INTRODUCTION TO NATURAL LANGUAGE PROCESSING

SoSe 2025

ASSIGNMENT -5

Submitted by,

Group – 30

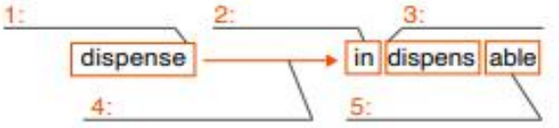
Arundas Mohandas (127115)

Ganga Sunil (127161)

Glen Paul Chirayth Shaju (127125)

Exercise 1 : Morphology (2+2+1+3+1=9 Points)

(a) Fill in the blanks in the following illustration with the correct morphological terms from the given set.



derivation prefix suffix
inflection stem infix
morpheme lexeme root lemma
affix

(b) What affixes are in the word "reactors"? Identify whether each affix is *derivational* or *inflectional*.

(c) Using the word "reactors" explain the difference between *root* and *stem*.

(d) You are given the following excerpt of rules from the Porter stemmer.

Index	Ruleset	Premise	Suffix	Replacement
(I)	1a	null	SSES	SS
(II)	1b	(*v*)	ING	null
(III)	1b	(*v*)	IZ	IZE
(IV)	1c	(*v*)	Y	I
(V)	2	(m>0)	BILITI	BLE
(VI)	2	(m>0)	IVENESS	IVE
(VII)	2	(m>0)	IZATION	IZE
(VIII)	3	(m>0)	NESS	null
(IX)	4	(m>1)	AL	null
(X)	4	(m>1)	IVE	null
(XI)	4	(m>1)	ABLE	null
(XII)	4	(m>1)	ITI	null
(XIII)	4	(m>1)	IZE	null
(XIV)	5	(m>1)	E	null

Stem the following words using the Porter stemmer with the rules given above. Note down the index of the rules you apply in order.

1. recognizability
2. recognization
3. recognizing
4. universalness
5. universe
6. university

(e) Are any of the words in the previous exercise under- or over-stemmed? Which problems (if any) can arise if the words are under- or over-stemmed?

Answer:

a) 1: derivation
2: prefix
3: suffix
4: root
5: stem

b) Prefix: re- = derivational

Suffix: -s = inflectional

c) The **root** of reactors is ***act***, which carries the core meaning. The **stem** is ***reactor***, which is the form to which the inflectional suffix *-s* is added. So, root is the core word, while stem includes derivational affixes but not inflectional ones.

d) Stemming the words using the Porter stemmer rules:

1. **recognizability**
    - Apply (XII): ABILITY → null → *recogniz*
    - Apply (III): IZ → IZE → *recognize*
    - Apply (XIV): E → null → *recogniz*

Final stem: **recogniz**
Rules used: (XII), (III), (XIV)

2. **recognization**
    - Apply (VII): IZATION → IZE → *recognize*
    - Apply (XIV): E → null → *recogniz*

Final stem: **recogniz**
Rules used: (VII), (XIV)

3. **recognizing**
    - Apply (II): ING → null → *recogniz*
      (Since rule (III) is not applicable anymore due to context, no IZE step here.)

Final stem: **recogniz**
Rules used: (II)

4. **universalness**
    - Apply (VIII): NESS → null → *universal*

Final stem: **universal**
Rules used: (VIII)

5. **universe**
   No applicable rule directly matches *universe*. No stemming is performed.

Final stem: **universe**
Rules used: None

6. **university**
    - Apply (XIII): ITI → null → *univers*

Final stem: **univers**
Rules used: (XIII)

e) Under-stemming occurred with *universe* because no rule was applied even though it shares a base with *university*. Over-stemming happened with *university* → *univers*, which may lose

meaning. *Recognizability*, *recognization*, and *recognizing* all reduce to *recogniz*, which is acceptable unless the nuance of the original word is needed.

Over-stemming can merge unrelated words into the same stem, causing ambiguity. Under-stemming, on the other hand, may fail to group similar words, affecting tasks like search or clustering. 0.5P

Exercise 2 :

Exercise 2 : Word Classes & Named Entities (2+1+1+1=5 Points)

You are given the following sentence:

Tokens	He	works	at	the	corporate	headquarters	of	Microsoft	in	Redmond	United	States	.
UD POS													
PENN POS													
NE BIO													
NE BIO													

(a) Annotate the word classes of the sentence using the Penn Treebank and Universal Dependency (UD) tagsets.

(b) What is the major difference between the Penn and the UD tagset? Why were the changes from Penn to UD necessary for UD's mission?

(c) Convert the following recognized spans into the BIO tagging format:

He works at the corporate headquarters of [Microsoft]$_{\text{ORG}}$ in [Redmond]$_{\text{Loc}}$ [United States]$_{\text{Loc}}$.

(d) The IO tagging scheme is a simpler version of BIO. Tag the above-given sentence with IO scheme. What specific information is lost when using IO instead of BIO?

2P a) Annotated table for the sentence " He works at the corporate headquarters of Microsoft in Redmond United States ." is as shown below. Each word of this sentence is broken down into tokens and are annotated using Penn Treebank and Universal Dependency (UD) tag sets.

Token	Penn POS	UD POS
He	PRP	PRON
works	VBZ	VERB
at	IN	ADP
the	DT	DET
corporate	JJ	ADJ
headquarters	NNS	NOUN
of	IN	ADP
Microsoft	NNP	PROPN
in	IN	ADP

Redmond	NNP	PROPN
United	NNP	PROPN
States	NNPS	PROPN
.	.	PUNCT

b) The Penn Treebank tag set is highly detailed and was designed specifically for English. It includes a wide range of part-of-speech tags like VB for base verbs, VBD for past tense, NN for singular nouns, NNS for plural nouns etc. This level of granularity makes Penn Treebank very precise for English but also makes it language-specific and harder to adapt to other languages with different grammatical structures.

In contrast, the Universal Dependencies (UD) tag set was created to support multilingual natural language processing. It uses a coarser and more generalized set of POS tags, such as: VERB for all types of verbs, regardless of tense or number, NOUN for common nouns etc. The goal of UD is to standardize POS tagging across languages, enabling cross-lingual tools, consistent syntactic parsing, and interoperability between linguistic resources.

The changes from Penn to UD was necessary for UD's mission because UD was created for cross-linguistic consistency, easier parsing across languages, and simplified annotation schemes while Penn is very English-specific.

c)

Token	**Named Entity BIO**
He	O
works	O
at	O
the	O
corporate	O
headquarters	O
of	O
Microsoft	B-ORG
in	O
Redmond	B-LOC
United	B-LOC
States	I-LOC
.	O

d)

Token	**Named Entity IO**
He	O
works	O
at	O
the	O
corporate	O
headquarters	O

of	O
Microsoft	I-ORG
in	O
Redmond	I-LOC
United	I-LOC
States	I-LOC
.	O

IO does not distinguish between the beginning and continuation of entities. Also we loose the exact boundaries of multi-token entities, like in this case “United States”.

Exercise 3 :

Exercise 3 : Word Classes: Part-of-Speech Tagging & Ambiguity (4+1+2=7 Points)

(a) Suppose that you want to build a part-of-speech tagger. An example input x and output y is:

Input x =	I	ate	the	fish	in	the	freezer
Output y =	PRON	VERB	DET	NOUN	ADP	DET	NOUN

Your goal is to build a probabilistic model $P(y|x)$ that predicts the part-of-speech tags y given the input x.

(a1) How would you model this problem using a Transformer? In your answer, state which archetype discussed in the lectures is suitable for this task and sketch the architecture for this example.

(a2) Write down the probability distribution that your Transformer models.

(b) The word “`fast`” can be an adjective (JJ), noun (NN), adverb (RB), or verb (VB). Create a short, unambiguous sentence for each of these four-word classes.

(c) Word Sense Disambiguation (WSD) is a task in NLP that aims to determine which sense of a word is used in a given context [NLP:VII-11].

Assume that you are building a WSD model that represents the context of each occurrence of the target word as a vector, consisting of the lemmas of the previous two open class words and the lemmas of the next two open class words, in that order. Then if the target word is “`market`”, what would the vector be for representing its context in the following sentence?

```
The stocks went up as much as the market expected them in the current climate
```

a1) This problem can be solved well using a Transformer encoder, especially one like BERT. In part-of-speech tagging, it's important to know the meaning of each word based on the whole sentence. Transformer encoders are good at this because they look at all the words in the sentence, not just the ones before or after. This helps the model understand how a word is used. For example, the word "fish" can be a noun or a verb, and the model can tell the difference by looking at the words around it.

The architecture overview would be:

- **Input Embedding Layer**:

Each word in the input sentence is first turned into a vector using a token embedding, which maps words to numerical representations based on a vocabulary. To help the model understand the order of words in the sentence, positional encoding is added to these vectors. For example, a sentence like “I ate the fish in the freezer” would be transformed into a sequence of vectors, one for each word, such as

$[x_1, x_2, x_3, x_4, x_5, x_6, x_7]$.

- **Transformer Encoder Layers**:

The Transformer encoder has layers that help the model focus on all the words in the sentence and understand how they are connected. These layers also adjust the information to make it more useful for the task. In the end, each word gets a meaning that depends on the full sentence.

- **Classification Layer**:

For each word’s final representation (after understanding the sentence), a simple classifier, typically followed by a softmax layer, is used to predict its part-of-speech tag. The softmax layer converts the raw logits into a probability distribution, and the model predicts the most likely tag (e.g., noun or verb) for each word.

Output: y_1 = POS tag for x_1, ... , y_7 = POS tag for x_7

The output tag for the sentence "I ate the fish in the freezer" will be
[PRON VERB DET NOUN ADP DET NOUN]

2P

Correct, but encoder-decoder would be a better choice

a2) The Transformer model aims to model the conditional probability distribution of the POS tags, given an input sentence.

Let x=(x1,x2,…,xn) represent the input word sequence for which y=(y1,y2,…,yn) represent the corresponding output POS tag sequence.

The modelled distribution looks like :

$$P(y|x) = \prod_{i=1}^{n} P(y_i|x)$$ ✓ 1P

This implies that, given the entire input sequence, the POS tag at each position is conditionally independent of the others, which is made possible by the self-attention mechanism in the Transformer. To model each individual probability $P(y_i|x)$, the Transformer uses a SoftMax function, where the input is a linear transformation of the contextualized representation of the i^{th} word derived from the Transformer encoder.

not agreed

The resulting output is passed through the SoftMax function to predict the POS tag for that word.

b) An example for the POS tag JJ is, she bought a fast car. Here fast is adjective as it describes the car.

An example for the POS tag NN is, Her fast lasted for three days. Here fast is noun as it refers to a period of not eating.

An example for the POS tag RB is, He drives fast on the highway. Here fast is adverb as it modifies the verb *drives*.

An example for the POS tag VB is, Many people fast during the holy month. Here fast is verb as it shows the action of abstaining from food.

c) The sentence given is, “The stocks went up as much as the market expected them in the current climate.” Here we need 2 lemmas before and 2 after the word “market” which should be open class words. So, the first step here would be to tokenize and find open class words. Looking at the two words before "market," we identify "stocks" (a noun) and "went" (a verb), whose lemmas are "stock" and "go," respectively. After "market," the next open-class words are "expected" (a verb) and "current" (an adjective), with lemmas "expect" and "current." Although the word "them" follows "expected," it is a pronoun and therefore not considered an open-class word.

Thus, the final vector representing the context around "market" is: [stock, go, expect, current].