spelling correction

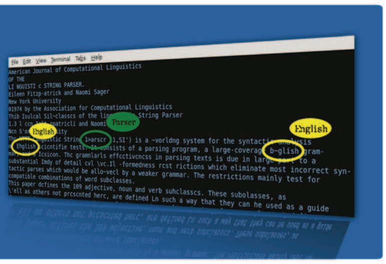
Hailian Jiang, Irene Kolb, Nadya Georgieva

Software Project WS 2010/11

I. Motivation and Goal:

Spelling mistakes can arise, for example, from incorrect encoding of text which ruins special character (a legacy problem). Another source is OCR (Optical Character Recognition)

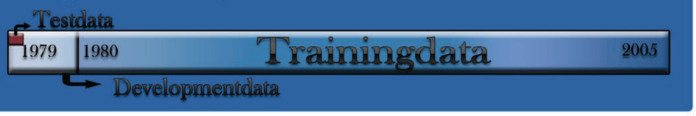
Our project's goal is to improve OCR–Output

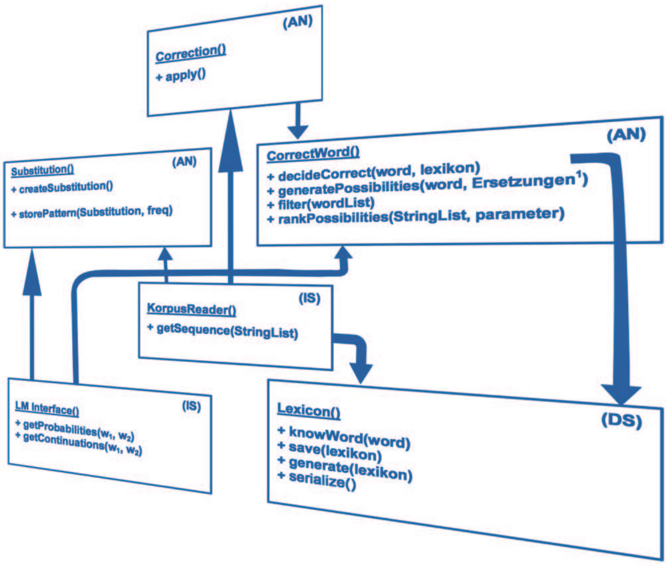


```
American Journal of Computational Linguistics
OF THE
LI NGUISTI c STRING PARSER.
Eileen Fitzp-atrick and Naomi Sager
New York University
01974 by the Association for Computational Linguistics
Thib Isulcal Sil-classcs of the lin      String Parser
1.3 l ccn           zpatricli and Naomi
Ncn S'o        sity
The              ngustic Strin  1>arscr  1,SI') is a ~vorldng system for the syntacti    lysis
o  Ehglish  scicntifie tests. It consists of a parsing program, a large-coverag  b~glish  gram-
mar, and a Icsicon. Thc gramlarls effcctivcncss in parsing texts is due in large part to a
substantial Imdy of detail cvl \vc.Il -formedness rcst rictions which eliminate most incorrect syn-
tactic parses which would be allo~vecl by a weaker grammar. The restrictions mainly test for
compatible combinations of word subclasses.
This paper dcfines the 109 adjective, noun and verb suhclasscs. These subolasses, as
\'ell as others not prcscnted herc, are defined Ln such a way that they can he used as a guide
```

II. Resources:

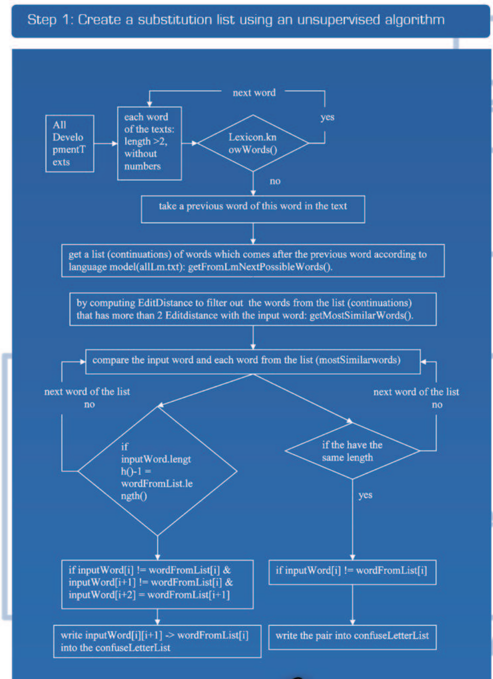
- SRILM (Stanford Research Institute Language Modeling Toolkit)
- JSBD–1.6 (Sentence Splitter Tool)
- ACL ARC (Anthology Reference Corpus)



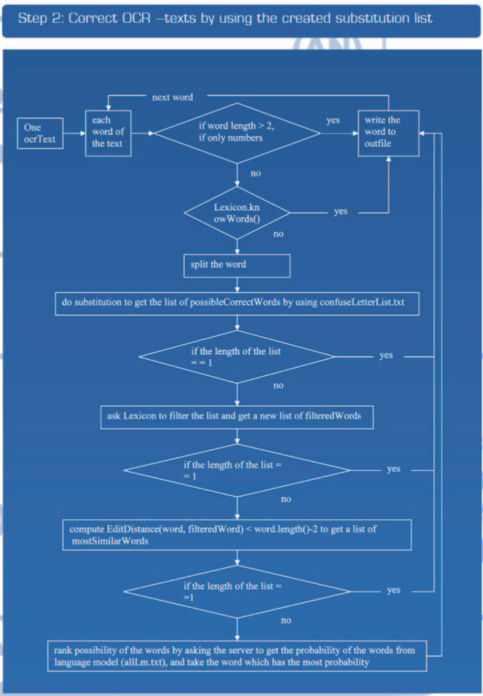

Correction()
+ apply()
(AN)
Substitution()
+ createSubstitution()
+ storePattern(Substitution, freq)
(AN)
CorrectWord()
+ decideCorrect(word, lexikon)
+ generatePossibilities(word, Ersetzungen¹)
+ filter(wordList)
+ rankPossibilities(StringList, parameter)
(AN)
KorpusReader()
+ getSequence(StringList)
(IS)
LM Interface()
+ getProbabilities(w₁, w₂)
+ getContinuations(w₁, w₂)
(IS)
Lexicon()
+ knowWord(word)
+ save(lexikon)
+ generate(lexikon)
+ serialize()
(DS)

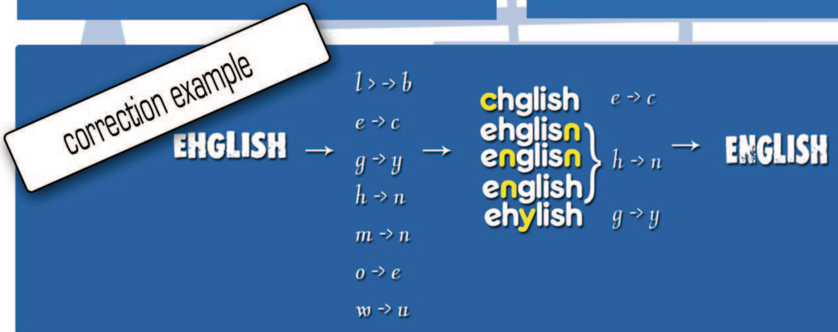
III. The Method:

Step 1: Create a substitution list using an unsupervised algorithm


All DevelopmentTexts
each word of the texts: length >2, without numbers
Lexicon.knowWords()
next word
yes
no
take a previous word of this word in the text
get a list (continuations) of words which comes after the previous word according to language model(allLm.txt): getFromLmNextPossibleWords().
by computing EditDistance to filter out the words from the list (continuations) that has more than 2 Editdistance with the input word: getMostSimilarWords().
compare the input word and each word from the list (mostSimilarwords)
next word of the list
no
next word of the list
no
if inputWord.length()-1 = wordFromList.length()
if the have the same length
yes
if inputWord[i] != wordFromList[i] & inputWord[i+1] != wordFromList[i] & inputWord[i+2] = wordFromList[i+1]
if inputWord[i] != wordFromList[i]
write inputWord[i][i+1] -> wordFromList[i] into the confuseLetterList
write the pair into confuseLetterList

Step 2: Correct OCR –texts by using the created substitution list


One ocrText
each word of the text
next word
if word length > 2, if only numbers
yes
write the word to outfile
no
Lexicon.knowWords()
yes
no
split the word
do substitution to get the list of possibleCorrectWords by using confuseLetterList.txt
if the length of the list = = 1
yes
no
ask Lexicon to filter the list and get a new list of filteredWords
if the length of the list = = 1
yes
no
compute EditDistance(word, filteredWord) < word.length()-2 to get a list of mostSimilarWords
if the length of the list = =1
yes
no
rank possibility of the words by asking the server to get the probability of the words from language model (allLm.txt), and take the word which has the most probability



IV. Evaluation:

n = Number of the letters of each word
m = Number of the possible Substitutions for a letter
s =Number of the letters to be corrected
s ≤ n: Relation between s and n

Formula for the number of the possible corrected words after substitution:

$$(m1+1)*(m2+1)*\ldots*(ms + 1) - 1 = (m+1)s - 1$$

For example: for the word "Granner" : n ==7
If there are substitutions for all letters, then s=n=7
Assume that there are 9 substitutions for each letter,
then the number of the possible correct words is:

$$\boxed{(9+1)^*(9+1)^*\ldots^*(9+1)^*(9+1)-1=10^7-1=10.000.000-1=9{,}999{,}999}$$

In order to generate just one correct word, the computer has to generate 9,999,999 possible combinations between those letters. Further more, since the final result is depend on the value of the possibilities between the words after filter, it is quite possible to generate a word that has nothing to do with the word to be corrected. Therefore, the method is insufficient.

1. The substitutionList is too long: for one letter, there are 50 candidates for substitution.
   There is always outOfMemory Exception.
   Reason: There are many non–words from examples of the articles: For the word "lle", the following mostSimilarWords are generated: [the, old, are, see, lie, oke, she, one, re, cue, lcb, gee, une, due, IXI, ble, PIA, all, BIC, rule, ate, use, are].
2. The substitutionList contains all 26 letters, it means that no matter how many letters are recognized incorrectly by OCR, all letters, both correct and incorrect, of the word to be corrected will be substituted. It is too much unnecessary work.
3. If the substitutionList is reduced to be just 5 candidates for one letter, the program will run endlessly in case that the length of a word is above 8.
4. If the substitutionList is reduced to be just 2 candidates for one letter, the program will run endlessly in case that the length of a word is above 14.

V. Optimization Suggestions:

1. Reduce the substitutionList by using a real dictionary when the mostSimilarWords are generated; the dictionary will filter out those non-words from examples of the article
2. Define a method to filter out the part of the examples from the articles.
3. Not to consider the words that contain more than 10 letters.
4. Not to generate the substitutionList, but directly correct the word after the mostSimilarWords are generated

VI: References:

- Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min–Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev and Yee Fan Tan. 2008. The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics. In Proc. Of Language Resources and Evaluation Conference (LREC 08). Marrakesh, Morocco, May.
- Stolcke, Andreas. SRILM – An Extensible Language Modeling Toolkit. Speech Technology and Research Laboratory SRI International, Menlo Park, CA, U.S.A.

- http://www.free–ocr.com/
- http://www–speech.sri.com/projects/srilm/
- http://en.literateprograms.org/Levenshtein_distance_%28Java%29#chunk%20def:base_conditions
- http://java.sun.com/developer/technicalArticles/Programming/serialization/