GovdeTurk: A Novel Turkish Natural Language Processing Tool for Stemming, Morphological Labelling and Verb Negation

Sait Yucebas[1] and Rabia Tintin[2]
[1]Computer Engineering Department, Canakkale Onsekiz Mart University, Turkey
[2]Department of Student Affairs, Canakkale Onsekiz Mart University, Turkey

Abstract: *GovdeTurk is a tool for stemming, morphological labeling and verb negation for Turkish language. We designed comprehensive finite automata to represent Turkish grammar rules. Based on these automata, GovdeTurk finds the stem of the word by removing the inflectional suffixes in a longest match strategy. Levenshtein Distance is used to correct spelling errors that may occur during suffix removal. Morphological labeling identifies the functionality of a given token. Nine different dictionaries are constructed for each specific word type. These dictionaries are used in the stemming and morphological labeling. Verb negation module is developed for lexicon based sentiment analysis. GovdeTurk is tested on a dataset of one million words. The results are compared with Zemberek and Turkish Snowball Algorithm. While the closest competitor, Zemberek, in the stemming step has an accuracy of 80%, GovdeTurk gives 97.3% of accuracy. Morphological labeling accuracy of GovdeTurk is 93.6%. With outperforming results, our model becomes foremost among its competitors.*

Keywords: *Natural language processing, stemming, morphological analysis, Turkish language.*

Received June 18, 2019; accepted April 18, 2020
https://doi.org/10.34028/iajit/18/2/3

1. Introduction

The most important means of communication between people is language. The ability to derive the structural, syntactic and semantic rules of a language and model it in a computer environment, so that electronic devices can analyse, understand and even speak that language is called Natural Language Processing (NLP).

In order to obtain a complete computer model of the language, three main analysis are needed. Morphological analysis includes tasks and functions such as searching the roots of words, affixes and their types. Syntactic analysis measure the compliance of the elements that make up the sentence by comparing the hierarchical rules. Semantic analysis includes tasks and functions such as matching discrete words with appropriate objects in the database by using the knowledge base, and creating correct modelling structures in response to the meaning formed by interconnected discrete words [20].

One of the important aspect in NLP is the stemming that is to find last derived form of a given word. Stemmers are used in many areas such as text retrieval, filtering, mining, summarization, classification, question and answering systems [13]. One may group the studies on stemmers into three categories of suffix removal [23], statistical [19] approaches and hybrid methods [10, 15, 25].

Usually the language suffixes are defined and largely fixed. In suffix removal method, the morphological analysis of the language is of great importance. Suffix removal methods use two general approaches. In one approach, words along with their morphological information are stored in the dictionary. Relevant rules are applied and morphological results are obtained directly. In the second approach, only word roots are stored in the dictionary. Morphological results are obtained according to the language rules [20]. Both approaches are based on deletion of affixes or suffixes. Lovins' algorithm [16], Paice-Husk algorithm [22] and Porter's stemming algorithm are among the well-known stemmers that use suffix removal.

As the name implies, statistical stemming is based on statistical analysis. Most of the methods appear to remove the suffixes after performing some statistical procedures such as N-Grams and Hidden Markov Models. In general, statistical methods are independent of the morphological structure and grammar rules of the language. N-Gram [7], hidden markov model [12] and Yet Another Suffix Stripper (YASS) algorithm [18] are the well-known statistical stemmers.

Hybrid methods are designed to eliminate the disadvantages of a single method by combining the strengths of different methods. These methods are based on the word replacements that are related to the elements of the sentence and part of the speech.

Stemming studies on Turkish language mostly focus on suffix removal and statistical models. Statistical models suffers from time and calculation complexity.

Suffix removal models are either light stemmers or greedy. According to literature search, there is no study on Turkish Language that provides a stemmer with dictionary and sound event controls, morphological analysis and verb negation.

In this study, we developed a novel Turkish NLP Tool capable of stemming, morphological labelling and verb negation.

The paper is organized as follows; section 2: related work, section 3: methodology, section 4: performance analysis, section 5: conclusion.

2. Related Work

Stemmer studies in the literature can be grouped under three categories as suffix removal [22], statistical [19] approaches and hybrid methods [10, 15, 25].

Suffix removal methods are based on deletion of affixes or suffixes. A pioneer study on suffix removal is the Lovins' algorithm [17]. The algorithm removes the longest suffix at the end of the word. In a goal oriented manner, Lovins' provides a greedy approach and has advantages such as speed, removing double letters (e.g., setting -> set) and ability to find the roots of irregular plural words (indices -> index). Rules for word changes and suffixes are predefined. However, suffix list does not cover the most of the suffixes. That causes the algorithm to find incorrect stems. Another drawback is the search time of the suffixes. To overcome these, Dawson algorithm [3] uses a more comprehensive list of suffixes and tree-based indexing for mapping.

Unlike the longest match in Lovins [17] and Dawson [3], Paice- Husk algorithm [22], offers a repetitive structure based on the last letter. The last letter of the word is removed or modified according to predefined rules. The algorithm is terminated with respect to the three conditions: if a matching rule cannot be found for the last letter of the token; if the remaining word starts with a vowel and has two characters; if the remaining word starts with a consonant and has three characters.

One of the most popular stemming method is the Porter's stemming algorithm [23], which focuses on different word forms composed by vowels “V” and consonants “C”. Porter [23] states that four main word forms in English language can be represented by a single function. This function is given as [C](VC)m[V] and “m” denotes the repetition of the form. Based on the given function, five main rules are used to remove the letters at the end of the word. When a rule is accepted, the appropriate letter is removed and the next step is taken. When the fifth step is over, the remaining token is labeled as root. Compared to the algorithms given above, Porter eliminates the need for suffix storage. However, algorithmic complexity increases due to the number of rules applied.

Poters' [23] algorithm is applicable for multiple languages with the snowball structure [10]. Eryigit and Adali [6], and Cilden [2] made some extensions to the snowball to reflect agglutinative structure of the Turkish language. By this way they were able to adapt Porter stemmer to the Turkish language.

To eliminate the language dependency of the greedy methods, statistical calculations are used. N-Gram [7], hidden markov model [12] and YASS algorithm [18] are among well-known statistical stemming methods.

N-Gram [7] uses the occurrence patterns of the letters. Sub-arrays consisting of n consecutive characters are called n-grams. The specified number of n is limited to four or five, although it is language-specific. Inverse Document Frequency (IDF) measure is used to find the similarity between the word sequence and n-grams [10]. This model suffers from time complexity for statistical calculations and the space to store n-grams.

Hidden markov model based stemmer [19], uses a word list to train the model and probability functions for transitions between states. The beginning state is the root of the word. The viterbi algorithm [11] calculates the possibility of each path between the states. Then the most likely path is discovered. This learning scheme provides language independency but the probability calculations are time consuming.

Hybrid approaches combine the advantages of several methods. For example, statistical methods do not use a dictionary to control the root or stem. To eliminate this drawback, Krovetz [15] uses dictionary control. The algorithm is based on two-step conversion; suffix elimination, and dictionary control. In the first step, the word is converted to its singular form. In the second step, the past tense, present tense transformation is done. Then suffixes are removed from the token. The found root is checked via a dictionary. This stemmer can suffer from understemming because it does not provide a comprehensive morphological analysis.

The work by Hull and Grefenstette [9] presents a more detailed stemmer. In addition to dictionary control, they have introduced an English word database that provides morphological analysis. This analysis is based on both suffix and prefix removal. Four reduction steps are applied in the form of singular nouns, verbs, adjectives and pronouns. If a difference in meaning occurs between the word and the stem, no reduction is made. This method requires an extensive word database and a comprehensive dictionary. Therefore, if this approach is applied to Turkish language, a dictionary size of a few million words will be needed.

The corpus-based stemmer, by Xu and Croft [25], was developed to cover conflation problems of the Porter's stemmer. Porter's algorithm could fail to assign correct stems for homographic and synonyms. In this algorithm, Porter stemmer is used in first place then statistical calculations are used for co-occurrences to handle conflation problems. This method reduces under and over stemming but highly dependent to the corpus.

When the studies on Turkish language are examined, greedy and statistical approaches comes forward [5].

Dincer and Karaoglan [4] developed a probabilistic stemmer. This stemmer is based on Ordered Pairs (OP) and the word length. Three probabilities are calculated. These are the probabilities of OP belongs the stem, OP belongs the suffix and the OP is between stem and suffix. The stem of the word is assigned according to the probability calculation of the ordered pairs. Although stemming performance is high, their model struggles for pronoun stemming.

A two phased approached is proposed in [12]. At first phase, words are grouped as noun or verb by using a Hidden Markow Model, and then suffixes are removed accordingly. In the second phase, stem boundaries and N-Grams are used to find the actual stems. This model could suffer from time complexity for large datasets.

The greedy Turkish stemmers are based on suffix removal. In Koksal's [14] study, fixed number of letters is removed at the end of the word without using grammar rules or statistical inference. The remaining token is labelled as the root. This model is a very light stemmer and suffers from understemming.

Solak and Can [24] developed the A-F algorithm based on letter removal and added a dictionary control. Words in the dictionary contain different flags that indicate the types and statuses they can take. If a given word is not located in the dictionary, a letter at the end of the word is removed and the new token is searched. Algorithm continues until one letter remains in the token. All matching sub tokens are marked as candidate stems. If the original word is built from a specific candidate stem, it is labelled as actual stem. This method does not differentiate the roots and stems. Instead of finding a single stem, it offers possible stems.

The algorithm developed by Eryigit and Adali [6], starts at the end of the word and suffixes are removed according to the rules defined by FSMs. They state that, a dictionary check is not needed because all suffix addition rules are covered by the FSMs. This algorithm is applied to the Poters' snowball [23] by Cilden [2]. The algorithm proposes approaches for the development of snowball stemmer for languages that have strict rules. Cilden [2] defined rules to control vowel harmony and merging letters, but in Eryigit and Adali [6], these letter changes are coded into affix tables.

Akın and Akın [1] is a multi-purpose natural language processing tool for the Turkish language. Its stemmer module uses a dictionary like structure. This structure contains the root words, their type and special flags. A given word is analysed from left to right. Starting from the first letter the word is scanned on a tree structure that contains the Turkish grammar rules. If the input word can be created by following the nodes in the tree, the root of the related word is assigned as the root in the dictionary.

Most of the stemming studies on Turkish language are based on the suffix removal. Some methods start at the end of the word while others start from the beginning and try to reproduce the given word. Another difference is the dictionary usage. The disadvantages of the dictionary control are search time and dictionary dependency. However, without using a dictionary, it is impossible to identify whether the word is root or not.

For Turkish Language stemming, it is very important to control the found stem, to control the letter changes caused by sound events and to conduct a morphological analysis. However, the studies carried out for the Turkish language, have been skipped one or more of these elements due to the purpose of the study or performance limitations.

In the light of the discussions presented above, a stemming algorithm on Turkish language was developed and presented in the [26]. This preliminary algorithm uses two different FSMs to represent the suffix attachment rules and a single dictionary to check the stems. The biggest shortcoming of the preliminary algorithm is the morphological analysis. Morphological labelling is crucial, because the type of a given word affects the suffixes it can take and the grammar rules for these suffixes.

In this study, the preliminary algorithm is enhanced and a morphological labelling step was added. Other deficiencies of the preliminary algorithm are also addressed in this study. FSMs are extended to cover all the word types, privative affixes, and possessive (determinative) suffixes. Instead of using different FSMs, all FSMs are integrated in a single FSM. The dictionary check mechanism is extended to cover all word types. A control mechanism is used for sound events and homonyms. A new module is developed for verb negation. This module will be used for stylistic pattern and emotion analysis in future studies.

As a summary, preliminary GovdeTurk algorithm is extended to a novel NLP tool capable of stemming, morphological labelling and verb negation. It is tested on a dataset with one million words. The results are compared with other well-known stemmers. With outperforming results, the proposed method proves its potential on Turkish language.

3. Methodology

GovdeTurk algorithm is based on removal of the inflectional suffixes. These suffixes are deleted according to the rules defined by FSMs and the longest match strategy. The found stems are checked via dictionaries, which are based on tree structure. Although rarely seen, there can be errors in the suffix removal phase. These spelling errors are handled via Levenshtein distance [16].

3.1. Dictionary Design

Dictionaries, based on Turkish language association, are used to check if the found stems are correct. All dictionaries use simple text lists with letter indexing. The preliminary algorithm uses two dictionaries for verbs and nouns. Additional dictionaries are used in this work. These are verb, noun, adjective, adverb, pronoun, preposition, exclamation, conjunction and homonymic word dictionaries.

Each dictionary is organized in the memory as tree structure. The nodes refer to the letters. Maximum number of children for a given node is equal to the number of letters in the alphabet. In reading step, a node is created for each different character. Figure 1 shows an example of the tree structure for "*Bal*" (honey), "Bale" (ballet) and "Balık" (fish).

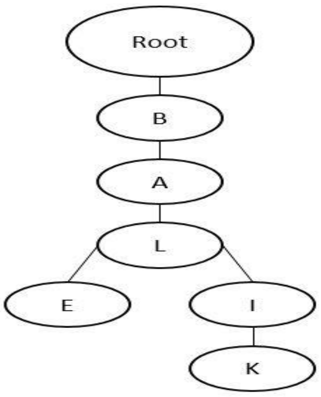


Figure 1. An example of tree structure for words bal (honey), bale (ballet) and balık (fish).

3.2. FSM Design

The words in Turkish language are grouped under two types, verbs or nouns. These main word groups can change their types by derivational suffixes. Each word type can take certain inflectional suffixes. The inflectional suffixes of verbs cannot be used for nouns and vice versa. Thus, two discreet FSMs are designed at first step. Then these FSMs are integrated to reflect the suffix attachment rules for both verbs and nouns.

The FSM designed for nouns is based on RULE-1 given in the Figure 2.

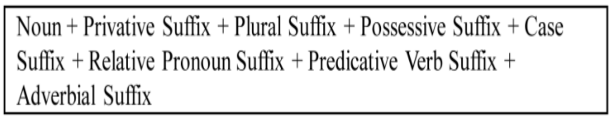

Noun + Privative Suffix + Plural Suffix + Possessive Suffix + Case Suffix + Relative Pronoun Suffix + Predicative Verb Suffix + Adverbial Suffix

Figure 2. Suffix addition order rule for nouns in Turkish language.

Genitive case was designed as a node in the FSM in the preliminary algorithm. In this work, suffixes that form genitive case are represented as transitions of case suffixes. Another extension is done for relative pronoun. The word that takes relative pronoun forms a new noun. Therefore, RULE-1 can start over. The FSM for nouns is given in the Figure 3.

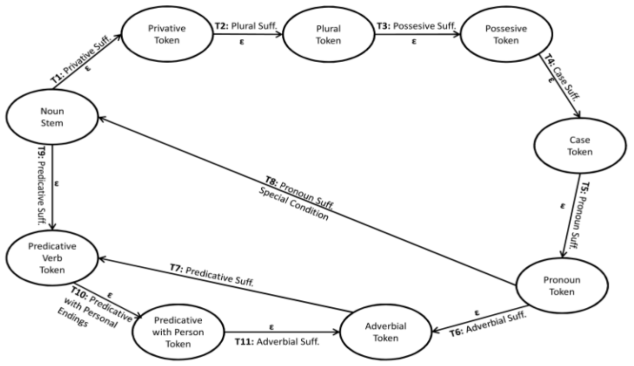


Figure 3. FSM for inflectional suffix ordering in nouns.

The word "arkadaşlarımınkiler" (one of my friends') is an example for relative pronouns. The word "arkadaşlarımınkiler" do not have a direct English word equivalent. It means "one of my friends' X". X here can be any property. "Arkadaşlarımın" (my friends') takes possessive suffix "-ki" and becomes "arkadaşlarımınki". This word is treated as a new noun and rewinds the suffix sequence. This is the only exception that the suffix attachment sequence starts-over. According to the FSM given in the Figure 3, this word can be formed follows:

- Arkadaş {Friend} (noun root) -> Transition-2 (+lar/ plural suffix)
- Arkadaşlar {Friends} (plural noun) -> Transition-3 (+ım / possesive suffix)
- Arkadaşlarım {MyFriends} (possessive noun) -> Transition-4 (+ın/case suffix)
- Arkaaşlarımın {My Friends'} (case noun) -> Transition-5 (+ki/relative pronoun suffix)
- Arkadaşlarımınki {One of my frends'} (pronoun) -> Transition-8 (special condition)
- Arkadaşlarımınki {One of my frends'}(noun root) -> Transition-2 (+ler/plural suffix)
- Arkadaşlarımınkiler (plural noun)

The inflectional suffix order for verbs is also strictly defined. RULE-2 in the Figure 4 explains this order and the FSM for verbs are designed according to it.

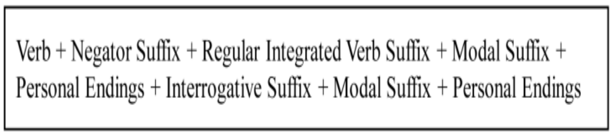

Verb + Negator Suffix + Regular Integrated Verb Suffix + Modal Suffix + Personal Endings + Interrogative Suffix + Modal Suffix + Personal Endings

Figure 4. Suffix addition order rule for verbs in Turkish Language.

In the preliminary algorithm the negation suffix was grouped under derivational suffixes. However, it does not derivate a new meaning or change the word type. Thus, in this work, it is grouped under inflectional suffixes. The new algorithm allows the plural suffix can be added before or after the modals. Also, regular integrated verb suffix and transitions are added. The FSM for verbs is given in the Figure 5.

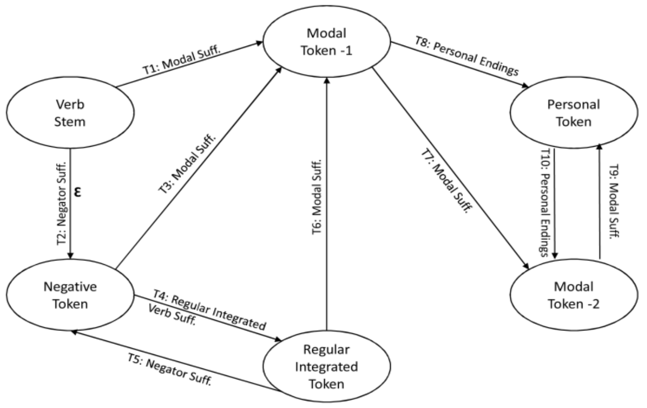


Figure 5. FSM for inflectional suffix ordering in verbs.

The word "*geliyordular*" (they were coming) is an example for third plural verb. According to FSM given in the Figure-5, the word "*geliyordular*" is formed as follows:

- Gel {Come} (verb root) -> Transition-1 (+(i)yor/modal present cont.)
- Geliyor {He is coming} (modal-1) -> Transition-7 (+du/past tense)
- Geliyordu {He was coming} (modal-2) -> Transition-10 (+lar/3rd plural person)
- Geliyordular {They were coming} (personal verb)

The FSMs of verbs and nouns are integrated by gerundial is given in Figure 6. The root of the gerundial is verb but they can take all of the noun inflectional suffixes. Thus, both FSMs must be activated. The stemming starts with either the removal of noun or verb suffixes. The transition in between is done by gerundial.

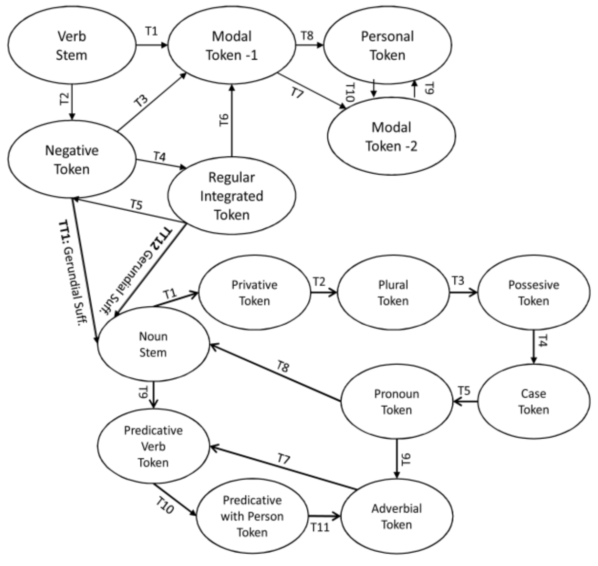


Figure 6. Main FSM for inflectional suffix ordering.

The word "uğramamanız" is an example for verb to noun transition by a gerundial suffix. "Uğramamanız" is a noun token with the meaning "the condition for you, not stopping by". According to main FSM, given in the Figure 6, the transition between verb and noun is formed as follows:

- Uğra {Stop by} (verb root) -> T-2 (+ma/negator suffix)
- Uğrama {Don't stop by} (negative token) -> TT-1 (+ma /gerundial suffix)
- Uğramama (noun stem) -> T-3 (+nız/possesive suffix)
- Uğramamanız {the condition for 2nd plural person not stopping by} (possessive token)

3.3. Matching and Control Mechanisms

The longest match scheme is used to map the suffixes. If the suffixes are removed by a single letter or the shortest match scheme, incorrect suffix removal could occur. In this condition, type of words and/or suffixes can be puzzled that leads finding the wrong stem. When the shortest match is applied to the word "*içtim*" (I drunk), the letters "*m*" and "*i*" are matched as predicative verb suffixes. Thus, the stem of the word will be found incorrectly as "*içt*". In longest match scheme, the word phrase "*-tim*" will be mapped to the past tense modal with first person singular suffix "-*dim*". When this suffix is removed, the actual stem of the word "*iç*" (drink) is found.

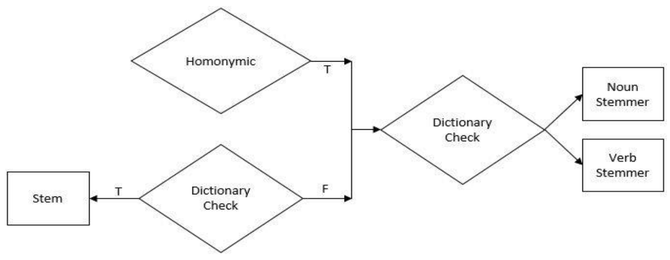
Even the longest match could find incorrect stems. Levenshtein distance [16] is used to correct such cases. This method is also used when the word and stem types do not match. The stem of the word "*okuyorum*" (I'm reading) is found as "*ok*" (arrow), because the longest match algorithm removes the suffix "*-uyorum*" by mistake. However, the stem "*ok*" is a noun type and the removed suffix "*-uyorum*" is a verb suffix. In such conditions, the Levenshtein distance assigns the found stem to its closest verb neighbour, "*oku*" (read).

The proposed method uses dictionary controls. However, we have noticed that multiple derived words are not found in the dictionaries. Derivational suffix control method is developed for this condition. This control works as follows. After all inflectional suffixes are removed; the remaining suffixes are checked from derivational suffix lists. If the derivational suffix is matched to a noun or verb type, then only the derivational suffixes for the appropriate type are searched.

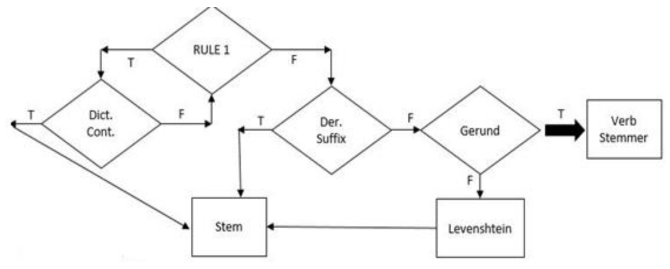
Another important control mechanism is developed for the homonymic words. For example, the word "*yüz*" can have different meanings such as the number one hundred, face, swim and striping a skin. The same word can have different meanings and according to this meaning, word type can change. Thus, identification of homonymic word types is very crucial. The control mechanism tries to find a homonymic word in the sentence. If one is found, then a verb is searched. If the verb is found, then the homonymic is more likely to be a noun. If no verb is found, then the homonymic word itself is a verb. Although this method provides a solution for homonymic words, it is not fully accurate. The predicative suffixes "*-idi, -imiş*" can be confused with past tense modal suffix "*-di, -miş*". If the homonymic word is noun, the control mechanism can detect the word type as a noun but cannot understand the meaning of the word. Semantic analysis is needed for a full-scale solution and our research continues on this topic.

3.4. Stemming Algorithm

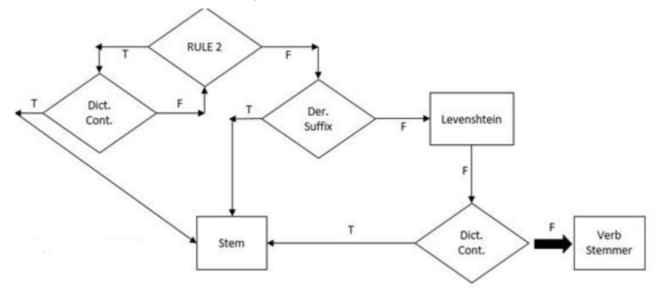
The stemming algorithm consists of 10 main steps and their substeps. Figure 7 represents a summarization of the stemmer algorithm.



a) Main controller.



b) The noun stemmer.



c) The verb stemmer.

Figure 7. Framework for stemmer algorithm.

The step by step flow of the algorithm is given below:

1. Input text parsed line by line and each line divided into sentences.
2. A parser used to exclude the punctuation marks and the suffixes that are added by apostrophe, from the text. Then all letters converted to lower case.
3. Each sentence divided into word tokens.
4. Each token spell checked.
5. The spell checked tokens given to stemmer algorithm.
6. Homonymic word control is used. If the word is homonymic, the control mechanism labels the word as verb or noun. According to the label, one of the sub FSMs activated.
7. If the word is not homonymic, main FSM activated.
   a. The token searched in the verb and noun dictionaries. If the token is a nominative stem, it is identified in this step and the stemmer algorithm is terminated. Else,
8. RULE-2 and RULE-3 activated in reverse order via verb FSM.
   a. Modal and personal ending deleted, the remaining token searched in the verb dictionary. If a match found then the token labeled as verb stem.
   b. If there is no match, then the negation suffixes deleted from the token. The remaining token searched in the verb dictionary. If a match found then the token is labeled as verb stem.
   c. If there is no match, regular integrated verb suffixes deleted at the end of the token. The remaining token searched in the verb dictionary. If a match found then the token labeled as verb stem. Else,
   d. Derivational suffix control is applied. If the remaining token has a derivational suffix that means the last derived word, the stem, is found. Thus, the token labeled as stem. Else,
   e. The condition indicates a spelling error. A two-phase approach used for correction.
      1. In the first phase, all suffix deletions reversed gradually. In each step, the token searched in the dictionaries. If a match occurs, then the matching token labeled as a stem. Else,
      2. The phase two is proceeded and Levenshtein method used to assign the token to its nearest neighbour. The corrected token searched in the dictionaries. If a match occurs, then the matching token labeled as a stem.
   f. If there are no errors and a stem is not found yet, the token is labeled as noun and FSM for nouns is activated.
9. RULE-1 activated in reverse order via noun FSM.
   a. The adverbial suffix deleted then remaining token is searched in the dictionary. If the token located in the dictionary then labeled as stem. Else,
   b. If the token has predicative verb suffix, this suffix deleted. Remaining token is searched in the dictionary, if the token is located then it labeled as stem. Else,
   c. If the token has relative pronoun suffix, this suffix deleted (Relative pronoun suffix has a special condition. The token with a relative

pronoun can be used as a new noun. Thus, RULE-1 can be applied more than once. A control mechanism is used to check this condition). Remaining token searched in the dictionary, if the token located then it labeled as stem. Else,

d. If the token has case suffix, this suffix deleted. Remaining token searched in the dictionary, if the token is located then labeled as stem. Else,
e. If the token has possessive suffix, this suffix deleted. Remaining token searched in the dictionary, if the token is located then labeled as stem. Else,
f. If the token has plural suffix, this suffix deleted. Remaining token is searched in the dictionary, if the token is located then labeled as stem. Else,
g. If the token has privative suffix, this suffix deleted. Remaining token searched in the dictionary, if the token is located then labeled as stem. Else,
h. For the reaming token derivational suffix control is applied. If the token has one of the derivational suffixes, it is labeled as a stem. Else,
i. Gerundial suffix is deleted from the token. The remaining token is searched in the verb dictionary, if it is located then labeled as stem. Else,
j. Transition between noun and verb stemmer is done. The gerundial token passed to verb stemmer and the algorithm starts from 8-b. Else,

10. This indicates a spelling error. A two-phase approach used. In the first phase, all suffix deletions reversed gradually. In each step, the token searched in the dictionaries. If a match occurs, then the matching token labelled as a stem. Else, phase two is proceeded and Levenshtein method used to assign the token to its nearest neighbour. The neighbour labelled as a stem.

3.5. Morphological Labelling

Suffix attachments in Turkish language are given by definitive rules. Depending on the current token type, the suffix sequence may indeed be changed. Thus, it is very crucial to identify the exact type of the word for stemming. For this reason, a module for morphological labelling is developed. This labelling also helps the address morphological disambiguation [8]. The labelling of the verbs and nouns are done in the stemming step. The word types of adjective, adverb, pronoun and conjunctions are labelled in this module. The morphological labelling algorithm is summarized in a case statement including six steps as follows:

1. Nominative words that are not affected from the word sequence are labelled before entering the algorithm. If the word is not labelled in this step, second step is followed
2. The existence of the word in the adjective dictionary is examined. If the word is located in the dictionary, other words in the sentence are examined:
   a. If the following word is verb or gerundial, the main word is labelled as noun-adverb
   b. If the following word is noun, the main word is labelled as noun-adjective
   c. If there is no following word, the main word is labelled as noun-noun
3. The existence of the word in the pronoun dictionary is examined. If the word is located in the dictionary, other words in the sentence are examined:
   a. If the following word is verb or gerundial, the main word is labelled as noun-pronoun
   b. If the following word is noun, the main word is labelled as noun-adjective
   c. If there is no following word, the main word is labelled as noun-pronoun
4. The existence of the word in the conjunctive dictionary is examined. If the word is located in the dictionary, other words in the sentence are examined:
   a. If the following word is verb or gerundial, the main word is labelled as noun-postposition.
   b. If the following word is noun, the main word is labelled as noun-conjunctive.
   c. If there is no following word, the main word is labelled as noun-postposition.
5. The existence of the word in the homonymic dictionary is examined. If the word is located in the dictionary, other words in the sentence are examined
   a. The homonymic word control mechanism used to label the word as noun or verb.
6. If the word is not located in the homonymic dictionary, it is directly passed to the stemming algorithm where the type of the word can be identified as a verb or as a noun.

If all the steps above fail to give a label, then the word is tagged as noun-noun and the morphological labelling algorithm terminated.

3.6. Verb Negation Module

The main aim of the GovdeTurk is the stemming. However, GovdeTurk is developed to be a comprehensive NLP tool for Turkish language. Thus, in future studies, it can be used for translation-based dictionaries and lexicon based sentiment analysis. In order to determine the conditions that are not present in the dictionary translations but given in the language itself and to determine the negativity in the sentence, a verb negation module is developed. This module takes the verbs in sentence and converts them to their negative form. The module takes first two letters of the suffix and applies the following rules:

- If the suffix starts with a consonant and the second letter is a sanserif (a, e, ı, i), the negation suffix is "-*ma*" or "-*me*". The form of the following suffix does not change
- If the suffix starts with a consonant and the second letter is round narrow letter (u, ü), the negation suffix is "-*ma*" or "-*me*". The form of the following suffix changes
- If the suffix starts with a consonant and the second letter is round narrow letter (u, ü), the negation suffix is "-*ma*" or "-*me*". The form of the following suffix changes
- If the suffix starts with narrow vowel (ı, i, u, ü) and continues with the letter "*y*", the negation suffix is "-*m*". This situation is seen in present continuous tense and the form of the following suffix does not change
- If the suffix starts with a vowel, the negation suffix is "-*may*", "-*mey*". The following suffix does not change
- If the verb is in nominative, the letters of the verb is searched from right to left. If the first encountered vowel is a back vowel, then the negation suffix becomes "-*ma*". If the first encountered vowel is a front vowel, then the negation suffix becomes "-*me*" The algorithm takes two string inputs. The first string is the stem that found in the stemmer algorithm and the second is the original word. These two inputs are compared and all the suffixes are identified. Then according to the rules above, the appropriate negation suffix and the following inflectional suffixes are attached.

4. Performance Analysis and Comparisons

To test the stemmers' performance, 6,608,734 words are retrieved via crawlers. Among these, a million words are selected with the highest frequencies to form a dataset.

The speed of the algorithm is compared to Zemberek v2.1.1 [1]. A laptop with 2.6 GHz Intel Core5 64-bit processor 6GB of ram is used for the comparison. The stemming phase lasts 7.6 second in GovdeTurk. It is one micro-second slower per word then Zemberek.

The first accuracy comparison is done with Zemberek 2.1.1. The results are given in Table 1.

Table 1. Accuracy comparison of GovdeTurk and Zemberek.

Class of The Word	GovdeTurk	Zemberek
Correct	973053	808590
Incorrect	18023	182486
Unclassified	8924	8924

The accuracy of the GovdeTurk is 97.3%. Zemberek on the other hand, gives 80.8% of accuracy. The unclassified words are the homonymic words. This control is excluded from the algorithm because Zemberek does not provide such control. If the unclassified words are excluded from the dataset, the accuracy of the Govde Turk increases to 98.2% and for Zemberek to 81.6%. The type-1 and type-2 error comparison is given in the Table 2.

Table 2. Error comparison of Govde Turk and zemberek.

GovdeTurk	Zemberek	Number of Stems
Correct	Incorrect	92,462
Incorrect	Correct	5903
Incorrect	Incorrect	3583

When we analyse the result given in Table 2 we see that Zemberek fails to classify the verbal words. Instead of finding the last derived word, the stem, Zemberek finds the first derived word or root.

The work of Eryigit and Adali [6] and the work of Cilden [2] are among the prominent Turkish NLP studies. These are designed to find the root of the words but if the derivational suffix elimination phase is excluded, they will find the stems. The Turkish Snowball algorithm [2] shows a fractional difference in stemmer performance when compared to Eryigit and Adali [6]. The difference caused by additional rules to control vowel harmony and merging letters. Thus, the stemming performance of Gövde-Turk is compared to Turkish Snowball [2]. The results are given in the Table 3.

Table 3. Accuracy comparison of GovdeTurk and Turkish snowball.

Dataset	Turkish SnowBall	GovdeTurk
Derived Noun Stems	88%	98%
Derived Verb Stems	26%	98%
Nominative Words	58%	96%

Table 3 shows that GovdeTurk is not affected by the type of the word. It is much better than snowball for derived noun stems and it shows an outstanding performance for derived verbs and nominative words. The Turkish snowball algorithm fails for gerundial, integrated and regular integrated verbs. The high performance of GovdeTurk can be based on the dictionary control and the Levenshtein correction.

A sentence-based dataset of 2019 words is used to test the homonymic word control and the morphological word labelling. The time needed to label all the words in the dataset is 33 seconds (16 mille sec. per word). The labelling and stemming performances are given in the Table 4.

Table 4. Stemming and labelling performance of GovdeTurk.

	Number of Words	Percentages
Correct Stems	1929	95.33
Correct Label	1891	93.66
Incorrect Stems	90	4.45
Incorrect Label	128	6.33

When only the homonymic words are considered, the accuracy for stemming is 78% and accuracy for labelling is 65%. If the homonymic word control is excluded, the stemming accuracy decreased to 43% and labelling accuracy decreased to 8%.

A new dataset is formed by 10% randomly selected words from the main dataset. This subset is used to test the verb negation module. The accuracy is measured as 87%.

5. Conclusions

The aim of the stemming study is to find last derived form of a given word. Suffix removal [23], statistical [19] approaches and hybrid methods are widely used for stemming [10, 15, 25]. When the studies on Turkish language are examined, greedy and statistical approaches comes forward [5].

To overcome the challenges indicated in [21], it is very important to control the found stem, to control the letter changes caused by sound events and to conduct a morphological analysis. However, the studies carried out for the Turkish language, have been skipped one or more of these elements. With this study we try address these issues and develop a complete and comprehensive NLP tool for Turkish language. Thus, govde-turk covers different modules in a single framework. The application is online at www.GovdeTurk.org.

In this work basic stemmer of the [26] is enhanced. FSMs cover all suffix attachment rules. Genitive case and pronoun transitions are added to noun FSM. Negation suffix, regular integrated verb suffix and the transitions concerning them are added to the verb FSM. Verb FSM also allows attachment of plural suffix before and after the modals. The sub FSMs are integrated under a main FSM. Gerundial suffix and derivational word control provide the transition between sub FSMs.

In addition to Levenshtein distance and longest match scheme of the preliminary algorithm, we provide additional control mechanisms to increase the robustness. Derivational controls for verbs and nouns are designed to cover multi-derived words. Homonymic word control mechanism is developed to identify the type of the homonymic words. To the best of our knowledge, GovdeTurk is the first Turkish stemmer that provides homonymic word control. Besides the stemmer, morphological labeling module is developed to identify the exact word types according to their intra sentence functionalities. Verb negation module is developed for translation-based dictionaries and lexicon based sentiment analysis. This module can detect the negativity of the sentence by examining the verb. It can also convert the positive verb to its negative form without changing the original attached suffixes.

GovdeTurk is compared with the foremost Turkish stemmers. A dataset of one million words is used for the comparison.

The first comparison is done with zemberek 2.1[1]. The stemming accuracy of GovdeTurk is 97.3% where the accuracy of zemberek is 80%. GovdeTurk is also compared with the turkish snowball algorithm [2]. To find the words that increase the error rate of the methods, the data set is divided into three subgroups of derived nouns, derived verbs and nominative words. The stemmer accuracy of the GovdeTurk over these datasets is 98%, 98% and 96% respectively. The accuracy of the snowball is 88% for derivative verbs, 26% for nouns and 58% for nominative words.

A dataset of 2019 words is used to test the labeling and the homonymic word control modules. The labeling accuracy is found as 93.66%. The homonymic word control increases the accuracy of the stemmer by 35%, and increases the accuracy of morphological labeling by 57%.

GovdeTurk outperforms its competitors and show its potential on being a comprehensive Turkish NLP tool. This tool can be used for main NLP proposes such as stem/root finding and morphological labelling. It can also be used for search engines and information retrieval proposes. As future work, we plan to develop another module for sentiment analysis that will be based on verb negation and morphological labeling of the current system.

References

[1] Akın A. and Akın M., "Zemberek, an Open Source Nlp Framework for Turkic Languages," *Structure*, vol. 10, pp. 1-5, 2007.

[2] Cilden E., "Stemming Turkish Words Using Snowball," Retrieved 08.05.2019 from: http://snowball.tartarus.org/algorithms/turkish/stemmer.html, Last Visited, 2006.

[3] Dawson J., "Suffix Removal and Word Conflation," *ALLC Bulletin*, vol. 2, no. 3, pp. 33-46, 1974.

[4] Dinçer B. and Karaoğlan B., "Stemming in Agglutinative Languages: A Probabilistic Stemmer for Turkish," *in Proceedings of Computer and Information Sciences-ISCIS 2003, 18th International Symposium*, Antalya, pp. 244-251, 2003.

[5] Sever H. and Duran G., "Turkce Govdeleme Algoritmalarinin Analizi," *in Proceedings of Annual Conference of TBD'96*, İstanbul, pp. 23-243, 1996.

[6] Eryigit G. and Adali E., "An Affix Stripping Morphological Analyzer for Turkish," *in Proceedings of Artificial Intelligence and Appplications*, Innsbruck, pp. 299-304, 2004.

[7] Freund G. and Willett P., "Online Identification Of Word Variants and Arbitrary Truncation Searching Using A String Similarity Measure," *Information Technology: Research and Development*, vol. 1, no. 3, pp. 177-187, 1982.

[8] Hakkani-Tür D., Saraçlar M., Tür G., Oflazer K., and Yuret D., *Turkish Natural Language Processing*, Springer, 2018.

[9] Hull D. and Grefenstette G., "A Detailed Analysis of English Stemming Algorithms," *Xerox Research and Technology*, vol. 6, pp. 1-16, 2016.

[10] Jivani A., "A Comparative Study of Stemming Algorithms," *International Journal of Computer Applications in Technology*, vol. 2, no. 6, pp. 1930-1938, 2011.

[11] Jurafsky D. and Martin J., *Speech and Language Processing*, Prentice Hall PTR, 2018.

[12] Kısla T. and Karaoglan B., "A Hybrid Statistical Approach to Stemming in Turkish: an Agglutinative Language," *Anadolu University Journal of Science and Technology-A Applied Sciences and Engineering*, vol. 17, no. 2, pp. 401 -412, 2016.

[13] Khan S., Anwar W., Bajwa W., and Wang X., "Template Based Affix Stemmer for A Morphologically Rich Language," *The International Arab Journal of Information Technology*, vol. 12, no. 2, pp. 146-154, 2015.

[14] Koksal A., "Tümüyle Özdevimli Deneysel Bir Belge Dizinleme Ve Erisim Dizgesi: TÜRDER," *in Proceedings of Bilisim 80' Bildiriler*, Ankara, pp. 37-44, 1981.

[15] Krovetz R., "Viewing Morphology As An Inference Process," *in Proceedings of 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, NewYork, pp. 191-202, 1993.

[16] Levenshtein V., "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *In Soviet Physics Doklady*, vol. 10, no. 8, pp. 707-710, 1996.

[17] Lovins J., "Development of A Stemming Algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11 no. 1-2, pp. 22-31, 1986.

[18] Majumder P., Parui S., Mitra M., Kole G., Mitra P., and Datta K., "YASS: Yet Another Suffix Stripper," *ACM Transactions on Information Systems*, vol. 25, no. 4, pp. 18, 2007.

[19] Melucci M. and Orio N., "A Novel Method for Stemmer Generation Based on Hidden Markov Models," *in Proceedings of the 12th International Conference on Information and Knowledge Management*, New Orleans, pp. 131-138, 2003.

[20] Nabiyev V., *Yapay Zeka Insan Bilgisayar Etkileşimi*, Seçkin Yayıncılık, 2010.

[21] Oflazer K. and Saraçlar M., *Turkish Natural Language Processing*, Springer Link, 2018.

[22] Paice C., "Another Stemmer," *ACM Special Interest Group on Information Retrieval Forum*, vol. 24, no. 3, pp. 56-61, 1990.

[23] Porter M., "An Algorithm for Suffix Stripping," *Program: Electronic Library and Information Systems*, vol. 14, no. 3, pp. 130-137, 1980.

[24] Solak A. and Can F., "Effects of Stemming on Turkish Text Retrieval," *in Proceedings of the Computer and Information Sciences ISCIS'94*, Antalya, pp. 49-56, 1994.

[25] Xu J. and Croft W., "Corpus-Based Stemming Using Co-Occurrence of Word Variants," *ACM Transactions on Information Systems*, vol. 16 no. 1, pp. 61- 81, 1998.

[26] Yucebas S. and Tintin R., "GovdeTurk: A Turkish Stemming Method," *in Proceedings of International Conference on Computer Science and Engineering UBMK-17*, Antalya, pp. 343-347, 2017.

Sait Yucebas Received his MsC in Computer Engineering from Baskent University, Turkey, in 2006. Received his PhD degree in Medical Informatics from Middle East Technical University (METU), Turkey, in 2013. Assistant professor in the Department of Computer Engineering Canakkale Onsekiz Mart University, Turkey. His main research areas are; Natural Language Processing, Artificial Intelligence, Datamining, Big Data Analysis, Medical Informatics, Bio-Informatics.

Rabia Tintin Received her Bachelor's Degree in Computer Engineering from Canakkale Onsekiz Mart University, Turkey, in 2015. Received her MsC in Computer Engineering from Canakkale Onsekiz Mart University, Turkey, in 2018. She is computer engineer in the Department of Student Affairs in Canakkale Onsekiz Mart University, Turkey. Her main research area is; Natural Language Processing.