects the accuracy. With the supervised morphological segmentation system Zemberek [8]

[4] http://research.ics.aalto.fi/events/morphochallenge2009/datasets.shtml/

we had 82.36% of accuracy, whereas it dropped to 68.9% when the unsupervised morphological segmentation system Morfessor CatMAP [16] was used in the fully unsupervised setting. The number of the generated words also dropped when Morfessor CatMAP was used because the morphological system undersplits the words and FSA tend to be more shallow when compared to the supervised setting.

Our system generated around 1 million words from 3049 unique stems. This shows that the morphotactics was applied effectively in the system with a generalization over the stems and suffixes. One of the advantages of our system was to induce stem and suffix categories, we generalized over stem categories that lead to generate more word forms. Using stem and suffix categories was one of the main differences with the other works in the literature. For example, Rasooli et al. [15] build the finite-state machines using a language model for the morphemes without generalizing over stem or suffix categories like our model. On the other hand, learning stem and suffix categories also have an impact on the accuracy of the system. We believe that adopting better clustering methods in these steps will also have a positive impact on the accuracy. We leave this direction for future work.

As for the error analysis, we observed the common errors in the results. For example, there were word forms such as, *indiriliminde*, *giyimindeyi*, *giyimlerindeyi*, *giyilmeyerektir*, *giyebilereklerdir*, *eğililmediğini*. Those errors were mainly because of the noisy stem categories, where merging the FSAs led to more general word generation rules such as *de-yi*, *erek-tir*, *erek-ler*. However, some of them require more information about the stems and the suffixes that they can take. For example, the stem *eğil* (meaning *to incline*) cannot take a passive suffix such as *-il*, which should be handled as an exception case. Once an invalid suffix is added to a word, all the other generations using this suffix lead to an erroneous word form. In this case, the model generates erroneous word forms such as *eğililebil*, *eğililebilir*, *eğililebilirlik*, *eğililebilirliği*, *eğililebilme*, *eğililebilmesi*, *eğililebilmeler*, *eğililebilmeleri*, *eğililebilerek* using the invalid word form *eğilil*. Some of the errors were not orthographic but semantical. For example, a word form such as *birikmeliğin* is orthographically and morphologically correct since *-me* can be added to *birik* (meaning *to accumulate*) because it is a verb, *-lik* can appear after *-me* as a suffix that converts a verb to a noun, and *-in* can appear after *-lik* since it can only be used with nouns and *-lik* can also be added to a noun. However, the resulting word form *birikmeliğin* is not valid and does not correspond to a valid meaning, which is also hard to learn in such a minimally supervised setting without using any exceptions.

7. Conclusion and future work

We proposed a novel word generation model using FSA for Turkish. We clustered stems and morphemes in a fully unsupervised framework in order to generalize the morphological generation rules so that they can be applied to syntactically similar stems. There is limited work on unsupervised or minimally supervised lexicon expansion on Turkish. To our knowledge, Rasooli et al. [15] proposed the only unsupervised lexicon expansion model. However, they did not generalize the lexicon expansion based on stem and suffix clusters. They built finite-state machines based on the morpheme language model that was obtained from unsupervised segmentation of the data. Therefore, our final FSAs mostly resemble the FSAs designed by Oflazer [4] (such as nominal FSAs, verbal FSAs, adjective FSAs, etc). Hence, we believe that our work will fill this gap in the field. Many natural language processing applications in Turkish suffer from sparsity due to the morphological productivity of the Turkish language. A word generation system can be used to reduce the size of the out-of-vocabulary words in any NLP task.

References

[1] Göksel A, Kerslake C. Turkish: A Comprehensive Grammar. London, UK: Routledge Comprehensive Grammars, 2005.

[2] Lewis G. Turkish Grammar. Oxford, UK: Oxford University Press, 2001.

[3] Koskenniemi K. Two-level morphology: a general computational model for word-form recognition and production. In: ACL '84/COLING '84 Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics; 2–6 July 1984; Stanford, CA, USA. pp. 178-181.

[4] Oflazer K. Two-level description of Turkish morphology. In: Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics; 21–23 April 1993; Utrecht, Netherlands: ACL. p. 472.

[5] Sak H, Güngör T, Saraçlar M. A stochastic finite-state morphological parser for Turkish. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers; 2–7 August 2009; Singapore: ACL. pp. 273-276.

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

[7] Çöltekin Ç. A freely available morphological analyzer for Turkish. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation; 2010; Valletta, Malta: European Language Resources Association (ELRA). pp. 820-827.

[8] Afşin, A, Akın MD. Zemberek, an open source NLP framework for Turkic languages. Structure 2007; 10.

[9] Beesley KR. Arabic finite-state morphological analysis and generation. In: Proceedings of the 16th Conference on Computational Linguistics - Volume 1; 1996; Copenhagen, Denmark: ACL. pp. 89-94.

[10] Cavalli-Sforza V, Soudi A, Mitamura T. Arabic morphology generation using a concatenative strategy. In: Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference; 2000; Seattle, WA, USA: ACL. pp. 86-93.

[11] Habash N. Large scale lexeme based Arabic morphological generation. In: Traitement Automatique des Langues Naturelles; 2004. pp. 271-276.

[12] Backwalter T. Arabic morphological analyzer version 1.0. In: Linguistic Data Consortium; 2002; University of Pennsylvania.

[13] Habash N, Rambow O, Kiraz G. Morphological analysis and generation for Arabic dialects. In: Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages; 29 June 2005; Ann Arbor, MI, USA: ACL. pp. 17-24.

[14] Kiraz GA. Multi-tape two-level morphology: a case study in Semitic non-linear morphology. In: Proceedings of the 15th conference on Computational Linguistics-Volume 1; 5–9 August 1994; Kyoto, Japan: ACL. pp. 180-186.

[15] Rasooli MS, Lippincott T, Habash N, Rambow O. Unsupervised morphology-based vocabulary expansion. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); June 2014; Baltimore, MA, USA: ACL. pp. 1349-1359.

[16] Creutz M, Lagus K. Inducing the morphological lexicon of a natural language from unannotated text. In: Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning; 2005. pp. 106-113.

[17] Köprü S, Miller J. A unification based approach to the morphological analysis and generation of Arabic. In: 3rd Workshop on Computational Approaches to Arabic Script-based Languages at MT Summit XII; 26–30 August 2009; Ottawa, Ontario, Canada. pp. 89-94.

[18] Tantuğ AC, Eryiğit G. Probabilistic Turkish word root generation. In: Proceedings of the 3rd Asia Pacific International Symposium on Information Technology; 13–14 January 2004; İstanbul, Turkey.

[19] Oflazer K, Göçmen E, Bozsahin C. An outline of Turkish morphology. In: Report on Turkish Natural Language Processing Initiative Project; 1994.

[20] Öztürk B, Can B. Clustering word roots syntactically. In: Proceedings of the 24th Signal Processing and Communication Application Conference; 16–19 May 2016; Zonguldak, Turkey.

[21] Baek DH, Lee H, Chang RH. Conceptual clustering of Korean concordances using similarity between morphemes; 2009.

[22] Kullback S, Leibler RA. On information and sufficiency. The Annals of Mathematical Statistics 1951; 22: 79-86.

[23] Can B. Unsupervised learning of allomorphs in Turkish. Turkish Journal of Electrical Engineering & Computer Sciences 2017; 25: 3253-3260.

[24] Oflazer K, Nirenburg S, McShane M. Bootstrapping morphological analyzers by combining human elicitation and machine learning. Computational Linguistics 2001; 27: 59-85.