Regular Expression

- Course Instructor: Tanmoy Chakraborty (NLP, Social Computing)

  tanchak@iitd.ac.in
- Guest Lecture: TBD
- Course page: https://lcs2.in/nlp2402
- Piazza: http://piazza.com/iit_delhi/winter2025/deeplearningfornaturallanguageprocessing

  [Code: **rd6ikjkzp7m**]
- TAs:
  - Sahil Mishra*, Aswini Kumar Padhi, Anwoy Chatterjee, Vaibhav Seth
- Group Email:

Course Directives

- **Class Time:** Mon & Thu, 2 pm – 3:30 pm
- **Office Hour:** Mon 5-6 pm
- **Room:** LH-308

HashLearn

- Meet your instructor at least once per 15 days to resolve your doubts.
- Mon 5-5:30 pm **(appointment based, email me at least 1 hr before coming)**

Marks distribution (tentative):

- Minor: 20%
- Major: 30%
- Quiz (3): 15%
- Assignment (2): 15%
- Mini-project: 18% (group-wise)
- Paper reading (1): 2% (group-wise)

- **Audit**: Discouraged!
  B- (threshold to pass the course)
- **Grading Scheme**: Relative?
- 75% attendance **mandatory**
  - If <75%, one grade down

Content (Tentative)

- Introduction

Classical NLP (1980-2010)

- Regular Expressions, Text Normalization, and Edit Distance
- Morphology & Finite-state Transducers
- N-grams, smoothing and entropy
- HMM, Viterbi and A* decoding
- Word classes and POS tagging
- Semantics & distributional semantics

Neural NLP (2011 - 2017)

- Word vectors and word window classification (Word2Vec, GloVe, etc.)
- RNNs and language models (vanishing gradients, fancy RNNs)
- Sequence-to-sequence models and applications
- Attention mechanisms & self-attention
- Transformers

LLMs (2018 – till date)

- More about Transformers (BERT, RoBERTA, ELMo, transfer learning)
- Prompting and In-context learning
- Alignment – RLHF, PPO, DPO
- Efficient LLMs
- LLM Agents
- Fairness and ethics in NLP

- For ScAI students: Application bucket
- For MT students: DE

- Decide if you want to take this course ASAP – Many students are waiting
  - Would be happy to increase the class size limit, if needed

- Finish Piazza enrolment
- Start forming your group for the course project

- Start learning Deep Learning, assuming you know ML well

Recap of the last lecture

NLP layers

- Understanding the semantics is a non-trivial task.
- Needs to performs a series of incremental tasks to achieve this.
- NLP happens in layers

Pragmatics & Discourse	*Study of semantics in context.*
Semantics	*Meaning of the sentence.*
Parsing	*Syntactic structure of the sentence.*
Chunking	*Grouping of meaningful phrases.*
Part of speech tagging	*Grammatical classes.*
Morphology	*Study of word structure.*

Increasing
Complexity Of
Processing

Regular Expression

The most important tool for describing text patterns

used to specify strings we might want to extract from a document

Regular Expression (RE)

- A standard notation of characterizing a text sequence
- How can we search for any of the following:
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks

- RE search requires a pattern and a corpus of texts to search through.

Regular Expression (RE)

RE	Example patterns matched
`woodchunks`	"interesting links to woodchanks and…."
`a`	"Mary Ann stopped by Mona's"

- RE is case-sensitive
- Letters inside square brackets []

Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit

- Ranges `[A-Z]`

Pattern	Matches	
`[A-Z]`	An upper case letter	`Drenched Blossoms`
`[a-z]`	A lower case letter	`my beans were impatient`
`[0-9]`	A single digit	`Chapter 1: Down the Rabbit Hole`

Negation

- Negations [^Ss]
  - Carat means negation only when it appears immediately after "["

Pattern	Matches	Example patterns matched
[^A-Z]	Not an upper case letter	Oyfn pripetchik
[^Ss]	Neither 'S' nor 's'	I have no exquisite reason"
[e^]	Either 'e' or '^'	Look here
a^b	The pattern 'a' carat 'b'	Look up a^b now

It solves the problem of woodchuck vs. Woodchuck

But not woodchuck vs woodchucks

Not woodchuck vs groundhog

Question mark ?

- ? Makes optimality of the pervious expression

Pattern	Matches
Woodchucks?	Woodchuck or Woodchucks
Colou?r	Color or Colour

It solves woodchuck vs woodchucks

Not woodchuck vs groundhog

pipe | for disjunction

Pattern	Matches
`groundhog|woodchuck`	Woodchucks is another name for groundhog
`yours|mine`	yours mine
`??a|b|c`	= [abc]
`[gG]roundhog|[Ww]oodchuck`	

It solves woodchuck vs groundhog

But not woodchucksssssssss

Kleene *, Kleene +

- Kleene * => zero or more occurrences of the immediately previous character or regular expression
- Kleene + => one or more of the previous character
- Period (.) matches any single character (except a carriage return)

Pattern	Matches	
`oo*h!`	0 or more of previous char	oh! ooh! oooh! ooooh!
`[ab]*`	Zero or more a's or b's	aaa ababab bbbb
`o+h!`	1 or more of previous char	oh! ooh! oooh! ooooh!
`baa+`		baa baaa baaaa baaaaa
`beg.n`		begin begun begun beg3n

Anchors ^ $

- Caret ^ matches the start of a line
  - Negations [^Ss] (careful!)
- \b matches a word boundary
- \B matches a non-boundary
- $ matches the end of a line

Pattern	Matches
^[A-Z]	
^[^A-Za-z]	
\bthe\b	
\.$	
.$	

Anchors ^ $

- Caret ^ matches the start of a line
  - Negations [^Ss] (careful!)
- \b matches a word boundary
- \B matches a non-boundary
- $ matches the end of a line

Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 "Hello"
\bthe\b	the, not "other"
\.$	The end.
.$	The end? The end!

https://www.regexpal.com/

Quiz

- Find all instances of the word “the” in a text.

```
the
```

You may miss capitalized examples

```
[tT]he
```

Incorrectly returns `other` or `theology`

```
[^a-zA-Z][tT]he[^a-zA-Z]
```

It won’t find the word the when it begins or ends a line

```
(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)
```

⇒ Before the we require either the beginning-of-line or a non-alphabetic character, and the same at the end of the line.

Error

- We want to fix two kinds of errors
  - False positives (Type I)
    - Matching strings that we should not have matched (there, then, other)
  - False negatives (Type II)
    - Not matching things that we should have matched (The)
- Reducing error may require a trade-off between
  - Accuracy or Precision: minimizing false positives
  - Coverage or Recall: minimizing false negative

Morphology

Challenges…

- How do we know that
  - Both *woorchunk* and woodchuncks have same original/root word?
    - May be easy: the plural just tacks as s on to end
  - But what about goose vs geese or fox vs foxes?
- **Two kinds of knowledge:**
  - **Orthographic rules**: can solve *woorchunk* vs. woodchuncks
  - **Morphological rules:** can distinguish goose vs geese

Orthographic/Spelling Rules

- General rules used when breaking a word into its stem and modifiers.
- Example:
  - Singular English words ending with –y, when pluralized, end with –ies.
  - Peccary vs. Peccaries

Morphological Rules

- Morphological rules are exceptions to the orthographic rules used when breaking a word into its stem and modifiers.
- Example:
  - Goose vs Geese is due to vowel change

Morphology: Definition

The study of words, how they are formed, and their relationship to other words in the same language.

Morphological Parsing

- **Parsing:** Take an input and produce some sort of linguistic structure
- Morphological parsing the process of determining the morphemes from which a given word is constructed.

Terminologies

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Church and Gale (1990): $|V| > O(N^{½})$

- **Surface form:** Raw text present in a corpus
  - Example: going
- **Token:** a word present in running text (may be duplicated)
- **Type:** Unique word present in the running text
- **Vocabulary:** Set of types

they lay back on the San Francisco grass and looked at the stars and their

- 15 tokens (or 14)
- 13 types (or 12) (or 11?)

- **Stem**
- **Affix : prefix/suffix/infix/circumfix**

Infix

Tagalog: hingi (borrow) => h**um**ingi

Circumfix

German: Sagen (to say) =>ge sagt (said)

Terminologies

- A word can have more than one affix
  - Example: rewrites *(re-, write,-s),* unbelievably *(un-, believe, -able, -ly)*
- English doesn’t tend to stack more than 4 or 5 affixes
- Languages that tend to string affixes together like Turkish does are called **agglutinative** languages
  - Turkish can have words with 9 or 10 affixes

Terminologies

- **Inflection:** A word stem with a grammatical morpheme, usually resulting in a word of the same class as the original stem.
  - **Example:** Plural (-s) or past (-ed)

- **Derivation:** the combination of a word stem with a grammatical morpheme, usually resulting in a word of a different class
  - **Example:** computerize (verb) vs. computerization (noun)

Concatenative morphology is Easy!

- **Non-concatenative morphology**
  - Philipian language (Tagalog)
  - Um + hingi (request) = humingi (ask for)
- **Templatic morphology/root-and-pattern morphology**
  - In Hebrew, a verb is constructed from two components: a root (CCC) and a template (ordering of C and V to specify more semantic info)
  - *lmd* (learn/study) can be combined with active voice *CaCaC* template to produce *lamad* (he studied)
  - *CiCeC* => limed (he taught)
  - *CuCaC => lumad* (he was taught)

The Porter Stemmer (Porter, 1980)

- A simple rule-based algorithm for stemming
- An example of a HEURISTIC method
- Based on rules like:
  - ATIONAL -> ATE (e.g., *relational* -> *relate*)
- The algorithm consists of seven sets of rules, applied in order

The Porter Stemmer: definitions

- Definitions:
  - CONSONANT: a letter other than A, E, I, O, U, and Y preceded by consonant (e.g. SYZYGY)
  - VOWEL: any other letter
- With this definition, all words are of the form:
  $(C)(VC)^m(V)$
  C=string of one or more consonants (con+)
  V=string of one or more vowels
  m>=0
- E.g.,
  - Tr ou bl e s
  - C V C V C

The Porter Stemmer: rule format

- The rules are of the form:

  (condition) S1 -> S2

  Where S1 and S2 are suffixes

- Conditions:

m	The measure of the stem
*S	The stem ends with S
v	The stem contains a vowel
*d	The stem ends with a double consonant
*o	The stem ends in CVC (second C not W, X, or Y)

The Porter Stemmer: Step 1

- SSES -> SS
  - *caresses -> caress*
- IES -> I
  - *ponies -> poni*
  - *ties -> ti*
- SS -> SS
  - *caress -> caress*
- S -> ϵ
  - *cats -> cat*

The Porter Stemmer: Step 2a (past tense, progressive)

- (m>0) EED -> EE
  - Condition verified: *agreed -> agree*
  - Condition not verified: *feed -> feed*
- (*V*) ED -> ϵ
  - Condition verified: *plastered -> plaster*
  - Condition not verified: *bled -> bled*
- (*V*) ING -> ϵ
  - Condition verified: *motoring -> motor*
  - Condition not verified: *sing -> sing*

m	The measure of the stem
*S	The stem ends with S
v	The stem contains a vowel
*d	The stem ends with a double consonant
*o	The stem ends in CVC (second C not W, X, or Y)

The Porter Stemmer: Step 2b (cleanup)

- (These rules are ran if second or third rule in 2a apply)
- AT-> ATE
  - *conflat(ed) -> conflate*
- BL -> BLE
  - *Troubl(ing) -> trouble*
- (*d & ! (*L or *S or *Z)) -> single letter
  - Condition verified*: hopp(ing) -> hop, tann(ed) -> tan*
  - Condition not verified*: fall(ing) -> fall*
- (m=1 & *o) -> E
  - Condition verified*: fil(ing) -> file*
  - Condition not verified*: fail -> fail*

Why?

m	The measure of the stem
*S	The stem ends with S
v	The stem contains a vowel
*d	The stem ends with a double consonant
*o	The stem ends in CVC (second C not W, X, or Y)

(*V*) ED -> ϵ

Condition verified*: plastered -> plaster*

Condition not verified*: bled -> bled*

(*V*) ING -> ϵ

Condition verified*: motoring -> motor*

Condition not verified*: sing -> sing*

The Porter Stemmer: Steps 3 and 4

- Step 3: Y Elimination (*V*) Y -> I
  - Condition verified: *happy -> happi*
  - Condition not verified: *sky -> sky*
- Step 4: Derivational Morphology,
  - (m>0) ATIONAL -> ATE
    - *Relational -> relate*
  - (m>0) IZATION -> IZE
    - *generalization-> generalize*
  - (m>0) BILITI -> BLE
    - *sensibiliti -> sensible*

m	The measure of the stem
*S	The stem ends with S
v	The stem contains a vowel
*d	The stem ends with a double consonant
*o	The stem ends in CVC (second C not W, X, or Y)

The Porter Stemmer: Steps 5 and 6

- Step 5: Derivational Morphology, II
  - (m>0) ICATE -> IC
    - *triplicate -> triplic*
  - (m>0) FUL -> ϵ
    - *hopeful -> hope*
  - (m>0) NESS -> ϵ
    - *goodness -> good*
- Step 6: Derivational Morphology, III
  - (m>0) ANCE -> ϵ
    - *allowance-> allow*
  - (m>0) ENT -> ϵ
    - *dependent-> depend*
  - (m>0) ANT -> ϵ
    - *irritant -> irrit*
  - (m>0) IVE -> ϵ
    - *effective -> effect*

m	The measure of the stem
*S	The stem ends with S
v	The stem contains a vowel
*d	The stem ends with a double consonant
*o	The stem ends in CVC (second C not W, X, or Y)

The Porter Stemmer: Step 7 (cleanup)

- Step 7a
  - (m>1) E -> ϵ
    - *probate -> probat*
  - (m=1 & !*o) NESS -> ϵ
    - *goodness -> good*
- Step 7b
  - (m>1 & *d & *L) -> single letter
    - Condition verified*: controll -> control*
    - Condition not verified*: roll -> roll*

m	The measure of the stem
*S	The stem ends with S
v	The stem contains a vowel
*d	The stem ends with a double consonant
*o	The stem ends in CVC (second C not W, X, or Y)

Examples

- *computers*
  - Step 1, Rule 4: -> *computer*
  - Step 6, Rule 4: -> *compute*
- *singing*
  - Step 2a, Rule 3: -> *sing*
- *controlling*
  - Step 2a, Rule 3: -> *controll*
  - Step 7b : -> *control*
- *generalizations*
  - Step 1, Rule 4: -> *generalization*
  - Step 4, Rule 11: -> *generalize*
  - Step 6, last rule: -> *general*

Problems

- *elephants -> eleph*
  - Step 1, Rule 4: -> *elephant*
  - Step 6, Rule 7: -> *eleph*
- *Etc……*