HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

BBM203 PROGRAMMING LAB.
ASSIGNMENT 3

Subject : Data Structures and Algorithms
Submission Date : 26.11.2012
Deadline : 11.12.2012, 23:59 pm
Programming Language : JAVA (JDK7)
Advisors : Dr. Sevil ŞEN, Dr. Burcu CAN, R. A. Ali Osman SERHATOĞLU

1. INTRODUCTION / AIM:

In this experiment, you are supposed to develop a simple NLP (Natural Language Processing) program, BrillTagger. Main aim of the experiment is to develop your skill of using Data Structures and Algorithms with Java programming language.

Other aims of the experiment are to write an efficient program, to use exception handling in case of errors occurred during the execution of the program and also to perform file operations by using input/output (io) libraries of Java.

2. BACKGROUND INFORMATION

2.1 NLP (Natural Language Processing)

Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. NLP is known as computational linguistics. Ever wondered how Google Translate works, or how companies do automated resume processing? Want to build a computer that understands language? NLP is related to the area of human–computer interaction[4].

2.2 Hash Table

In computing, a hash table (also hash map) is a data structure used to implement an associative array, a structure that can map keys to values. A hash table uses a hash function to compute an *index* into an array of *buckets*, from which the correct value can be found. Ideally, the hash function should map each possible key to a unique slot index (Figure 1)[3].

In many situations, hash tables turn out to be more efficient than search trees or any other table lookup structure. For this reason, they are widely used in many kinds of computer software, particularly for associative arrays, database indexing, caches, and sets.

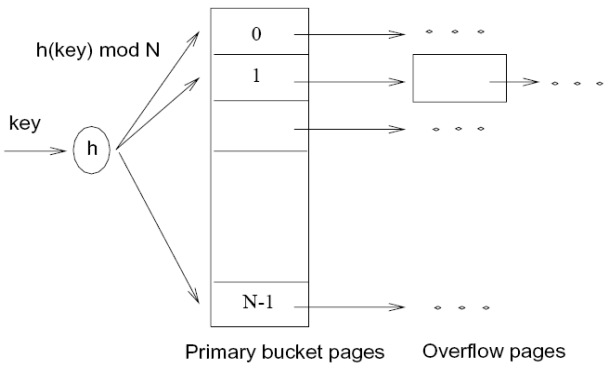

Figure 1: A Hash Table[1]

2.3 File Operations in JAVA

Java provides a rich API for file operations. It has a large library for different file operations. In this experiment you will use IO (Input/Output) classes of Java to perform file operations.

2.4 Brill Tagger

The Brill tagger is a method for doing part-of-speech tagging. It was described by Eric Brill in his 1993 PhD thesis. It can be summarized as an "error-driven transformation-based tagger". It is error-driven in the sense that it recourses to supervised learning transformation-based in the sense that a tag is assigned to each word and changed using a set of predefined rules. Applying over and over these rules, changing the incorrect tags, a quite high accuracy is achieved[6].

2.5 Part Of Speech Tagging

In corpus linguistics, part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context—i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc.

Once performed by hand, POS tagging is now done in the context of computational linguistics, using algorithms which associate discrete terms, as well as hidden parts of speech, in accordance with a set of descriptive tags. POS-tagging algorithms fall into two distinctive groups: rule-based and stochastic. E. Brill's tagger, one of the first and widely used English POS-taggers, employs rule-based algorithms[7].

A simple example: Ben eve gidiyorum.
‘Ben’ : Pronoun
‘eve’ : Noun
‘gidiyorum’ : Verb

2.6 Morphology

In linguistics, morphology is the identification, analysis and description of the structure of a given language's morphemes and other linguistic units, such as root words, affixes, parts of speech, intonation/stress, or implied context (words in a *lexicon* are the subject matter of *lexicology*). Morphological typology represents a method for classifying languages according to the ways by which morphemes are used in a language. Morphology is the study of the way words are built from smaller meaningful units called morphemes[8].

Morphological parsing, in natural language processing, is the process of determining the morphemes from which a given word is constructed. It must be able to distinguish between orthographic rules and morphological rules[9].

A simple example➔ 'kitaplaştırmak': kitap+laş+tır+mak

3. EXPERIMENT

You will write a demo Turkish morphological disambiguator using Brill tagging method. You will use the given corpus in order to create this Brill tagger. You will use some of these hand tagged files as a training set, and some of them as a test set.

3.1 Program

Your program should read and parse all the files in the training set.

Input

```
<S>
Hortumca:2
P:      hortum+Noun+A3sg+Pnon^DB+Adverb+Ly
D:      Hortumca+Noun+Prop+A3sg+Pnon+Nom

Akdenizli:1
P:      akdeniz+Noun+Prop+A3sg+Pnon+Nom^DB+Adj+With
D:      akdenizli+Noun+Prop+A3sg+Pnon+Nom

Yücelen'den:1
P:      Yücelen+Noun+Prop+A3sg+Pnon+Abl
```

"<S>" expression means 'start of sentence'. A sentence starts with "<S>" and ends with "<S>" expression of the other one.

'2' means "correctly tagged morphological parse" in "Hortumca:2" expression.

"Hortumca" is a word.

"hortum+Noun+A3sg+Pnon^DB+Adverb+Ly" is a morphological parse of "Hortumca".

Your program should read and parse all the files in the training set. Your program should create the following two tables, and write them into two files.

MostLikelyMorphParseForWord

For each word in the training set, *MostLikelyMorphParseForWord* table should hold how many times that word occurred in the training set, and which morphological parse of that word is most likely. Your program should also collect how many times each morphological parse of each word is selected as a correct parse.

Write the contents of this table into a text file (MostLikelyMorphParseForWord.txt) in the descending order (the first word in this file is the word with the highest frequency in the training set). On the first line of this file, put the total number of the words and the number of the unique words in the training set. For each word, put the following lines into this file.

word	*its frequency (how many times this word occurred in the training set)*
mostlikelymorphparse	*its frequency (how many times this MP as marked as correct)*
2ndmostlikelymorphparse	*its frequency (how many times this MP as marked as correct)*
…	
Leastlikelymorpparse	*its frequency (how many times this MP as marked as correct)*

Output Example:

```
-----------------------------------Start of Page-----------------------------------
Number Of Words:5279    Number Of Unique Words:2553
.:    0.08315968933510134(439)
.+Punc:     1.0(439)

,:    0.05550293616215192(293)
,+Punc:     1.0(293)

ve:   0.017238113279030118(91)
ve+Conj:    1.0(91)

bir:  0.015912104565258572(84)
bir+Num+Card:     1.0(84)
bir+Noun+A3sg+Pnon+Nom: 0.0(0)
bir+Adverb: 0.0(0)
bir+Adj:    0.0(0)
bir+Noun+Prop+A3sg+Pnon+Nom:  0.0(0)

":    0.010986929342678538(58)
"+Punc:     1.0(58)

':    0.008145482098882365(43)
'+Punc:     1.0(43)

-----------------------------------End of Page-------------------------------------
```

MostLikelyTag

For each part of speech tag in the training set, *MostLikelyTag* table should hold the frequency of that tag. We will use the last inflections of the words as part of speech tags. You should extract a part of speech tag from a morphological parse as follows.

a) If the morphological parse does not contain a derivational boundary[1] (^DB), except the root word assume that the rest of the morphological parse as a part of speech tag. See the following examples.

kişi+Noun+A3sg+P2sg+Gen➔Noun+A3sg+P2sg+Gen
60+Num+Card➔Num+Card
böyle+Adverb➔Adverb
.+Punc➔Punc
afgan+Noun+Prop+A3sg+Pnon+Nom➔Noun+Prop+A3sg+Pnon+Nom
uy+Verb+Pos+Past+A3sg➔Verb+Pos+Past+A3sg

b) If the morphological parse contains a derivational boundary (^DB), assume that the inflection after the last derivation boundary by dropping the type of the derivation as a part of speech tag. See the following examples.

ol+Verb+Pos^DB+Adverb+ByDoingSo➔Adverb
sağla+Verb+Pos^DB+Adj+PresPart➔Adj
öl+Verb+Pos^DB+Adj+PastPart^DB+Noun+Zero+A3sg+Pnon+Acc➔Noun+A3sg+Pnon+Acc
bombala+Verb^DB+Verb+Caus+Pos^DB+Noun+Inf1+A3sg+Pnon+Nom➔Noun+A3sg+Pnon+Nom
kes+Verb^DB+Verb+Caus+Neg+Imp+A2sg ➔ Verb+ Neg+Imp+A2sg
de+Verb^DB+Verb+Pass^DB+Verb+Able+Pos+Fut+A3sg➔Verb+Pos+Fut+A3sg

Write the contents of this table (tags together with their frequencies) into a text file (MostLikelyTag.txt) in the descending order (the first line in this file should be the highest frequency tag in the training set).

Output Example:

```
-------------------------------------Start of Page-------------------------------------
Number Of Tags:185
Punc: 0.16139420344762265(852)

Noun+A3sg+Pnon+Nom:     0.1392309149460125(735)

Adj:  0.11100587232430384(586)

Noun+Prop+A3sg+Pnon+Nom:      0.06459556734229968(341)

-------------------------------------End of Page---------------------------------------
```

Rules

Rule Templates:

A rule template should be one of the following two templates:

Select TAGa for WORDn if the tag of WORDn-1 is TAGb
Select TAGa for WORDn if the tag of WORDn+1 is TAGb

where TAGa and TAGb are possible tags from *MostLikelyTag* table.

[1] When combined with a root, change either the semantic meaning or part of speech of the affected word[10](ie. kitap➔kitapçı).

Select TAGa for WORDn if *Condition* is that we select the morphological parse with TAGa for WORDn if *Condition* is satisfied and TAGa is the tag of at least one of the morphological parses of WORDn. If there is more than one morphological parses with TAGa (belongs to that word) select the one with the highest frequency. If WORDn does not have a morphological parse with TAGa (or *Condition* is not satisfied), the rule does not have any effect on WORDn.

Output Example("rules.txt"):

```
Select Conj for WORDn if the tag of WORDn-1 is Noun+Prop+A3sg+Pnon+Nom.
Select Pron+A3sg+Pnon+Nom for WORDn if the tag of WORDn+1 is Conj.
Select Postp for WORDn if the tag of WORDn+1 is Punc.
Select Pron+A3sg+Pnon+Nom for WORDn if the tag of WORDn+1 is Punc.
Select Postp for WORDn if the tag of WORDn+1 is Adverb.
```

TestSetResults

Apply all rules on TestSet. You have to do the followings.

a) Mark each word W on TestSet with its most likely morphological parse. The most likely morphological parse of a word in TestSet is found as follows:

- If the word W has a most likely morphological parse in *MostLikelyMorphParseForWord* table, select that one as the most likely morphological parse for the word W.
- Otherwise, if the tag of at least one of morphological parses of the word W appears in *MostLikelyTag* table, select the morphological parse with the tag whose value is the maximum value in *MostLikelyTag* table.
- Otherwise, select the first morphological parse as the most likely morphological parse for the word W.

Find the precision value of the test set with most likely morphological parses, and write into a text file ("testsetresults.txt").

The precision value is evaluated as follows:

Precision = Number of Correctly Tagged Words / Number of Total Words

where Number of Correctly Tagged Words is the number of correctly tagged words in the test set and Number of Total Words is the number of the words in the data set.

b) Apply all the rules one by one to the test set. You only should apply rules on sentences (check input section of this document). After the application of each rule, find the precision value for the test set, and record the rule together with the precision value into the file ("testsetresults.txt").

Output Example("testsetresults.txt"):

```
0.8579662412515439
Select Conj for WORDn if the tag of WORDn-1 is Noun+Prop+A3sg+Pnon+Nom.
0.8583779333058872
Select Pron+A3sg+Pnon+Nom for WORDn if the tag of WORDn+1 is Conj.
0.8587896253602305
Select Postp for WORDn if the tag of WORDn+1 is Punc.
0.8587896253602305
Select Pron+A3sg+Pnon+Nom for WORDn if the tag of WORDn+1 is Punc.
0.8596130094689173
Select Postp for WORDn if the tag of WORDn+1 is Adverb.
0.8600247015232606
```

3.2 Execution Of The Program

Command-Line Arguments:

Main <trainingsetnum> <testsetnum> <rulesfile> <testresultfile>

<trainingsetnum>➔ number of files in "TrainingSet" folder (ie. s1.txt,s2.txt)
<testsetnum> ➔ number of files in "TestSet" folder (ie. s1.txt,s2.txt)
<rulesfile> ➔ rules' file name
<testresultfile> ➔ testresults' file name

Usage Example: Main 2 1 rules.txt testresults.txt

You can see sample input and output in ftp site.

3.3 Bonus Of Experiment

If your program learns the rules from the training set, you will take extra 10 point. According to Brill tagging method, the learning of the rules is done as follows.

a) Tag all the words in the training data set with their most likely morphological parses. The most likely morphological parse of a word is found as follows:

- If the word W has a most likely morphological parse in *MostLikelyMorphParseForWord* table, select that one as the most likely morphological parse for the word W.
- Otherwise, if the tag of at least one of morphological parses of the word W appears in *MostLikelyTag* table, select the morphological parse with the tag whose value is the maximum value in *MostLikelyTag* table.
- Otherwise, select the first morphological parse as the most likely morphological parse for the word W.

b) Find the precision value for the data set with most likely morphological parses (call this data set as DS1). The precision value is evaluated as follows:

Precision = Number of Correctly Tagged Words / Number of Total Words

where *Number of Correctly Tagged Words* is the number of correctly tagged words in the data set (here the data set is the training set with most likely morphological parses), and *Number of Total Words* is the number of the words in the data set.

c) Select a rule R (among all possible rules) such that the rule R will give biggest precision value increase if that rule is applied to the data set DSn in order to get the data set DSn+1. In other words, you have to try each of the possible rules, and select the one that gives the biggest precision-increase. Find the precision of the data set DSn+1 (the data set after the rule is applied to the data set DSn). Record your rule together with the precision value of the data set DSn+1.

d) Repeat step (b) in order to learn at least 10 rules, and save all the rules (in the learning order) into a rule table.
e) Write the precision value of the data set with the most likely morphological parses (the data set DS1), and all the rules in the rule table together with their precision values into a text file.

Execution Of The Program

Command-Line Arguments:

Main <trainingsetnum> <testsetnum> <rulesfile> <testresultfile> -r <numofrule>

<trainingsetnum>➔ number of files in "TrainingSet" folder (ie. s1.txt,s2.txt)
<testsetnum> ➔ number of files in "TestSet" folder (ie. s1.txt,s2.txt)

<rulesfile> ➔ rules' file name
<testresultfile> ➔ testresults' file name
<numofrule> ➔ number of rules which will be extracted from corpus

Usage Example: Main 2 1 rules.txt testresults.txt –r 10

3.3 Your Report

You ***also*** should evaluate time-space complexity in your report. (Report Format: ftp://ftp.cs.hacettepe.edu.tr/pub/dersler/genel/FormatForLabReports.doc)

LAST REMARKS:

- Use Eclipse IDE for JAVA Developers as development environment. Any other development enviroments will not be accepted!
- Regardless of the length, use **UNDERSTANDABLE** names to your variables, classes and functions.
- Write **READABLE SOURCE CODE** block
- Your submission code file structure is going to be announced later.
- **You will use online submission system to submit your experiments. https://submit.cs.hacettepe.edu.tr/ Deadline is: 23:59 pm. No other submission method (such as diskette, CD or email) will be accepted.**
- Do not submit any file via e-mail related with this assignment.
- **SAVE** all your work until the assignment is graded.
- The assignment must be original, **INDIVIDUAL** work. Duplicate or very similar assignments are both going to be punished. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms or source codes.
- You can ask your questions through course's news group and you are supposed to be aware of everything discussed in the newsgroup: **news://news.cs.hacettepe.edu.tr/dersler.bbm203**

REFERENCES

1. R. Ramakrishnan, J. Gehrke, Database Management Systems, Second Edition, The MIT Press
2. H. M. Deitel, P. J. Deitel, Java How to Program, Prentice Hall
3. http://en.wikipedia.org/wiki/Hash_table
4. http://en.wikipedia.org/wiki/Natural_language_processing
5. http://www.eclipse.org/downloads/
6. http://en.wikipedia.org/wiki/Brill_tagger
7. http://en.wikipedia.org/wiki/Part-of-speech_tagging
8. http://en.wikipedia.org/wiki/Morphology_(linguistics)
9. http://en.wikipedia.org/wiki/Morphological_parsing
10. http://en.wikipedia.org/wiki/Morpheme