CMPE 346
Natural Language Processing

Final Project Report

Spell Checker For Turkish Language Using Norvig's Spell Checker Algorithm

By
Group Members
Ali BEYAZ - 113200050
Burak GÜLER - 11575012
Efe HEPDOĞAN - 112200062

Department of Computer Engineering

Faculty of Engineering and Natural Sciences

Istanbul Bilgi University

2020

A spell checker or spell check is a software that searches through texts for correcting errors in spelling. These errors are usually caused by typos or lack of knowledge. Spell checkers are used in different kinds of software for various reasons. In search engines such as Google, Bing or Yandex, spell check is important to find the desired information in an efficient way. In email services such as Gmail, Outlook and Yahoo Mail, spell checkers prevent sending emails containing typos and provide clear communication between individuals. Similar to the e-mail services, chat programs on different platforms like smartphones, PCs and websites benefit from spell check. Another use of spell check is in word processors like Microsoft Word, Apple Pages and LibreOffice Writer. Since those programs are widely used in academic and professional life, spell check is crucial to prevent misspelling that may lead to misunderstandings in reports and documents.

According to Statista, English is the most used language on the internet. Due to its international acceptance, it is expected to see most of the spell checkers to work on English. The way spell checkers work is, by training from a pre-prepared corpus. It trains itself on correct spelling of words and uses it as a reference to detect misspellings in future content. The corpus choice is important to have a strong spell checker and by using corpora of other languages it is possible to have a spell checker for a particular language other than English.

The purpose of this project is to prepare a spell checker for Turkish language by implementing a Turkish corpus to Norvig's algorithm.

Norvig's algorithm takes a text or a list of words and separates them word by word, learning the correct spellings for them. Its "edit1()" function has four different listing techniques that study a word and alter it in order to correct its spelling. These techniques are:

Deletes: It deletes extra letters from the input word regardless of its place.

Transposes: It switches the position of two letters into correct order.

Replaces: It replaces the misspelled letter with the correct one.

Inserts: It adds the missing letter into the correct position.

know() method gives us the list of words that are examined and learned from the given dataset.

```
def known(words):

   "The subset of `words` that appear in the dictionary of WORDS."

   return set(w for w in words if w in WORDS)
```

edits1() method takes a word and finds if it is written correct or not. It runs its four techniques (deletes, transposes, replaces and inserts) to try to find the correct spelling. If the word is not misspelled then it returns it along with other possible known words. In our case, we used a Turkish dataset for words and replaced English characters with Turkish characters in letters of edit1() method.

```
def edits1(word):

   "All edits that are one edit away from `word`."

   letters    = 'abcçdefghıijklmnoöpqrsştuüvwxyz'

   splits     = [(word[:i], word[i:])    for i in range(len(word) + 1)]

   deletes    = [L + R[1:]               for L, R in splits if R]

   transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]

   replaces   = [L + c + R[1:]           for L, R in splits if R for c in
letters]

   inserts    = [L + c + R               for L, R in splits for c in
letters]

   return set(deletes + transposes + replaces + inserts)
```

edits2() takes the output of edits1() and runs them again with edits1(). The output is the list of words that are altered by two more techniques.

```
def edits2(word):

   "All edits that are two edits away from `word`."

   return (e2 for e1 in edits1(word) for e2 in edits1(e1))
```

Testing for the misspelled word “düşüdnüm”, the predicted result was “düşündüm”.

```
[62]  1 known(edits1('düşüdnüm'))

      {'düşündüm', 'düşünüm'}

[63]  1 known(edits2('düşüdnüm'))

      {'düşkündüm',
       'düşkünüm',
       'düşüdür',
       'düşümdüm',
       'düşündü',
       'düşündük',
       'düşündüm',
       'düşündün',
       'düşündür',
       'düşünü',
       'düşünül',
       'düşünüm',
       'düşünümü',
       'düşünün',
       'düşünüp',
       'düşünür',
       'düşünüz',
       'düşünüş',
       'düşürdüm',
       'düşürüm',
       'düşüşüm',
       'üşüdüm',
       'üşündüm'}
```

It transposed the letters “n” and “ü” to find “düşündüm”. Deleted the letter “d” to find “düşünüm”, both are known words.

When the same misspelled word was run in edits2(), we had more results because it applied more corrections on it.

```
unit_tests pass
68% of 146 correct (0% unknown) at 24 words per second
```

Due to the range of our dataset, 146 test cases passed with 68% success rate. This rate can be improved by using a wider-ranging dataset.

In conclusion, Peter Norvig’s spell checking algorithm is an easily understandable, simple but an effective algorithm. Although it is originally written for English language, the way it is written enables it to be implemented to other languages with minor modifications.