Natural Language Processing (NLP)

Renato R. Maaliw III, *DIT*
College of Engineering
Southern Luzon State University
Lucban, Quezon, Philippines

NLP

- An area of computer science & artificial intelligence concerned with the interactions between computers and **human (natural) languages**.

- Particular on how to program computers to process and analyze large amounts of **natural language data**.

- When performing analysis, lots of data is numerical (sales numbers, physical measurements, quantifiable categories)

- Computers are **very good** at handling direct **numerical** information

- But what do we do about **text data**?

- As humans we can tell there is a lot of information inside of text documents

- However, a computer **needs** specialized processing techniques in order to “understand” raw text data.

- Text data is **highly unstructured** and can be in multiple languages.

- NLP attempts to use a **variety of techniques** in order to create a structure out of text data

- We will discuss (first) some of the techniques using libraries such as **Spacy** and **NLTK**

Use Cases of NLP:

- Classifying emails as spam vs. legitimate
- Sentiment analysis of text movie reviews
- Analyzing trends from written customer feedback forms
- Understanding text commands, “Hey Google, play this song”

- NLP is constantly evolving and great strides are made every month.

- In this lessons, we will focus on the fundamental ideas that all state-of-the-art techniques are based off

- We will learn about the basics of using the Spacy library.

Spacy Basics

- Loading the language library
- Building a pipeline object
- Using tokens
- Parts-of-Speech tagging
- Understanding token attributes

- The **nlp()** function from Spacy automatically takes raw text and performs a series of operations to tag, parse, and describe the text data.

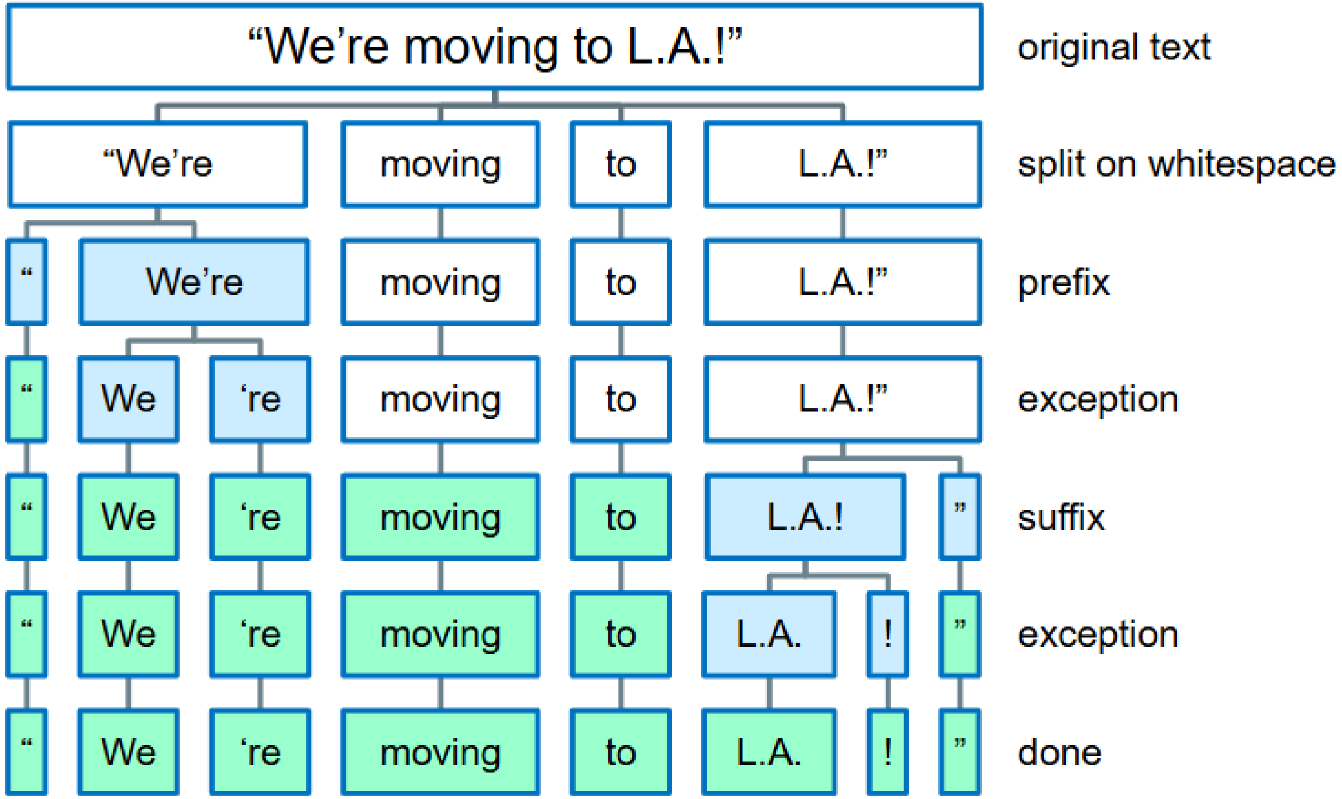
- We will create a pipeline objects and its series of operations

- Examples: Tokenization, POS, Stemming, Lemmatization, etc.

[Code Demo]

Tokenization

- Tokenization is the process of breaking up the original text into component pieces (tokens)



- Notice that tokens are pieces of the **original text**

- We don’t see any conversion to word stems or lemmas (base forms of words) and we haven’t seen anything about organizations/places/money etc.

- **Prefix**: character(s) at the beginning → *$ (“*
- **Suffix:** characters(s) at the end → *km) , . ! “*
- **Infix:** character(s) in between → *- -- / ...*
- **Exception**:

  special-case rule to split a string into several tokens or prevent a token from being split when punctuation rules are applied → *let’s U.S.*

- **Tokens** have a variety of useful attributes and methods

[Code Demo]

Tokenization Visualization

- Visualization of token relationships

[Code Demo]

Stemming

- Often when searching text for a certain keyword, it helps if the search returns variations of the word

- For example, searching for “boat” might also return “boats” and “boating”. Here “boat” would be the stem for [boat, boater, boating, boats]

Why Stemming is Necessary?

1. Text Normalization

- Words like *running*, *runs*, and *ran* share the same root (*run*). Stemming reduces these inflections and derivations to a common base form, making it easier to treat them as the same word.

- This simplifies text data by reducing redundant word variations.

Why Stemming is Necessary?

2. Improves Search and Matching

- In search engines or databases, stemming allows a query for *run* to return documents containing *runs*, *running*, or *ran*.
- It improves the recall of search results by accounting for morphological variations.

Why Stemming is Necessary?

3. Reduces Dimensionality

- By grouping different forms of a word into a single base form, stemming reduces the size of the vocabulary.

- This is especially useful in machine learning and NLP tasks, where large vocabularies increase computational complexity.

Why Stemming is Necessary?

4. Enhances Machine Learning Models

- Helps algorithms focus on the meaning of the word rather than its grammatical variations.

- Reduces noise and sparsity in feature space, improving the performance of models like text classifiers or topic models.

Why Stemming is Necessary?

5. Essential for Languages with Rich Morphology

- For languages with complex inflections (e.g., Finnish, Turkish), stemming simplifies words to their root forms, making analysis feasible.

Why Stemming is Necessary?

6. Improves Clustering and Similarity

- In clustering or similarity-based tasks, stemming ensures that words with similar meanings are grouped together.

Without Stemming:

Words: run, running, runner, runs
Vocabulary: ['run', 'running', 'runner', 'runs']

After Stemming:

Words: run, run, run, run
Vocabulary: ['run']

* This reduction ensures that all variations are treated as one, improving both efficiency and consistency.

Limitations:

Over-stemming: Some stemmers may cut too much, merging unrelated words (e.g., *universe* and *university* both reduced to *univers*).

Loss of Meaning: Stemming does not preserve the context or precise meaning of words, as it focuses on root forms.

Modern Alternative*: Lemmatization*, a more sophisticated approach, considers the context and grammar of a word to return its proper dictionary form.

- In fact, SpaCy doesn't include a stemmer, opting instead to rely on **lemmatization**.

- Because of this, we will jump over to using **NLTK** and learn about stemmers.

- **Porter Stemmer** and **Snowball Stemmer**

Porter Stemmer

- ***Porter Stemmer*** employs five phases of word reduction, each with its own set of mapping rules.

- In the first phase, simple suffix mapping rules are defined, such as:

S1		S2	word		stem
SSES	→	SS	caresses	→	caress
IES	→	I	ponies ties	→ →	poni ti
SS	→	SS	caress	→	caress
S	→		cats	→	cat

- From a given set of stemming rules only one rule is applied, based on the longest suffix S1. Thus, caresses reduces to caress but not cares

S1		S2	word		stem
SSES	→	SS	caresses	→	caress
IES	→	I	ponies ties	→ →	poni ti
SS	→	SS	caress	→	caress
S	→		cats	→	cat

- More sophisticated phases consider the length/complexity of the word before applying a rule.

S1		S2	word		stem
(m>0) ATIONAL	→	ATE	relational national	→ →	relate national
(m>0) EED	→	EE	agreed feed	→ →	agree feed

Snowball Stemmer

- ***Snowball Stemmer*** used a more accurate "English Stemmer" or "Porter2 Stemmer"

- Offers a slight improvement over the original Porter stemmer, both in logic and speed

[Code Demo]

Lemmatization

- In contrast to stemming, ***lemmatization*** looks beyond word reduction, and considers a language's full vocabulary to apply a morphological analysis to words.

- Generally better as it returns the dictionary form of a word (lemma) while considering context and part of speech (POS). Unlike stemming, it avoids over-simplifying words and ensures the output is meaningful and accurate.

Feature	**Stemming**	**Lemmatization**
Definition	Cuts the word to its root, often a crude form.	Reduces a word to its base form (lemma) using context.
Context-Sensitivity	Ignores context and parts of speech (POS).	Considers context and POS for accuracy.
Output	May produce non-existent or incorrect words.	Always produces valid dictionary words.
Examples	*Caring → car*	*Caring → care*

Example 1. Handling Verb Forms

Word	Stemming	Lemmatization (POS: Verb)
running	run	run
ran	ran	run
runs	run	run

* Why better? Lemmatization recognizes all verb forms and reduces them to the base form (*run*)

Example 2. Handling Nouns

Word	Stemming	Lemmatization (POS: Nouns)
geese	gees	goose
feet	feet	foot

* Why better? Lemmatization uses dictionary rules to handle irregular nouns.

Example 3. Avoiding Over-Stemming

Word	Stemming	Lemmatization (POS: Verb)
organization	organ	organization
organize	organ	organize

* Why better? Stemming cuts too aggressively, losing meaning, while lemmatization retains proper forms.

Example 4. Context Sensitivity

Word	Stemming	Lemmatization
He is building a house.	He is build a house	He is building a house.
They build houses	They build house.	They build houses

* Why better? Lemmatization retains proper inflection and pluralization based on sentence context.

Advantages of Lemmatization over Stemming

1. **Accuracy**: Produces correct dictionary words and considers POS tags.
2. **Context Awareness**: Handles irregular forms and context-sensitive words.
3. **Readability**: Outputs meaningful words suitable for downstream tasks like summarization or translation.

Cognate/Professional Electives

Tag	Description
`ADJ`	Adjective (e.g., *big, happy, round*).
`ADP`	Adposition (e.g., *in, at, of, under*).
`ADV`	Adverb (e.g., *quickly, silently*).
`AUX`	Auxiliary Verb (e.g., *is, has, do*).
`CCONJ`	Coordinating Conjunction (e.g., *and, but, or*).
`DET`	Determiner (e.g., *a, the, some*).
`INTJ`	Interjection (e.g., *wow, ouch*).
`NOUN`	Noun (e.g., *dog, city, happiness*).
`NUM`	Numeral (e.g., *one, two, 3.14*).
`PART`	Particle (e.g., *to* in *to run*).
`PRON`	Pronoun (e.g., *he, she, it, they*).
`PROPN`	Proper Noun (e.g., *John, Paris, Tesla*).
`PUNCT`	Punctuation (e.g., *, . ! ?*).
`SCONJ`	Subordinating Conjunction (e.g., *if, while, that*).
`SYM`	Symbol (e.g., *$, %, @*).
`VERB`	Verb (e.g., *run, think, be*).
`X`	Other (e.g., unknown words, foreign language text).

[Code Demo]

Stop Words

- Stop words are common words in a language (e.g. *the*, *is*, *in*, *and*) that are frequently filtered out in text processing because they are often not significant for tasks like information retrieval, text classification, or sentiment analysis.

- They are frequent but do not contribute much meaning to the overall text

- Spacy holds a built-in list of English stop words

Common English Stop Words

- is, are, the, a, an, of, in, on, and, it, to.

Example:

"The quick brown fox jumps over the lazy dog."

With Stop Words:

"quick brown fox jumps over the lazy dog."

[Code Demo]

Vocabulary and Matching

- We will identify and label specific phrases that match patterns that we can define ourselves

- We will take parts of speech into account for our pattern search

[Code Demo]

Thank you very much for listening.