UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B.Sc. Examination 2012

ALL COMPUTING PROGRAMMES

IS53036A
Introduction to Natural Language Processing

Duration: 2 hours 15 minutes

Date and time: 12th January 2012 2pm

There are FIVE questions in this paper. You should answer THREE of them. Each question is marked out of 25. The marks for each part of a question are indicated at the end of the part in [.] brackets.

No calculators should be used.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

QUESTION 1

Tables 1 and 2 below show a selection of past tense forms in English.

Table 1

Present	**Past**
bear	bore
bend	bent
creep	crept
keep	kept
lay	laid
lend	lent
pay	paid
pity	pitied
rely	relied
reply	replied
say	said
send	sent
sleep	slept
study	studied
swear	swore
tear	tore
wear	wore

Table 2

Present	**Past**
bind	bound
clear	cleared
fear	feared
find	found
hear	heard
mend	mended
play	played
seep	seeped
tend	tended

a) Give a list of rules for deriving past tense forms based on the cases in **Table 1** only. You should justify your answer with reference to specific examples. **[6]**

b) Revise your list of rules so that it covers all cases in both **Table 1** and **Table 2**.

[9]

c) The following Python function implements a highly simplified past tense morphology for English:

```
def past(verb):
        if verb == 'goes':
                return('went')
        elif verb == 'can':
                return('could')
        elif verb[-1] == 'e':
                return(verb+'d')
        else:
                return(verb+'ed')
```

Modify the function so that it correctly handles the past tense forms in **Tables 1** and **2**, following the rules you have given in your answers to (a) and (b). **[6]**

d) Let a1 and b1 be Python variables as follows:

```
a1 = ‘able baker charlie delta echo’
b1 = [‘able’, ‘baker’, ‘charlie’, ‘delta’, ‘echo’]
```

Show the results of executing the following Python commands. You should explain your answers. **[4]**

i. a1[2:4]
ii. b1[2:4]
iii. ‘able’ in a1
iv. ‘able’ in b1
v. ‘lie’ in a1
vi. ‘lie’ in b1
vii. a1[3]+a1[1]
viii. b1[3]+b1[1]

QUESTION 2

Consider the following grammar rules and answer questions (a – c) following.

S → NP VP
NP → Det N
NP → Det N PP
VP → V NP
VP → Cop Adj
PP → P NP

Det → a | the | my | twelve | some | most
N → friend | coat | sales | buttons | shoe | shoes | car | cars | student | students
V → bought | likes | like | has | have
Cop → is | are | was | were
P → in | with | of
Adj → happy | angry | sad

a) Explain what is meant by the following, with some simple examples:
   i. Recursive grammar rules
   ii. Attachment ambiguity
   iii. Left-recursion
   iv. Centre-embedding
   v. Recursive descent parsing **[9]**

b) Using the above grammar rules, draw syntax trees for:
   i. My friend is angry
   ii. My friend bought a coat with twelve buttons
   iii. Some students have cars **[6]**

c) Modify the grammar using **feature structures** so that it generates the unstarred sentences below as well as (2b)(i-iii) above but not the starred ones. Explain the reasons for your modifications. **[10]**

   i. My friend is a student
   ii. My friends are students
   iii. Most coats have buttons
   iv. * Most coats have button
   v. * The student are my friend.
   vi. * Twelve student bought a car.
   vii. * The students bought some shoe.

QUESTION 3

The following paragraph is taken from a government report, *How Fair is Britain?*

“Today, we live in a society where overt displays of prejudice are usually unlawful, and almost always socially unacceptable. Surveys suggest that we are more tolerant of difference, and less tolerant of discrimination. This is mirrored in the evolution of new laws which prohibit discrimination and require public bodies to promote equality. It is borne out by our expectations of public figures: a career in the public eye can be cut short by a bigoted comment.”

This is the result of running the above text through the Lancaster Stemmer:

['today', ',', 'we', 'liv', 'in', 'a', 'socy', 'wher', 'overt', 'display', 'of', 'prejud', 'ar', 'us', 'unlaw', ',', 'and', 'almost', 'alway', 'soc', 'unacceiv', '.', 'survey', 'suggest', 'that', 'we', 'ar', 'mor', 'tol', 'of', 'diff', ',', 'and', 'less', 'tol', 'of', 'discrimin', '.', 'thi', 'is', 'mir', 'in', 'the', 'evolv', 'of', 'new', 'law', 'which', 'prohibit', 'discrimin', 'and', 'requir', 'publ', 'body', 'to', 'promot', 'eq', '.', 'it', 'is', 'born', 'out', 'by', 'our', 'expect', 'of', 'publ', 'fig', ':', 'a', 'car', 'in', 'the', 'publ', 'ey', 'can', 'be', 'cut', 'short', 'by', 'a', 'bigot', 'com', '.',]

a) Explain what is meant by **word stems**, with references to examples from the above text. How can stemming be useful in applications such as machine translation? **[4]**

b)
   i. Make a list of rules which the stemmer has applied in this example.
   ii. Discuss the motivations for the rules including any cases in this example where you believe rules have been applied inappropriately. **[12]**

c) The following Python code implements a very simple stemmer which simply removes the endings ‘-e’, `-s' or `-es' if present, except for the words ‘be’ and 'is':

```
def stemmer(word):
        [(s,end)] = re.findall('^(be|is|.*?)(e|s|es)?$',word)
        return s
```

i. Explain the meaning of the regular expression in this example. What will be the result of stemming the word “loves”? Explain your answer.
ii. Write a regular expression to replace the one in this function, which implements the rules you have listed in your answer to (b). Discuss any cases which you cannot straightforwardly encode as a regular expression. **[9]**

QUESTION 4

a) Explain the difference between **information retrieval** and **information extraction**. **[3]**

b) Describe and give examples of the classes of strings matched by the following regular expressions: **[6]**

   i. [A-Z0-9]+
   ii. [A-Z][0-9]*
   iii. ([aeiou][^aeiou])*

c) The NLTK findall() method allows you to search for sequences of **tokens** in a text and choose what part of the result to display, enabling you to find instances of strings occurring in particular contexts. For example, assuming news is a tokenised text:

```
>>>  news.findall(r"<President><K.*>")
President Kennedy; President Kasavubu; President Kennedy's;
President Krushchev …
>>>  news.findall(r"<President>(<K.*>)")
Kennedy; Kasavubu; Kennedy's; Krushchev …
```

Construct search patterns to find the following types of phrase. You should think about what kinds of context these expressions can occur in. You may give more than one pattern for each answer. Explain your answers.

   i. Personal names of the form *firstname*, *lastname*: *Daniel Radcliffe*, *Nitin Sawhney*, *Barack Obama* etc.
   ii. First names which are more commonly male: *John, Abdul, Ivan*
   iii. First names which are more commonly female: *Laura, Fatima, Mary*
   iv. Names of streets: *Downing Street*, *St James, Pennsylvania Avenue*
   v. Place names like *Paris*, *Gaza, Germany, Tibet, New York.*

[10]

d) Explain what is meant by **precision** and **recall** in the context of information retrieval and extraction. What kinds of **false negatives** and **false positives** might result from the queries you specified in your answers to (4c) above?

[6]

QUESTION 5

a) The following sentences are all ambiguous in some way. Express their different meanings using paraphrases and explain the source of the ambiguity in each case. **[5]**
   i. The builders dumped the rubble in the skip
   ii. Every man loves a woman
   iii. Flying planes can be dangerous
   iv. Either Kim will stay or Dana will leave and everyone will be happy.
   v. Whenever John meets Bill he buys him a drink.

b) Explain how probabilistic grammars can deal with ambiguity, with reference to one of the examples above. **[6]**

c) The corpora that are distributed with the NLTK can be accessed in various ways, including:
   i. Raw text
   ii. Tokenised
   iii. POS-tagged
   iv. Parsed

   Explain the distinctions between these terms and give an example of a corpus that includes parsed text. **[5]**

d) The following is part of the output of a **bigram tagger**, tested on an excerpt from the Brown corpus news genre:

```
[('Then', 'RB'), ('Dick', 'NP'), ('Hyde', 'NP'), (',', ','), ('submarine-ball',
'NN'), ('hurler', 'NN'), (',', ','), ('entered', 'VBD'), ('the', 'AT'), ('contest',
'NN'), ('and', 'CC'), ('only', 'AP'), ('five', 'CD'), ('batters', 'NNS'),
('needed', 'VBD'), ('to', 'TO'), ('face', 'VB'), ('him', 'PPO'), ('before',
'IN'), ('there', 'RB'), ('existed', None), ('a', None), ('3-to-3', None),
('deadlock', None), ('.', None)]
```

   i. What is a probable reason that the last six tokens have been assigned the tag "None"?
   ii. Explain how more accurate tagging can be achieved by using a procedure that includes multiple taggers. **[9]**

APPENDIX: REGULAR EXPRESSIONS

.	Wildcard, matches any character
^abc	Matches some pattern abc at the start of a string
abc$	Matches some pattern abc at the end of a string
[abc]	Matches one of a set of characters
[A-Z0-9]	Matches one of a range of characters
ed|ing|s	Matches one of the specified strings (disjunction)
*	Zero or more of previous item, e.g. a*, [a-z]*
+	One or more of previous item, e.g. a+, [a-z]+
?	Zero or one of the previous item (i.e. optional), e.g. a?, [a-z]?
*?	“Non-greedy” star operator
a(b|c)+	Parentheses that indicate the scope of the operators