NLP and Machine Learning

Feature extraction by NLP

Text Pre-processing

- The process of converting data to something a computer can understand is referred to as pre-processing
- One of the major forms of pre-processing is going to be filtering out useless data.

Text Pre-processing steps

- Tokenization
- Stop word and punctuation removal
- Stemming/ Lemmatization
- Case-folding
- POS-Tagging

Tokenizing

- Splitting sentences and words from the body of text
- Token
  - Each word is a token when a sentence is "tokenized" into words.
  - Each sentence can also be a token, if you tokenized the sentences out of a paragraph

Stop Words

- In natural language processing, useless words (data), are referred to as stop words.
- "too common" words
- filler words
- We would not want these words taking up space in our database, or taking up valuable processing time
- Excluded from the vocabulary entirely.
  -

Stop word list in nltk

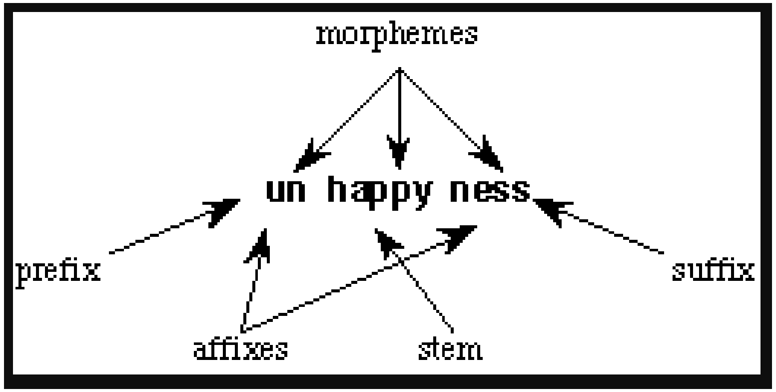
{'ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once', 'during', 'out', 'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its', 'yours', 'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's', 'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves', 'until', 'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me', 'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their', 'while', 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any', 'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves', 'then', 'that', 'because', 'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under', 'he', 'you', 'herself', 'has', 'just', 'where', 'too', 'only', 'myself', 'which', 'those', 'i', 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my', 'against', 'a', 'by', 'doing', 'it', 'how', 'further', 'was', 'here', 'than'}

Morphology

Morphology is the study of the structure and formation of words.

Morphemes:

- The small meaningful units that make up words
- **Stems**: The core meaning-bearing units
- **Affixes**: Bits and pieces that adhere to stems
  - Often with grammatical functions



Stemming

- Many variations of words carry the same meaning, other than when tense is involved
- I was taking a ride in the car.

  I was riding in the car.
- Having individual dictionary entries per version would be highly redundant and inefficient
- The reason why we stem it to shorten the lookup, and normalize sentences.
- One of the most popular stemming algorithms is the Porter stemmer, which has been around since 1979

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

Porter's Stemmer Algorithm

- Rule based stemmer.
- Porter stemmer is used for suffix stemming

Step 1a

sses	→	ss	caresses	→	caress
ies	→	i	ponies	→	poni
ss	→	ss	caress	→	caress
s	→	ø	cats	→	cat

Step 1b

(*v*)ing	→	ø	walking	→	walk
			sing	→	sing
(*v*)ed	→	ø	plastered	→	plaster

Step 2 (for long stems)

ational	→	ate	relational	→	relate
izer	→	ize	digitizer	→	digitize
ator	→	ate	operator	→	operate
…					

Step 3 (for longer stems)

al	→	ø	revival	→	reviv
able	→	ø	adjustable	→	adjust
ate	→	ø	activate	→	activ

http://people.scs.carleton.ca/~armyunis/projects/KAPI/porter.pdf

Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is → be*
  - *car, cars, car's, cars' → car*
- *the boy's cars are different colors → the boy car be different color*
- Lemmatization: have to find correct dictionary headword form

- **Stemming can often create non-existent words, whereas lemmas are actual words.**
- **Lemmatize takes a part of speech parameter, "pos" If not supplied, the default is "noun”.**
- **This means that an attempt will be made to find the closest noun**

Lemmatization

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("cats"))

print(lemmatizer.lemmatize("geese"))

print(lemmatizer.lemmatize("better", pos="a"))
```

Output:

cat
goose
good

Case folding (**Converting all letters to lower or upper case**)

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***

Part Of Speech Tagging

- **Part of Speech (pos) tagging is the problem of assigning each word in a sentence the part of speech that it assumes in that sentence.**
- **This means labeling words in a sentence as nouns, adjectives, verbs...etc. Even more impressive, it also labels by tense, and more.**

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

Problem of POS tagging is to resolve ambiguities, choosing the proper tag for the context.

Some Tagsets from Penn Treebank tags

DT– determiner ‘the’
IN -- preposition/subordinating conjunction ‘in’
JJ -- adjective ‘big’
NN -- noun, singular 'desk'
NNS-- noun plural 'desks‘
NNP --- proper noun, singular 'Harrison'
NNPS --- proper noun, plural 'Americans‘
RB --- adverb ‘very’, ‘silently’
VB --- verb, base form ‘take’
VBD --- verb, past tense ‘took’
VBG --- verb, gerund/present participle ‘taking
PRP--- personal pronoun ‘I’, ‘he’, ‘she’
VBP--Verb non-3rd person singular present form ‘refuse’
VBZ--- Verb 3rd person singular present form ‘eats’
To--- to ‘to’

Penn Treebank tags

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	*#*
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*'s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Open and Close classes

An **open class** is one that commonly accepts the addition of new words.
A **closed class** is one to which new items are very rarely added.

- Open vs. Closed classes
  - Closed:
    - determiners: *a, an, the*
    - pronouns: *she, he, I*
    - prepositions: *on, under, over, near, by, …*
    - Why “closed”?
  - Open:
    - Nouns, Verbs, Adjectives, Adverbs.

Function words have little ambiguous meaning and express grammatical relationships among other words within a sentence, or specify the attitude or mood of the speaker (Closed class)
Content words are words that name objects of reality and their qualities. (Open class)

Part Of Speech Tagging is difficult

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
  - I know *that* he is honest = IN
  - Yes, *that* play was nice = DT
  - You can't go *that* far = RB
- 40% of the word tokens are ambiguous

Exercise on Stemming (Porter Stemmer)

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Answer

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Exercise on POS Tagging

1. It is a nice night.
2. The grand jury commented on a number of other topics.
3. The planet Jupiter and its moons are in effect a minisolar system , and Jupiter itself is often called a star that never caught fire.”.

Answers

1. It/**PRP** is/**VBZ** a/**DT** nice/**JJ** night/**NN** ./.

2. The/**DT** grand/**JJ** jury/**NN** commented/**VBD** on/**IN** a/**DT** number/**NN** of/**IN** other/**JJ** topics/**NNS** ./.

3. “The/**DT** planet/**NN** Jupiter/**NNP** and/**CC** its/**PPS** moons/**NNS** are/**VBP** in/**IN** effect/**NN** a/**DT** minisolar/**JJ** system/**NN** ,/, and/**CC** Jupiter/**NNP** itself/**PRP** is/**VBZ** often/**RB** called/**VBN** a/**DT** star/**NN** that/**IN** never/**RB** caught/**VBN** fire/**NN** ./.”

Name Entity Recognision

Germany's representative to the European Union's veterinary committee Werner Zwingman said on Wednesday consumers should …

Name Entity Recognition

- A very important sub-task: find and classify names in text, for example:

  - The decision by the independent MP Andrew Wilkie to withdraw his support for the minority Labor government sounded dramatic but it should not further threaten its stability. When, after the 2010 election, Wilkie, Rob Oakeshott, Tony Windsor and the Greens agreed to support Labor, they gave just two guarantees: confidence and supply.

Person
Date
Location
Organi-
zation

Understanding needs & availabilities (Use-Case of NER Taggiing)

Category	Description
Resource	**What** are needed / available
Quantity	**How much** of each resource is needed / available
Location	**Where** is the resource needed / available
Source	**Who** needs / is offering the resource
Contact	**Whom** to contact (phone or email)

How many words?

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss's cat in the hat is different from other cats!
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

How many words

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

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.

How Many Tokens?
15 or 14
How many types?
13 (twice ‘and’ and twice ‘the’) or 12 (‘they’ and ‘their’ are same lemma) or 11 (if we consider Sun Francisco as single word)

Token vs Vocabulary

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

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

Bag Of Words (BOW)

- The bag-of-words model is one of the feature extraction algorithms for text
- The bag of words model ignores grammar and order of words.
- BOW method convert text to a numerical representation called a feature vector.
- A feature vector can be as simple as a list of numbers
- Computers are very well at handling numbers

Bag Of Words (BOW) Example

Step 1: Collect Data

‘All my cats in a row’,

‘When my cat sits down, she looks like a Furby toy!’,

Step 2: Design the Vocabulary

A list in then created based on the two strings above:

{‘all’: 0, ‘cat’: 1, ‘cats’: 2, ‘down’: 3, ‘furby’: 4, ‘in’: 5, ‘like’: 6, ‘looks’: 7, ‘my’: 8, ‘row’: 9, ‘she’: 10, ‘sits’: 11, ‘toy’: 12, ‘when’: 13 }

Step 3: Create Document Vectors

The list contains 14 unique words: the **vocabulary**

Then we can express the texts as numeric vectors:

[[1 0 1 0 0 1 0 0 1 1 0 0 0 0]

[0 1 0 1 1 0 1 1 1 0 1 1 1 1]]

We’ll get a vector, the bag of words representation.

Bag Of Words (BOW)

A bag-of-words is a representation of text that

Describes the occurrence of words within a document.

It involves two things:

•A vocabulary of known words.

•A measure of the presence of known words.

It is called a “*bag*” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

Shortcomings of Bag Of Words

The bag-of-words model is used with great success on prediction problems like documentation classification.

Nevertheless, it suffers from some shortcomings, such as:

Vocabulary: The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations.

Sparsity: Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.

Shortcomings of Bag Of Words

.

Meaning: Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modelled could tell the difference between the same words differently arranged (“this is interesting” vs “is this interesting”), synonyms (“old bike” vs “used bike”), and much more.

Managing Vocabulary

As the vocabulary size increases, so does the vector representation of documents.

There are simple text cleaning techniques that can be used as a first step, such as:

- Ignoring case
- Ignoring punctuation
- Ignoring frequent words that don’t contain much information, called stop words, like “a,” “of,” etc.
- Fixing misspelled words.
- Reducing words to their stem (e.g. “play” from “playing”) using stemming algorithms.

n-gram model

A more sophisticated approach is to create a vocabulary of grouped words.

This both changes the scope of the vocabulary and allows the bag-of-words to capture a little bit more meaning from the document.

•**In this approach, each word or token is called a “gram”.**

•**Creating a vocabulary of two-word pairs is, in turn, called a bigram model.**

•**An N-gram is an N-token sequence of words**

•**2-gram (more commonly called a bigram) is a two-word sequence of words like “please turn”, “turn your”, or “your homework”,**

•**3-gram (more commonly called a trigram) is a three-word sequence of words like “please turn your”, or “turn your homework”.**

n-gram model

For example, the bigrams in the first line of text in the previous section:

“It was the best of times” are as follows:

“it was”

“was the”

“the best”

“best of”

“ of times”

A vocabulary then tracks triplets of words is called a trigram model and the general approach is called the n-gram model, where n refers to the number of grouped words.

Often a simple bigram approach is better than a 1-gram bag-of-words model for tasks like documentation classification.

Scoring Words

In the worked of BOW example, we have already seen one very simple approach to scoring:

- A binary scoring of the presence or absence of words.

Some additional simple scoring methods include:

- **Counts**. Count the number of times each word appears in a document.
- **Frequencies**. Calculate the frequency that each word appears in a document out of all the words in the document.

TF-IDF

A problem with scoring word frequency is that highly frequent words start to dominate in the document (e.g. larger score), but may not contain as much “informational content” to the model as rarer but perhaps domain specific words.

- One approach is to rescale the frequency of words by how often they appear in all documents, so that the scores for frequent words like “the” that are also frequent across all documents are penalized.
- This approach to scoring is called Term Frequency – Inverse Document Frequency, or TF-IDF for short

TF-IDF

This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

Term Frequency: is a scoring of the frequency of the word in the current document. It measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

TF-IDF

Inverse Document Frequency:. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

The scores have the effect of highlighting words that are distinct (contain useful information) in a given document.

Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low.

$$\text{tf-idf}(t,d) = \text{tf}(t,d) \times \text{idf}(t)$$

Example

Consider a document containing 100 words wherein the word *cat* appears 3 times. The term frequency (i.e., tf) for *cat* is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word *cat* appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4.

Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12.

Exercise On BOW and n-gram

CORPUS

‘I love UEM Kolkata’
‘I like Mathematics’

References

1. https://www.youtube.com/watch?v=hwDhO1GLb_4
2. https://pythonprogramming.net
3. https://pythonprogramminglanguage.com
4. https://machinelearningmastery.com/gentle-introduction-bag-words-model/