ML for Text Analysis: Natural Language Processing

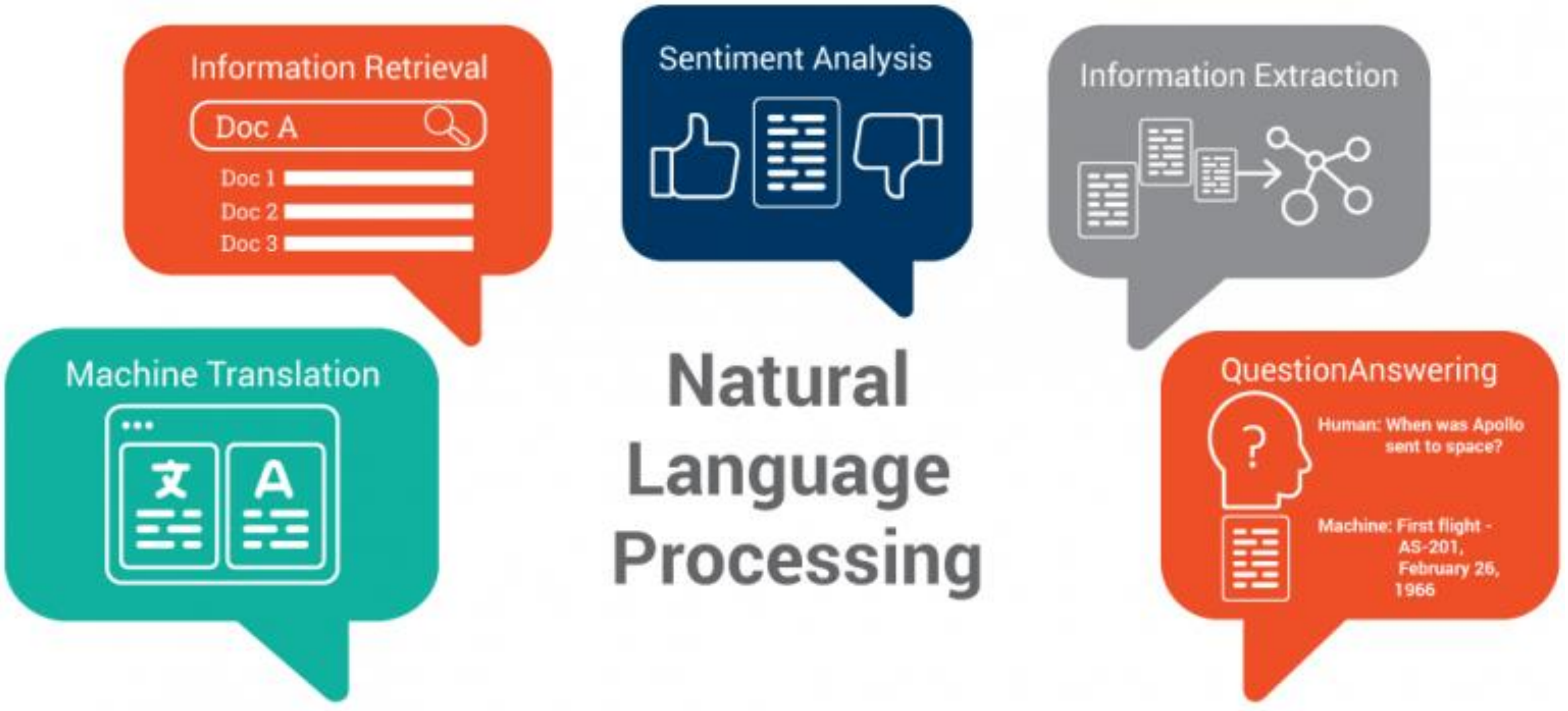
Urban Information Lab

Prof. Dr. Junfeng Jiao

TEXAS

The University of Texas at Austin

What is Natural Language Processing?

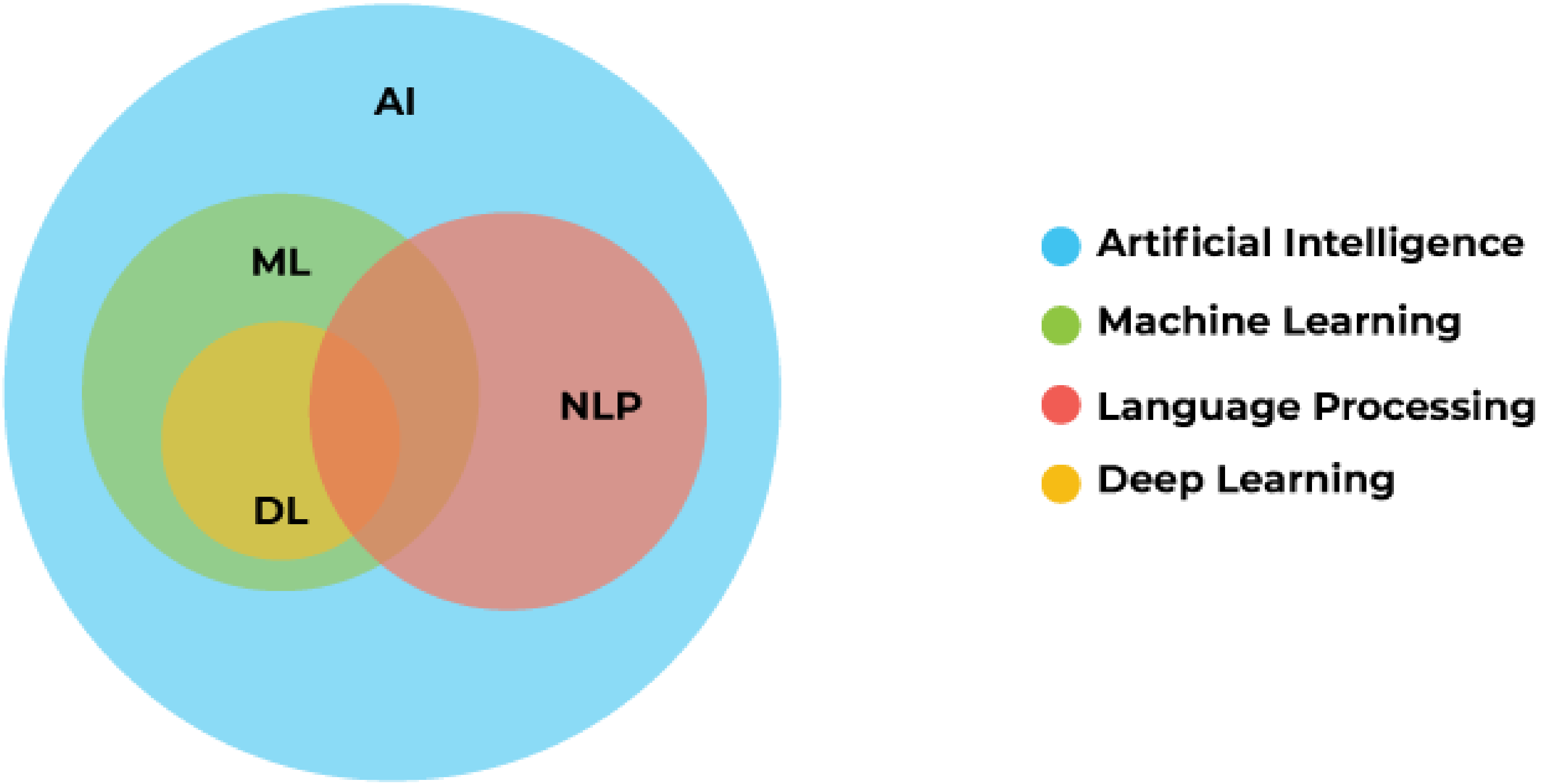


Natural language processing (NLP) is the process by which computers understand and process natural human language.

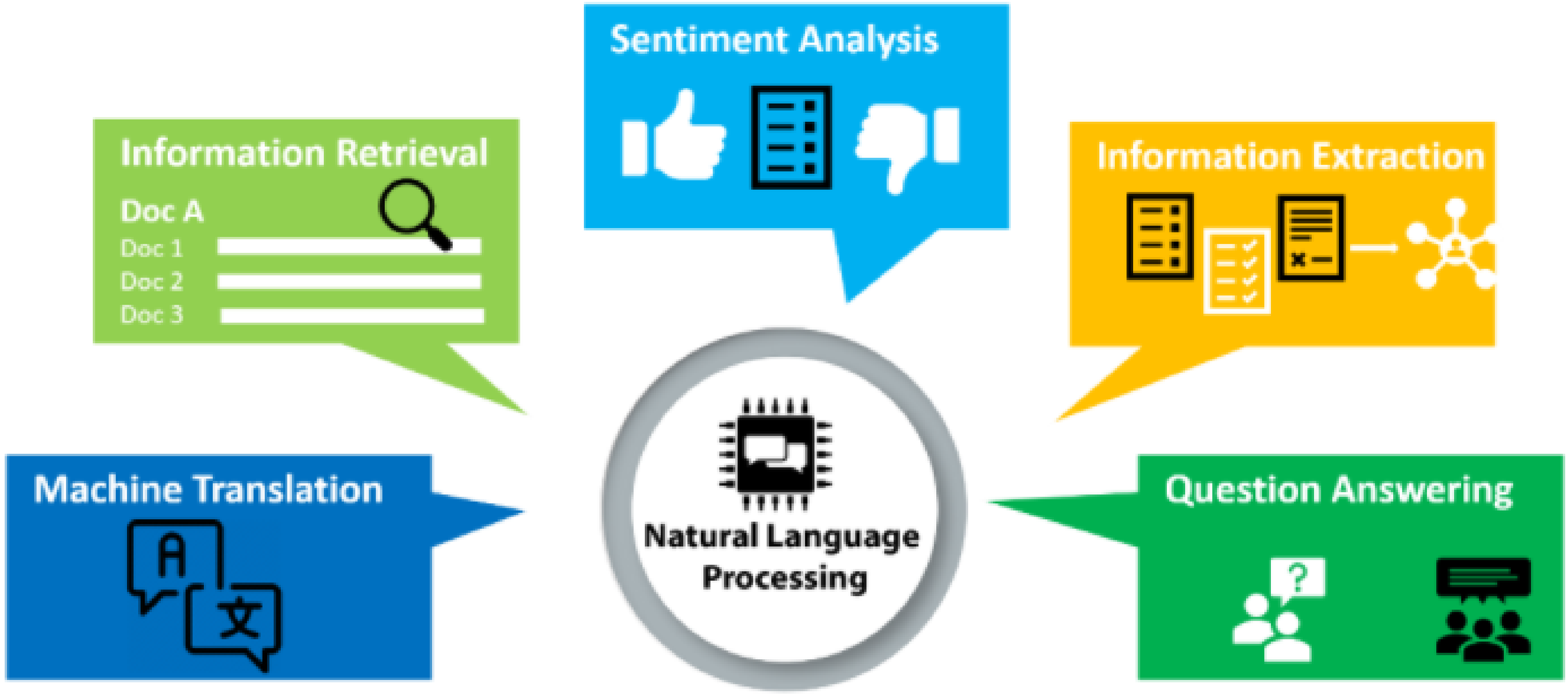
If you use Google Search, Alex, or Siri, you've already seen it at work. The advantage of NLP is that it allows users to make queries without first having to translate them into "computer-speak or computer-words"

1. Introduction

Natural Language Processing is a form of AI that gives machines the ability to not just read, but to understand and interpret human language. With NLP, machines can make sense of written or spoken text and perform tasks including speech recognition, sentiment analysis, and automatic text summarization.

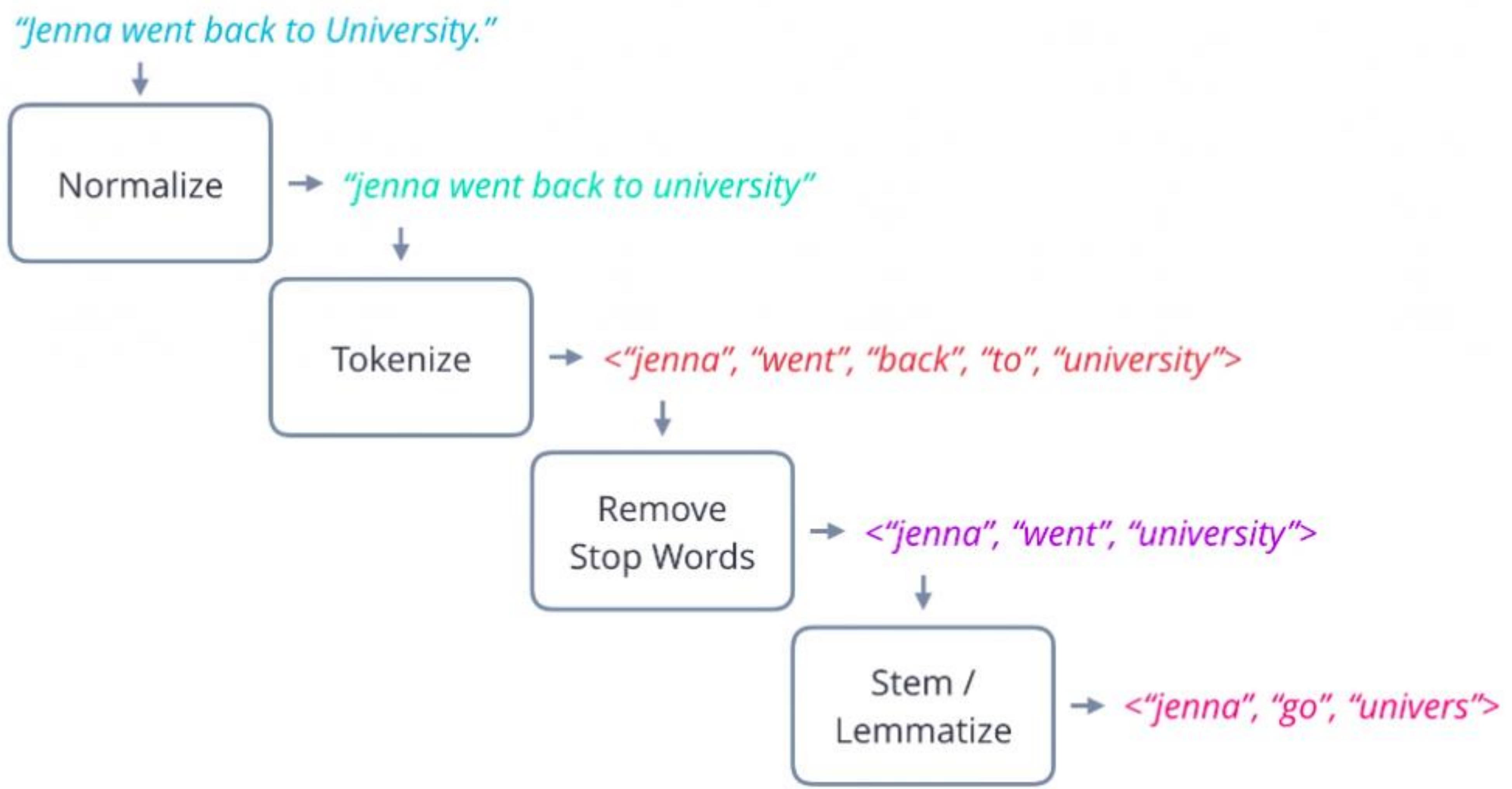


1. Introduction:
How Natural Language Processing can be applied

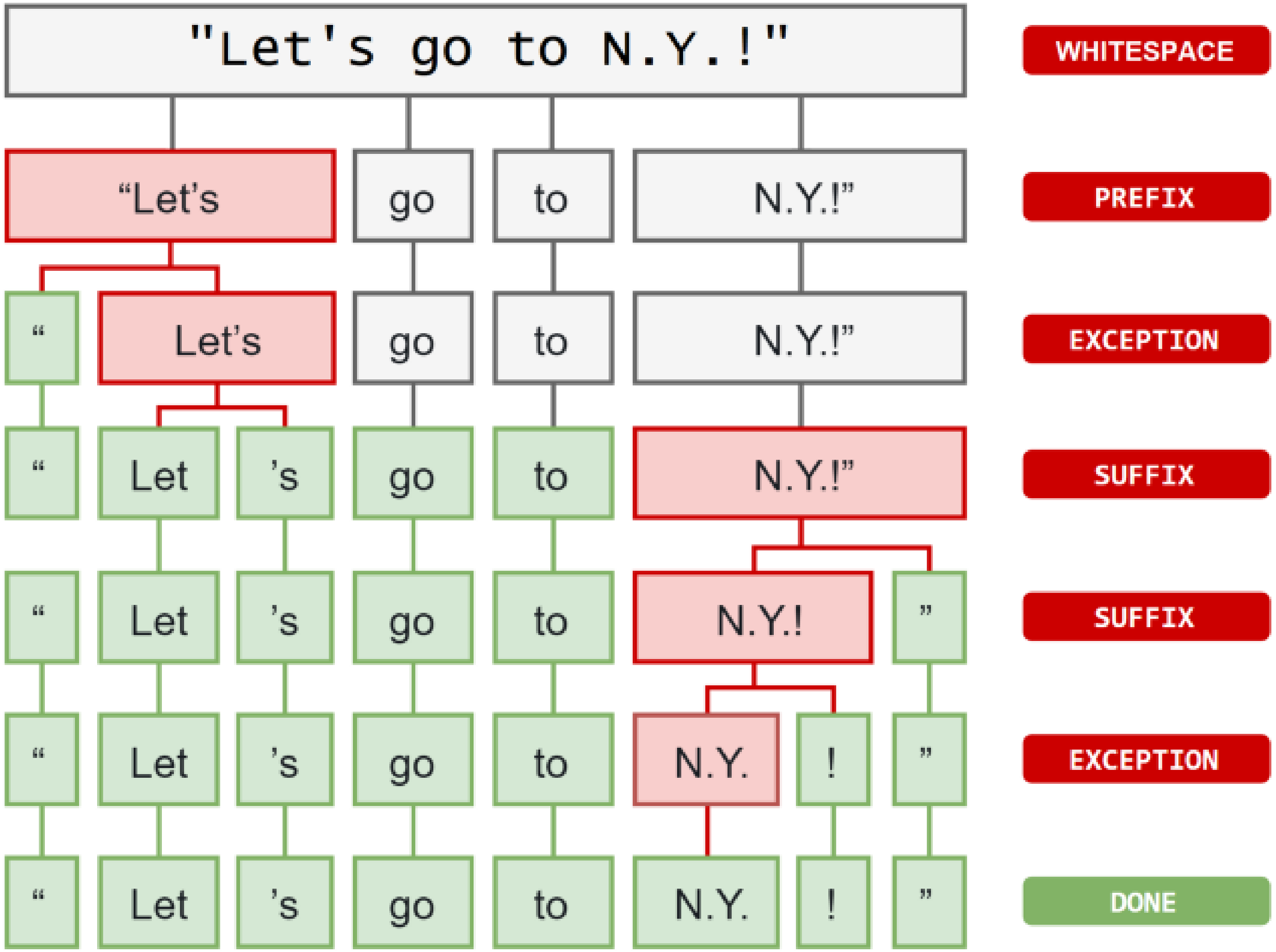


Chatbot,
Translation,
Speech recognition,
Question-Answer System,
Automatic text
Summarization

2. Text Processing in NLP

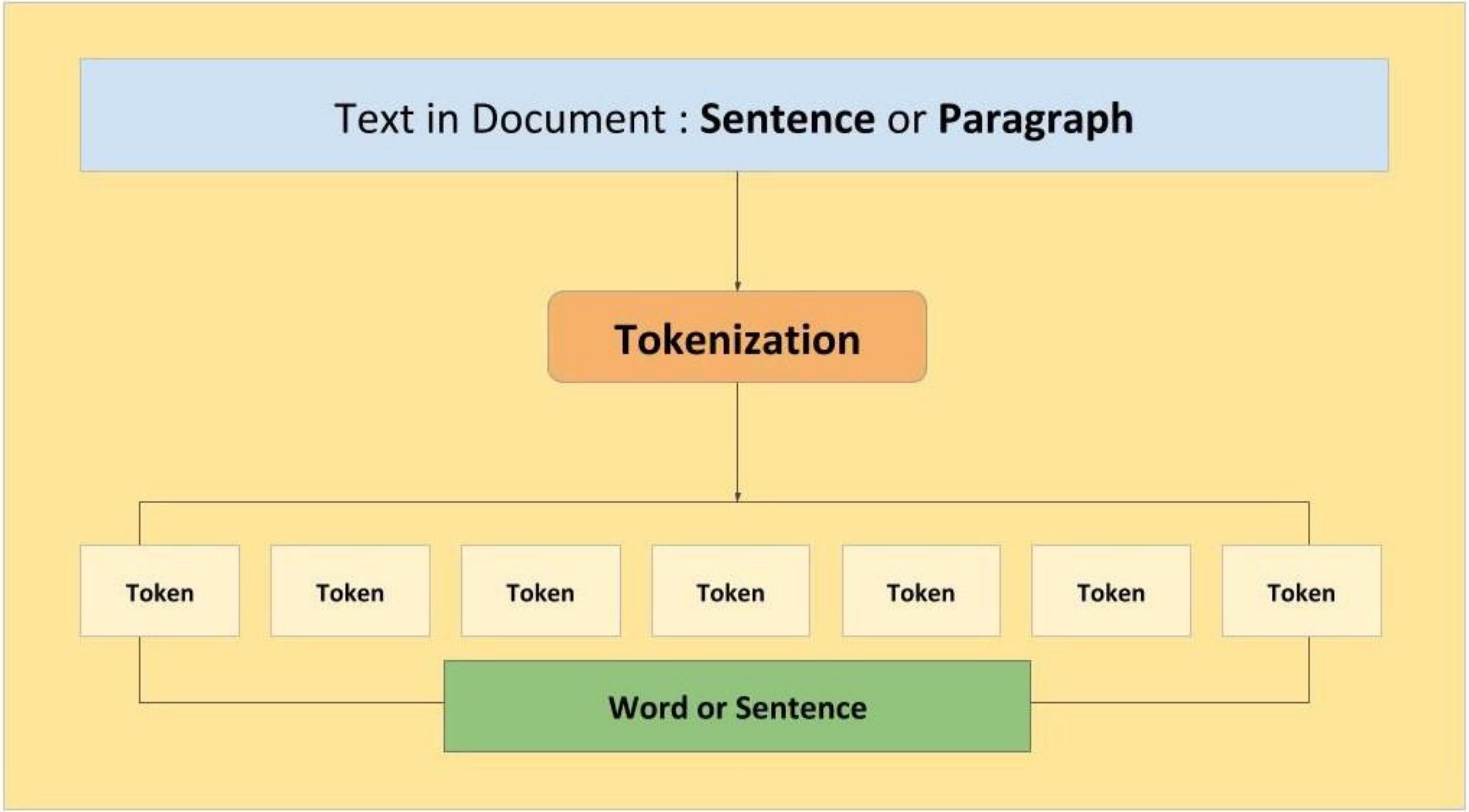


2. Text Processing in NLP: Tokenization



A *token* is an instance of a sequence of characters in some particular documents that are grouped together as a useful semantic unit for processing.

2. Text Processing in NLP: Tokenization

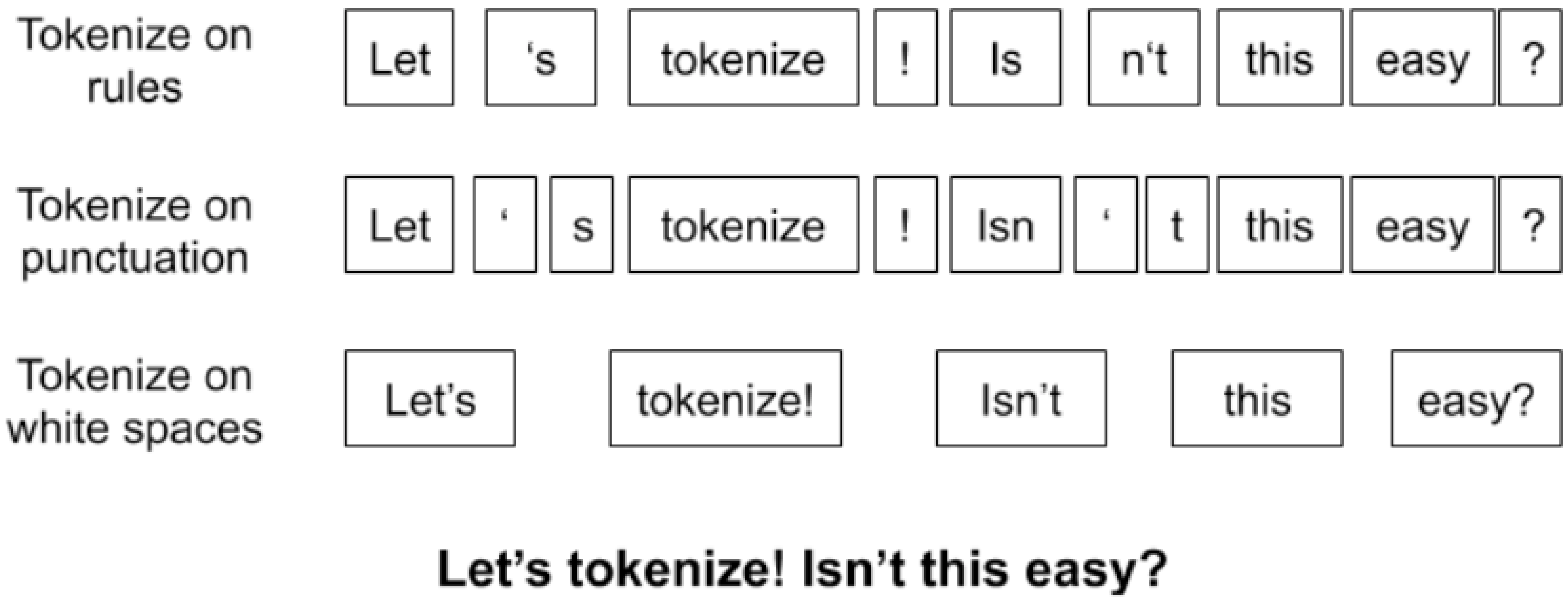


2. Text Processing in NLP: Tokenization

Challenges In Tokenization:
Some of the basic challenges lying in tokenization is to decide what is the best way to split/chop. One has to be smart enough to answer some the below questions.

•Will it be wise to just split on all non-alphanumeric characters, like **period**, **space bar,** etc.
•One has to decide how to treat apostrophes.
•What about splitting two-letter word like ‘West Bengal’
•What about the compound words in different languages like Sanskrit & German?

2. Text Processing in NLP: Tokenization

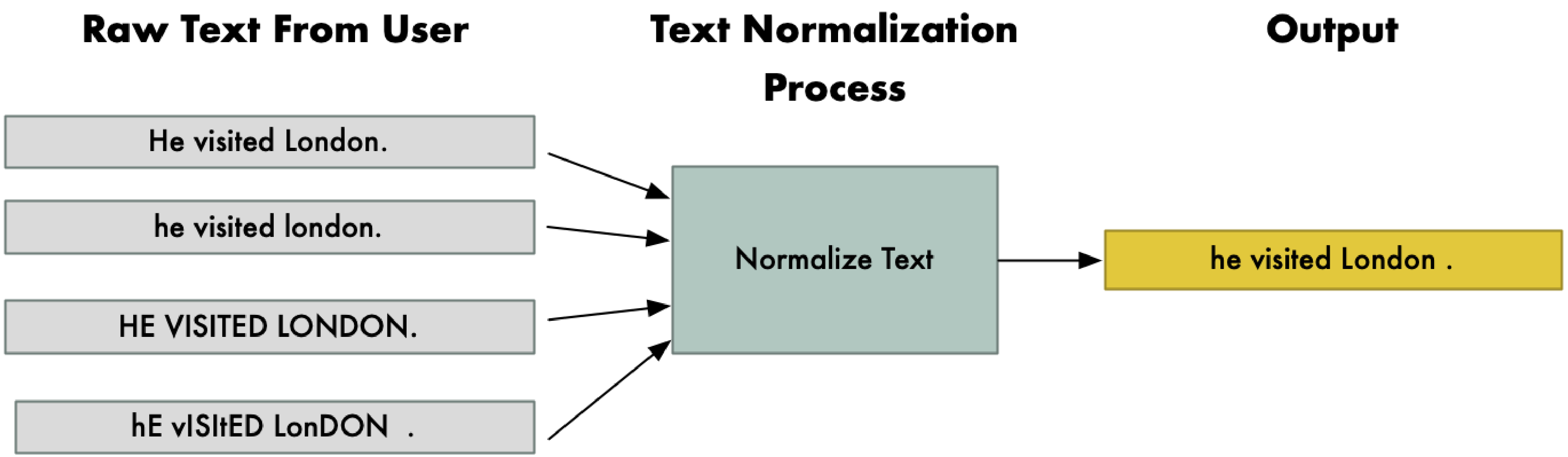


By word? Sentence? Spaces?

2. Text Processing in NLP: Normalization

Normalization is a process that converts a list of words to a more uniform sequence. This is useful in preparing text for later processing. By transforming the words to a standard format, other operations are able to work with the data and will not have to deal with issues that might compromise the process

Token normalization is the process of standardizing tokens so that matches occur despite superficial differences in the character sequences of the tokens



2. Text Processing in NLP: Stemming

Stemming is basically removing the suffix from a word and reduce it to its root word.

For example: “**Flying**” is a word and its suffix is “**ing**”,
if we remove “**ing**” from “**Flying**”
then we will get base word or root word which is “**Fly**”.

We uses these suffix to create a new word from original stem word

2. Text Processing in NLP: Stemming

OverStemming

Over-stemming is when two words with different stems are stemmed to the same root. This is also known as a false positive.

- universal
- university
- Universe

All the above 3 words are stemmed to univers which is wrong.
Though these three words are related, their modern meanings are very different, so treating them as synonyms in NLP is also incorrect.

2. Text Processing in NLP: Stemming

Porter Stemmer

•This is one of the most common and gentle stemmer, Its fast but not very precise.--- Natural Language Toolkit with python.

Porter Stemmer

In [53]:
```
import nltk
from nltk.stem.porter import *

porterStemmer = PorterStemmer()

sentence = "Provision Maximum multiply owed caring on go gone going was this"
wordList = nltk.word_tokenize(sentence)

stemWords = [porterStemmer.stem(word) for word in wordList]

print(' '.join(stemWords))
```

```
provis maximum multipli owe care on go gone go wa thi
```

2. Text Processing in NLP: Stemming

Snowball Stemmer

- The actual name of this stemmer is [English Stemmer]() or Porter2 Stemmer
- There were some improvements done on Porter Stemmer which made it more precise over large data-sets

Below is the implementation. You can use Jupyter Notebook to run the below code.

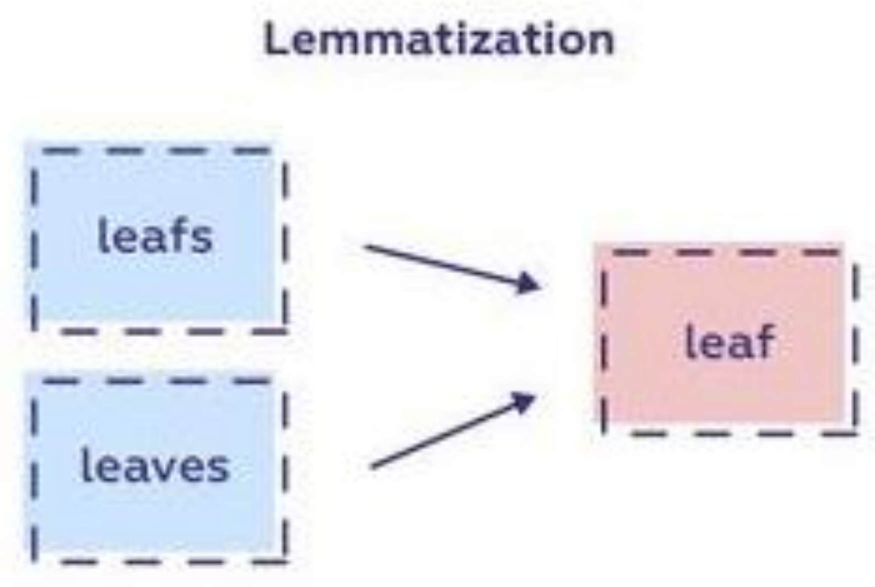
SnowBall Stemmer

In [50]:

```
import nltk
from nltk.stem.snowball import SnowballStemmer

snowBallStemmer = SnowballStemmer("english")

sentence = "Provision Maximum multiply owed caring on go gone going was this"
wordList = nltk.word_tokenize(sentence)

stemWords = [snowBallStemmer.stem(word) for word in wordList]

print(' '.join(stemWords))
```

```
provis maximum multipli owe care on go gone go was this
```

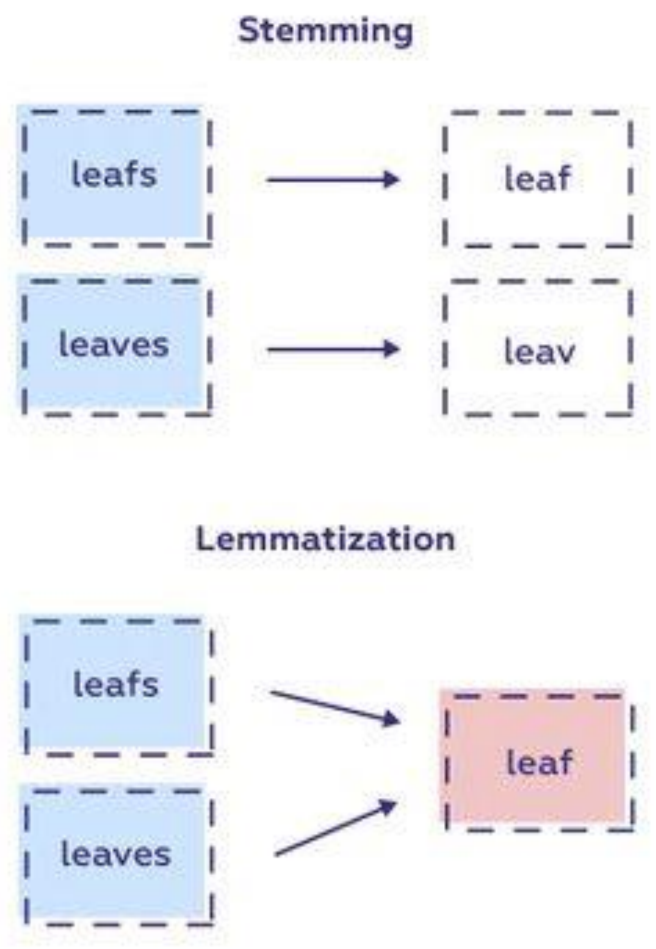
2. Text Processing in NLP: Lemmatization

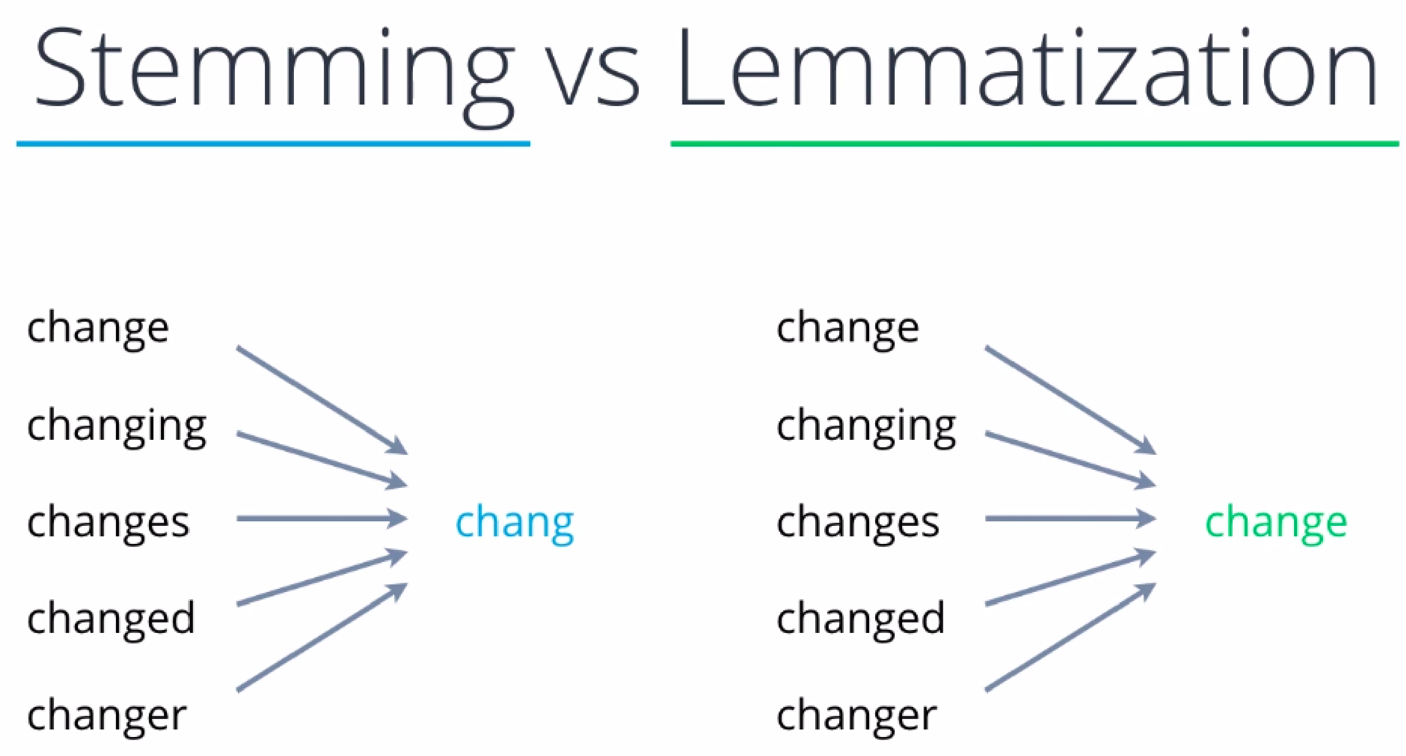


Lemmatization is another technique which is used to reduce words to a **normalized form**. In lemmatization, the transformation uses a **dictionary** to map different variants of a word back to its root format.

So, with this approach, we are able to reduce non-trivial inflections such as “is”, “was”, “were” back to the root “be”.

2. Text Processing in NLP: Lemmatization





3. Bag of Words (BOW)—general

- The **bag-of-words model** is a simplifying representation used in natural language processing. In this model, a text is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. (Wikipedia, 2022).

3. Bag of Words (BOW)—general

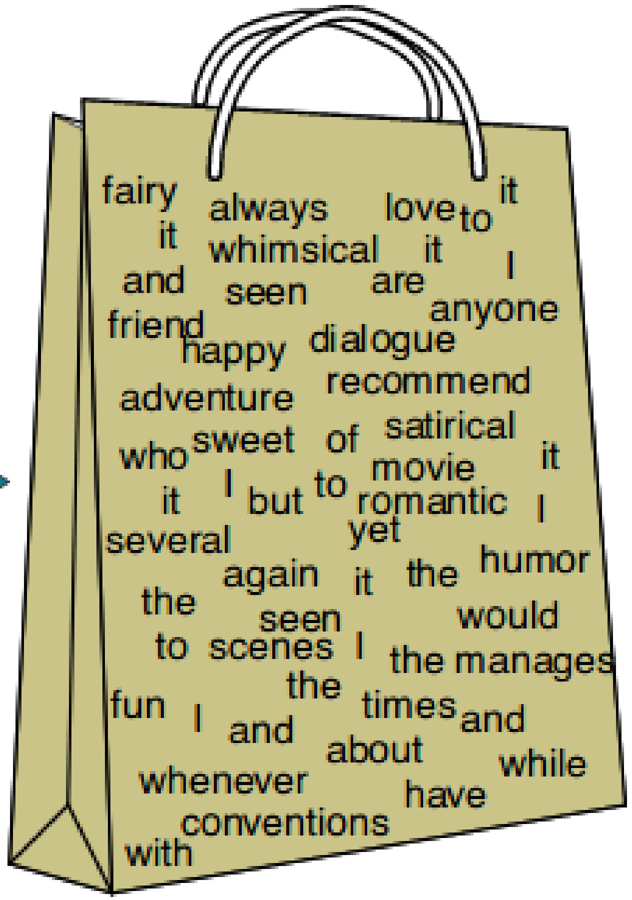
In bag of words, we take all unique words from the corpus, note the frequency of occurrence and sort them in descending order. All the words (vector mapping) we have will be used to represent each sentence. Let us take it in steps and examples to have a clear picture.

The bag-of-words model is simple to understand and implement. It is a way of extracting features from the text for use in machine learning algorithms.

(D'Souza, 2018)

3. Bag of Words—general

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

3. Bag of Words--frequency after tokenization

In this approach, we use the tokenized words for each observation and find out the frequency of each token.
Let’s take an example to understand this concept in depth.

“It was the best of times”
“It was the worst of times”
“It was the age of wisdom”
“It was the age of foolishness”

We treat each sentence as a separate document and we make a list of all words from all the four documents excluding the punctuation. We get,
‘It’, ‘was’, ‘the’, ‘best’, ‘of’, ‘times’, ‘worst’, ‘age’, ‘wisdom’, ‘foolishness’

3. Bag of Words-- frequency after tokenization

We take the first document — *"It was the best of times"* and we check the frequency of words from the 10 unique words.

"it" = 1

"was" = 1

"the" = 1

"best" = 1

"of" = 1

"times" = 1

"worst" = 0

"age" = 0

"wisdom" = 0

"foolishness" = 0

3. Bag of Words--vectorization

The process of converting NLP text into numbers is called **vectorization** in ML. Different ways to convert text into vectors are:

•Counting the number of times each word appears in a document.

Rest of the documents will be:
“It was the best of times” = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
“It was the worst of times” = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
“It was the age of wisdom” = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
“It was the age of foolishness” = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

•Calculating the frequency that each word appears in a document out of all the words in the document.

3. Bag of Words—Vectorization

Rome Paris word V

```
Rome   = [1, 0, 0, 0, 0, 0, ..., 0]
Paris  = [0, 1, 0, 0, 0, 0, ..., 0]
Italy  = [0, 0, 1, 0, 0, 0, ..., 0]
France = [0, 0, 0, 1, 0, 0, ..., 0]
```

3. Bag of Words—Problem

- The Bag-of-words model is an orderless document representation — only the counts of words matter.
- For instance, in the example "John likes to watch movies. Mary likes movies too", the bag-of-words representation will not reveal that the verb "likes" always follows a person's name in this text.
- As an alternative, the *n*-gram model can store this spatial information.
- Conceptually, we can view bag-of-word model as a special case of the n-gram model, with n=1.

(Wikipedia, 2022).

“John likes to watch movies. Mary likes movies too”

```
[
    "John likes",
    "likes to",
    "to watch",
    "watch movies",
    "Mary likes",
    "likes movies",
    "movies too",
]
```

3. Bag of Words—bigram

In this approach, each word or token is called a “gram”. Creating a vocabulary of two-word pairs is called a bigram model.
For example, the bigrams in the first document : “It was the best of times” are as follows:

“it was”
“was the”
“the best”
“best of”
“of times”

N=1, 2 and 3, n-gram Model

This is Big Data AI Book

Uni-Gram	This	Is	Big	Data	AI	Book
Bi-Gram	This is	Is Big	Big Data	Data AI	AI Book	
Tri-Gram	This is Big	Is Big Data	Big Data AI	Data AI Book		

3. Bag of Words—why N-gram

- For NLP, n-grams are used for a variety of things. Some examples include auto completion of sentences, auto spell check, and to a certain extent, check for grammar in a given sentence.
- How does N-gram do the above tasks?
- By calculating the word appearance frequencies.
- We'll show your examples of how to calculate the word appearances probabilities in n-grams.

4. N-gram Examples

- $P(w_n \mid w_1 w_2 \ldots w_{n-1})$ is called a parameter of the language model
- To estimating the values of the parameters of an N-gram model from the training data:

Unigram

$$P(w_i) = \frac{C(w_i)}{N}$$

$C(w_i)$ = count of occurrence of w_i

N = total number of words in the training data

Bigram

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1} w_i)}{C(w_{i-1})}$$

Trigram

$$P(w_i | w_{i-2} w_{i-1}) = \frac{C(w_{i-2} w_{i-1} w_i)}{C(w_{i-2} w_{i-1})}$$

4. N-gram Examples

The following formula will calculate the probability of word "w1" occurring after the word "w2"

```
count(w2 w1) / count(w2)
```

This will calculate the frequency of the words occurs in the required sequence, divided by the frequency of the word before the expected word occurs in the corpus.

4. N-gram Examples

If we have a training set like the following and use bigram to calculate words frequency and provide word suggestions

- I really like your iphone case.
- I am really happy for you.
- I really like your dress today.

4. N-gram Examples

- The probability of “like” after “really” will be:

count(really like) / count(really)
= 2 / 3
= 0.66

- The probability of “happy” after “really” will be:

count(really happy) / count(really)
= 1 / 3
= 0.33

4. N-gram Examples

Next time, when you type “I really”, our model will suggest “like”. It will get it correct 2 out 3 times.

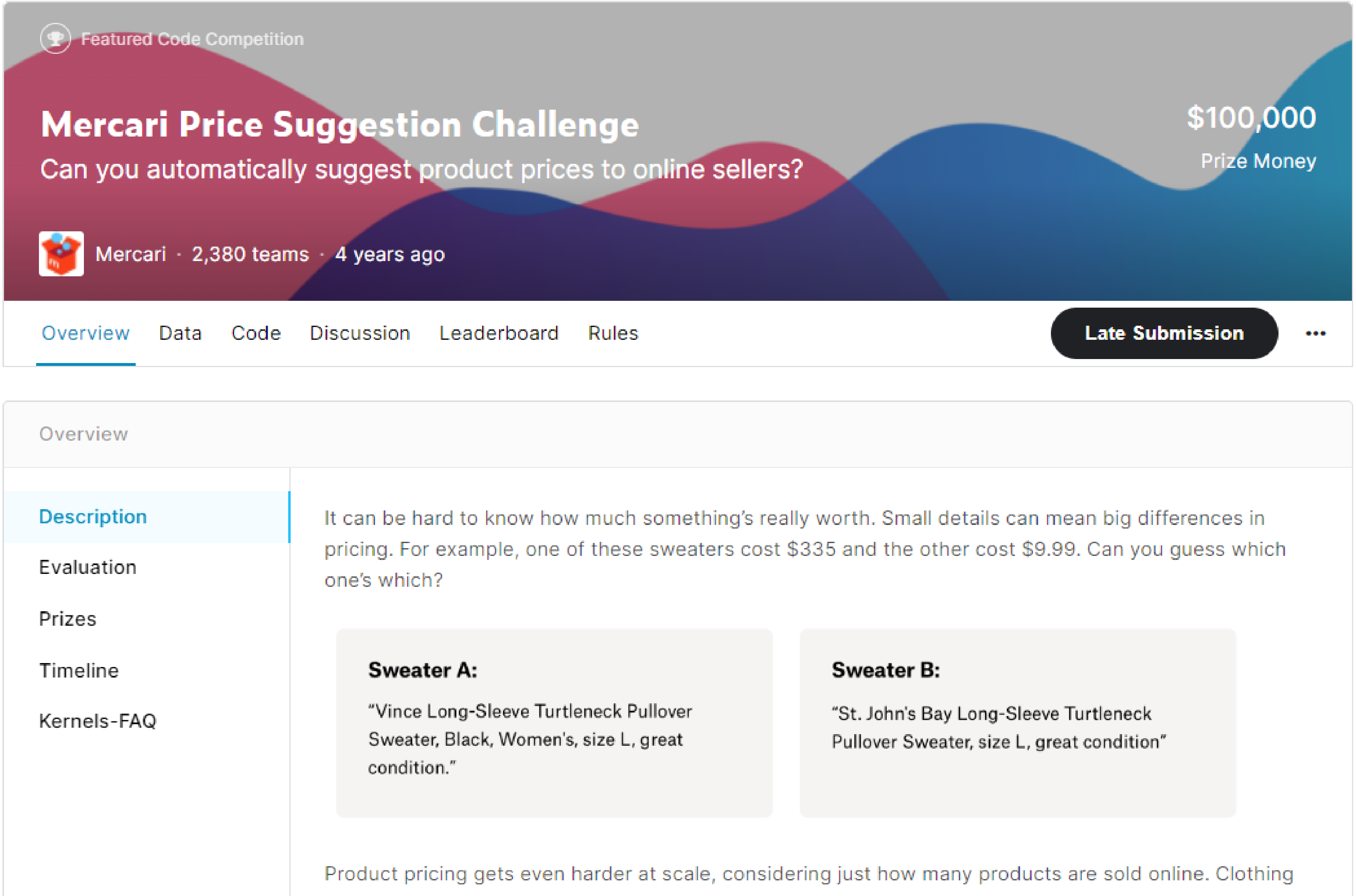
As you can see, the more training data (larger corpus) you have, the better the model will perform.

We can also use 3-gram or 4-gram model to improve the prediction results.

Lab Session : Price Suggestion with Text Analysis (Source Kaggle)

https://www.kaggle.com/c/mercari-price-suggestion-challenge

5. Data



https://www.kaggle.com/c/mercari-price-suggestion-challenge

Mercari Price Suggestion Challenge

- Mercari, Japan’s biggest community-powered shopping app, knows this problem deeply. They’d like to offer pricing suggestions to sellers, but this is tough because their sellers are enabled to put just about anything, or any bundle of things, on Mercari's marketplace.
- In this competition, Mercari’s challenging you to build an algorithm that automatically suggests the right product prices. You’ll be provided user-inputted text descriptions of their products, including details like product category name, brand name, and item condition.

The evaluation metric for this competition is Root Mean Squared Logarithmic Error.

The RMSLE is calculated as

$$\epsilon = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

\\(\epsilon\\) is the RMSLE value (score)
\\(n\\) is the total number of observations in the (public/private) data set,
\\(p_i\\) is your prediction of price, and
\\(a_i\\) is the actual sale price for \\(i\\).
\\(\log(x)\\) is the natural logarithm of \\(x\\)

5. Data

Victoria Secret
VS PINK
Michael Kor
Secret Pink
FREE SHIPPING
American Eagle
FREESHIP
Lularoe OS
Lularoe TC
Brand new
Rae Dunn
Kate Spade
Ralph Lauren
long sleeve
iPhone 6s
Lularoe legging
Funko Pop
North Face
Urban Decay
HOLD
sports bra
tank top
Vera Bradley
American Girl
Nike Air
skinny jean
Brandy Melville
Kendra Scott
Lilly Pulitzer
Old Navy
TC Legging
Sterling Silver
Body Work
Rose Gold
iPhone Plu
crop top
Calvin Klein
Bath Body
Reserved
Black White
Purple
Vintage
Bundle
Cute
Red
Large
Blue
Grey
Lot
Tory Burch
Baby girl
Dri Fit
size small
Secret Bra
OS Legging

Brand new
free shipping
great condition
never worn
description yet
Price firm
good condition
new tag
Victoria Secret
Size small
never used
new never
excellent condition
smoke free
Size medium
Size large
free home
Bundle save
Check listing
perfect condition
Super cute
without tag
business day
ask question
new box
long sleeve
worn time
PLEASE READ
Firm price
will ship
black white
Gently used

End of Doc.

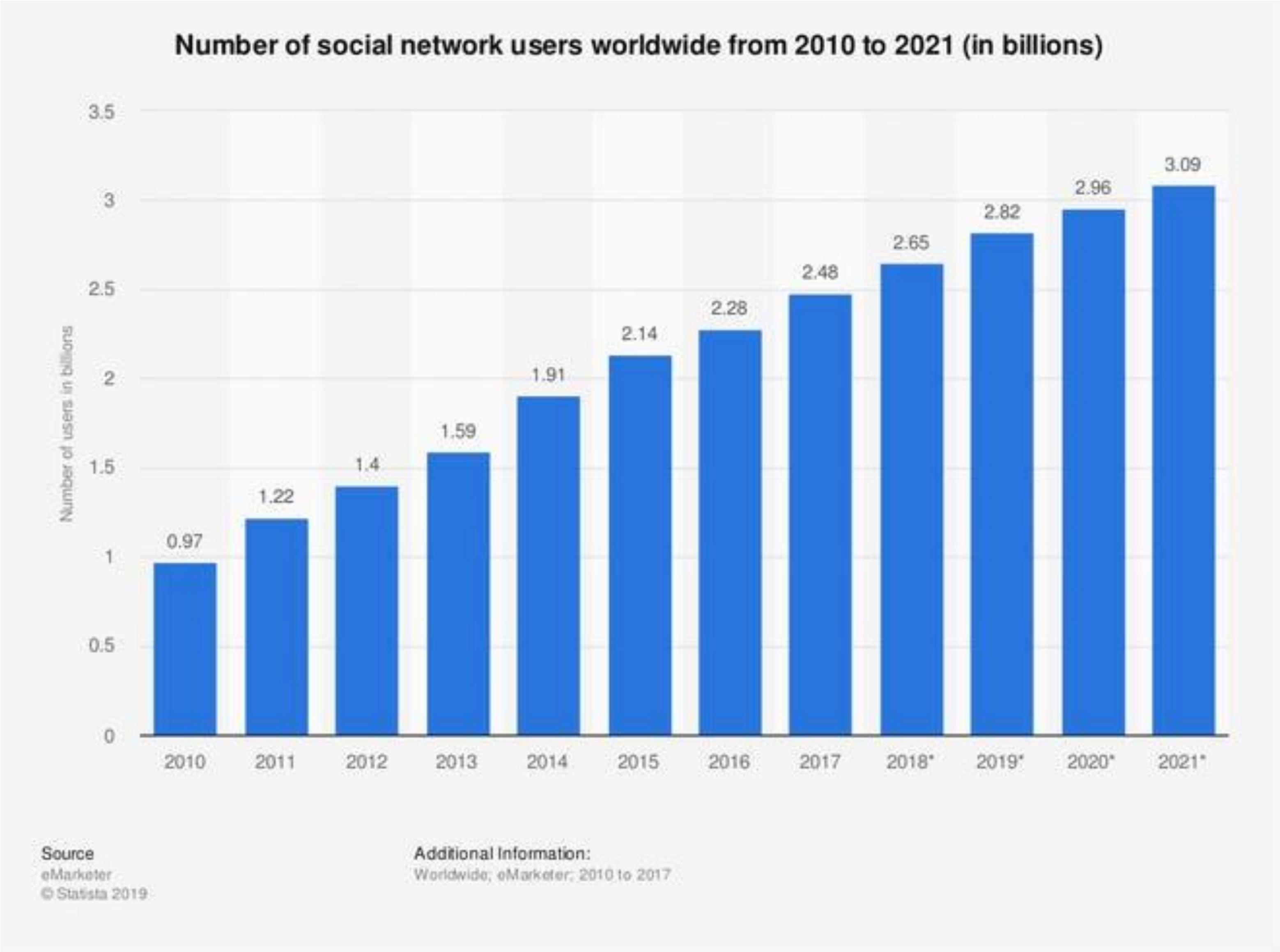
ML for Text Analysis:
Social Media Application

Urban Information Lab
Prof. Dr. Junfeng Jiao

TEXAS
The University of Texas at Austin

1. Social Media

1. Social Media



2. Encoder Decoder Model

Use Cases of the Sequence to Sequence Model

A sequence to sequence model lies behind numerous systems which you face on a daily basis. For instance, seq2seq model powers applications like Google Translate, voice-enabled devices and online chatbots.

2. Encoder Decoder Model

- *Speech recognition*

2. Encoder Decoder Model

- *Video captioning*

S2VT: A herd of zebras are walking in a field.

2. Encoder Decoder Model

Definition of the Sequence to Sequence Model

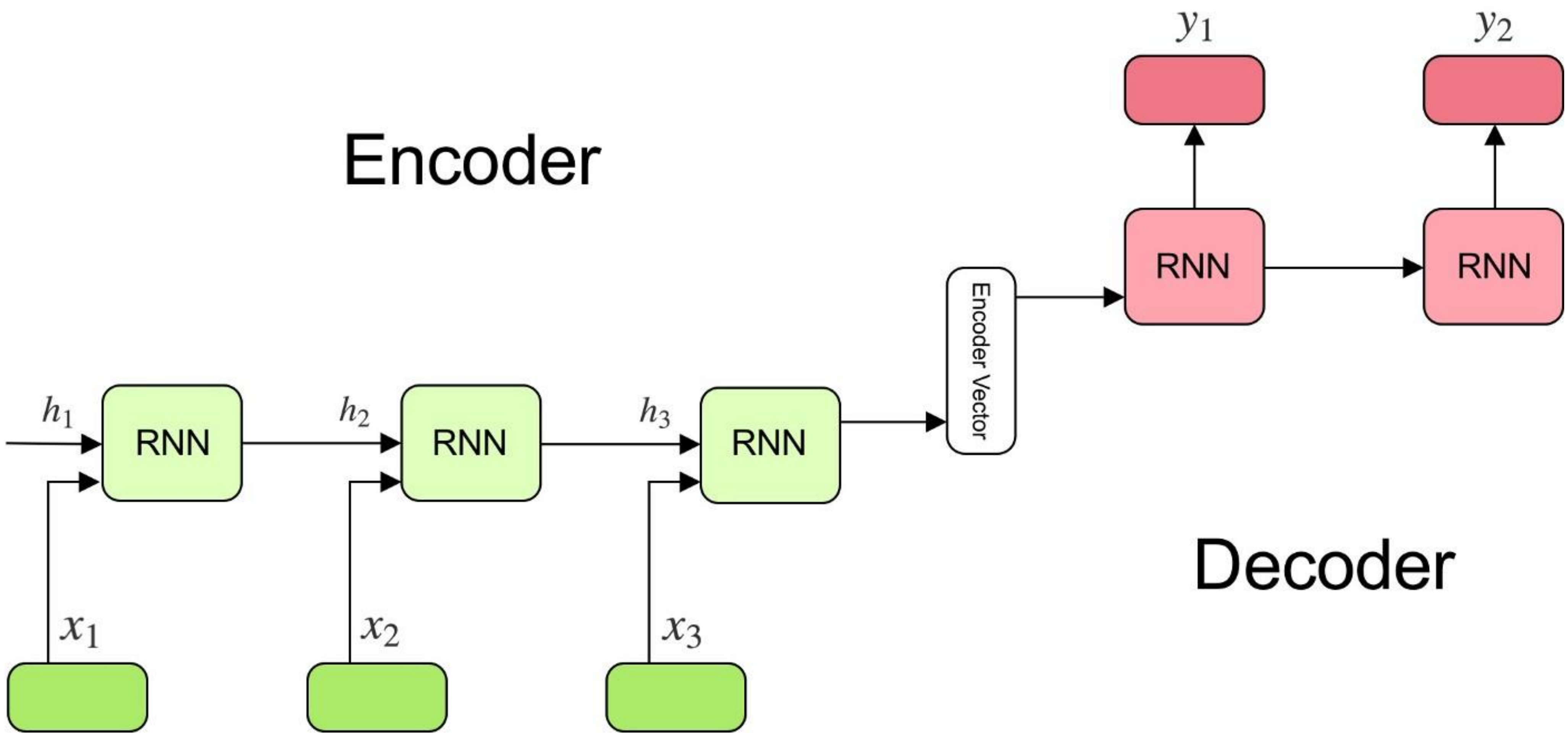
Introduced for the first time in 2014 by Google, a sequence to sequence model aims to map a fixed-length input with a fixed-length output where the length of the input and output may differ.

For example, translating “What are you doing today?” from English to Chinese has input of 5 words and output of 7 symbols (今天你在做什麼？).

Clearly, we can’t use a regular LSTM network to map each word from the English sentence to the Chinese sentence.

This is why the sequence to sequence model is used to address problems like that one.

3. How Does it Work?



The model consists of 3 parts: encoder, intermediate (encoder) vector and decoder.

3. How Does it Work?--Encoder

- A stack of several recurrent units (LSTM or GRU) where each accepts a single element of the input sequence, collects information for that element and propagates it forward.
- In question-answering problem, the input sequence is a collection of all words from the question. Each word is represented as *x_i* where *i* is the order of that word.
- The hidden states *h_i* are computed using the formula:

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

This simple formula represents the result of an ordinary recurrent neural network. As you can see, we just apply the appropriate weights to the previous hidden state *h_(t-1)* and the input vector *x_t.*

3. How Does it Work?- Encoder Vector

- This is the final hidden state produced from the encoder part of the model. It is calculated using the formula above.
- This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions.
- It acts as the initial hidden state of the decoder part of the model.

3. How Does it Work?--Decoder

- A stack of several recurrent units where each predicts an output *y_t* at a time step *t*.
- Each recurrent unit accepts a hidden state from the previous unit and produces and output as well as its own hidden state.
- In the question-answering problem, the output sequence is a collection of all words from the answer. Each word is represented as *y_i* where *i* is the order of that word.
- Any hidden state *h_i* is computed using the formula:

$$h_t = f(W^{(hh)} h_{t-1})$$

As you can see, we are just using the previous hidden state to compute the next one.

3. How Does it Work?

•The output *y_t* at time step *t* is computed using the formula:

$$y_t = softmax(W^S h_t)$$

We calculate the outputs using the hidden state at the current time step together with the respective weight W(S).
Softmax is used to create a probability vector which will help us determine the final output (e.g. word in the question-answering problem).

The power of this model lies in the fact that it can map sequences of different lengths to each other. As you can see the inputs and outputs are not correlated and their lengths can differ. This opens a whole new range of problems which can now be solved using such architecture.

REVISITING IMAGE OF CITY IN CYBERSPACE: ANALYSIS OF SPATIAL TWITTER MESSAGES DURING A SPECIAL EVENT

URBAN INFO LAB The University of Texas at Austin

TEXAS The University of Texas at Austin

Junfeng Jiao, Michael Holmes & Greg P. Griffin (2018) Revisiting Image of the City in Cyberspace: Analysis of Spatial Twitter Messages During a Special Event, Journal of Urban Technology, 25:3, 65-82, DOI: 10.1080/10630732.2017.1348881

What is the urban built environments in Cyberspace?

How people imagine their city, and how that structure appears in internet communications.

#City

2012 Super Bowl Tweets

- Scraped 600,000 tweets over the two week Super Bowl activity period in Indianapolis, Indiana
- Identified 78 locations in the city where tweets occurred
- Analyzed all the tweets related to the Super Bowl and Indianapolis and coded to Lynch's 5 city elements

Elements of a City’s Imageability

- <u>Node</u> - A center of activity with a purpose for travel
- <u>Landmark</u> - Prominent visual feature
- <u>Path</u> - Major and minor routes of circulation
- <u>Edge</u> - Dividing lines between districts
- <u>District</u> – A significant portion of the city with a common identifying character

…From Kevin Lynch’s The Image of the City

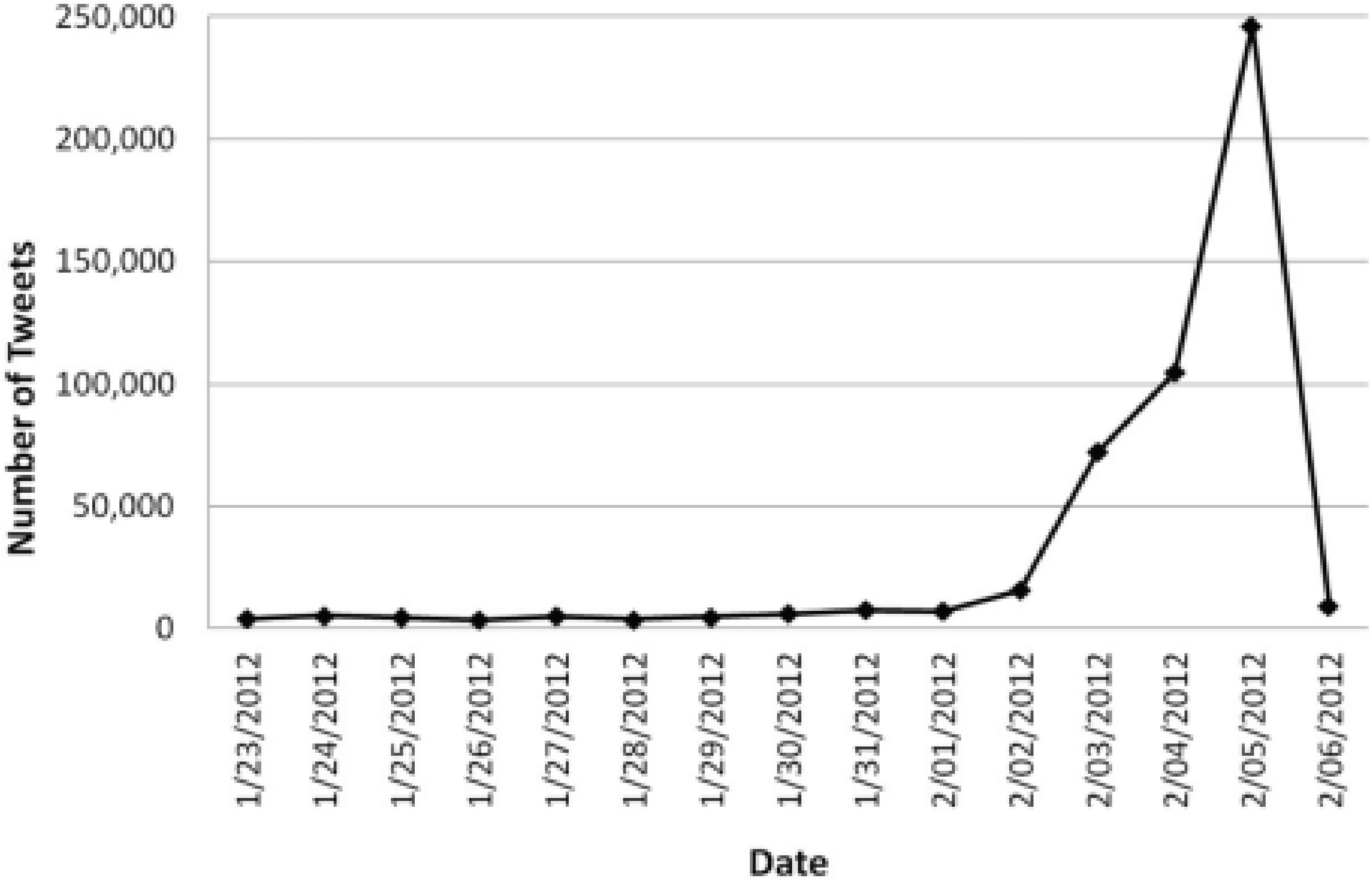
Table 1. Examples of deleted and retained tweets (user names omitted for clarity)

Examples of deleted tweets	Rationale
#Madonna look-A-like contest after #halftime Winner gets 520 in CASH&prizes! #SuperBowl	Super Bowl reference without spatial content.
@TwoWheeledBeard @JBaty83 @HoraceRawlins Patriots Vengeance Tour 2011–12. Next stop #SBXLVI in Indy.	No reference to anything within the city.
@GilletteStadium: Good morning back! Awake, Alive, Still excited!! #SBXLVI here we come!	Stadium reference is a team home stadium, not Lucas Oil Stadium in Indianapolis.
RT @itsyourboySham: Patriots are going to, without question, win the #SuperBowl	Simple retweet; also lacks spatial content.
Just walked downtown and they already getting ready for the super bowl #Indy	Includes references to a portion of the city and to the Super Bowl.
Is there a wait at kilroys? #social46 #superbowl2012	Names a local restaurant and has two event-related hashtags.
Check out #Indy downtown! RT @TheFieldhouse: Plenty of events in and around @TheFieldhouse in the coming weeks …	Includes a neighborhood reference and a venue reference (also an example of a retweet with added content).
XLVI Roman Numerals are up at Monument Circle. #superbowl #indianapolis	Contains a Super Bowl reference and a local landmark reference.

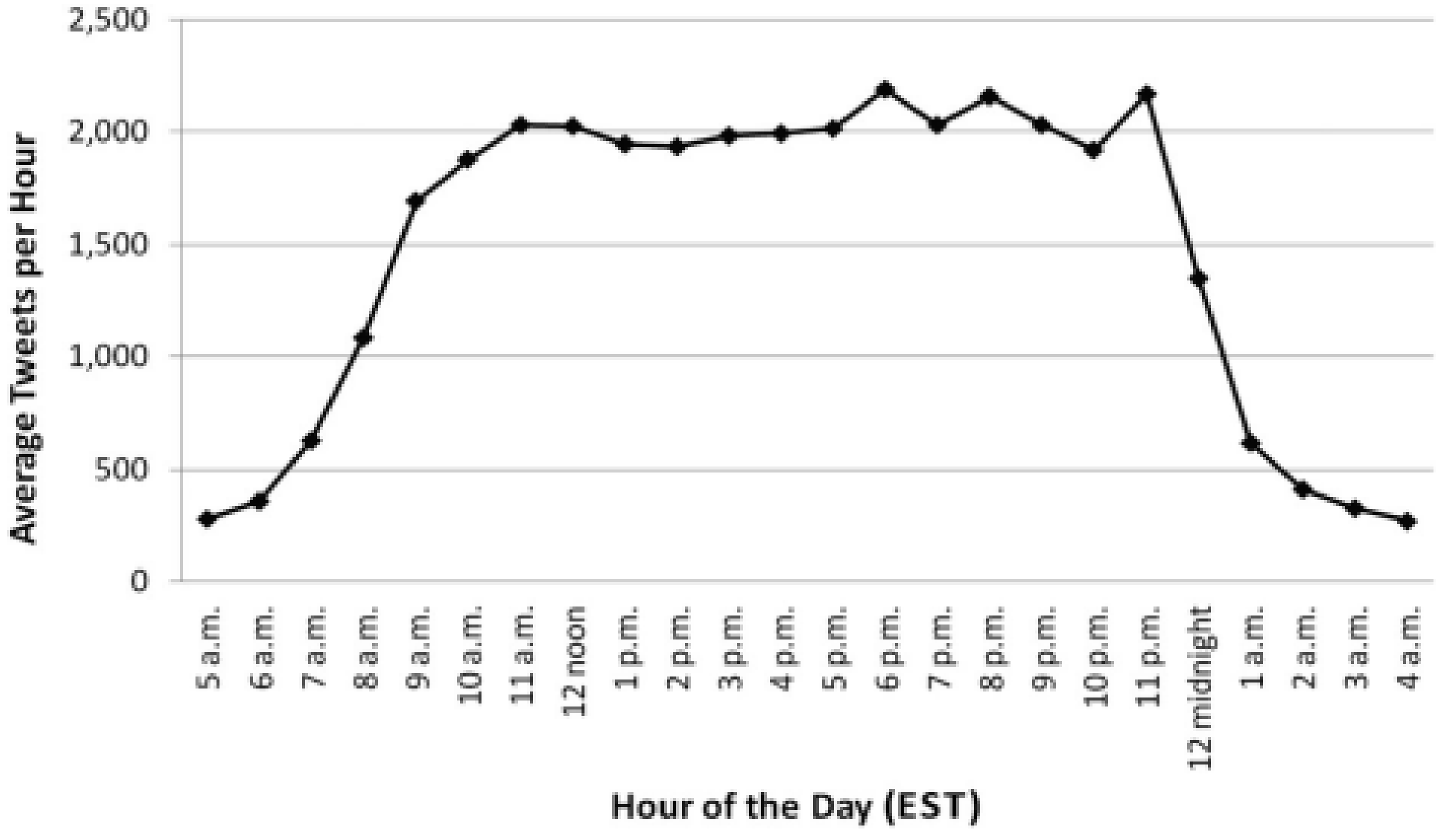
Table 2. IC coding categories

IC Element	Definition
Node	A center of activity “which the observer can enter, and which are intensive foci to and from which he [sic] is traveling” (Lynch, 1960: 72)
Landmark	Prominent visual feature; for Lynch, typically not entered but observable. For the purpose of this study we include in this category large buildings and structures with visual "referenceability" when providing directions.
Path	"Major and minor routes of circulation" (47)
Edge	Dividing lines between districts; "linear elements not used or considered as paths" (62)
District	"Medium to large sections of the city … recognized as having some common identifying character" (67)

Distribution of tweets per day during the observation period

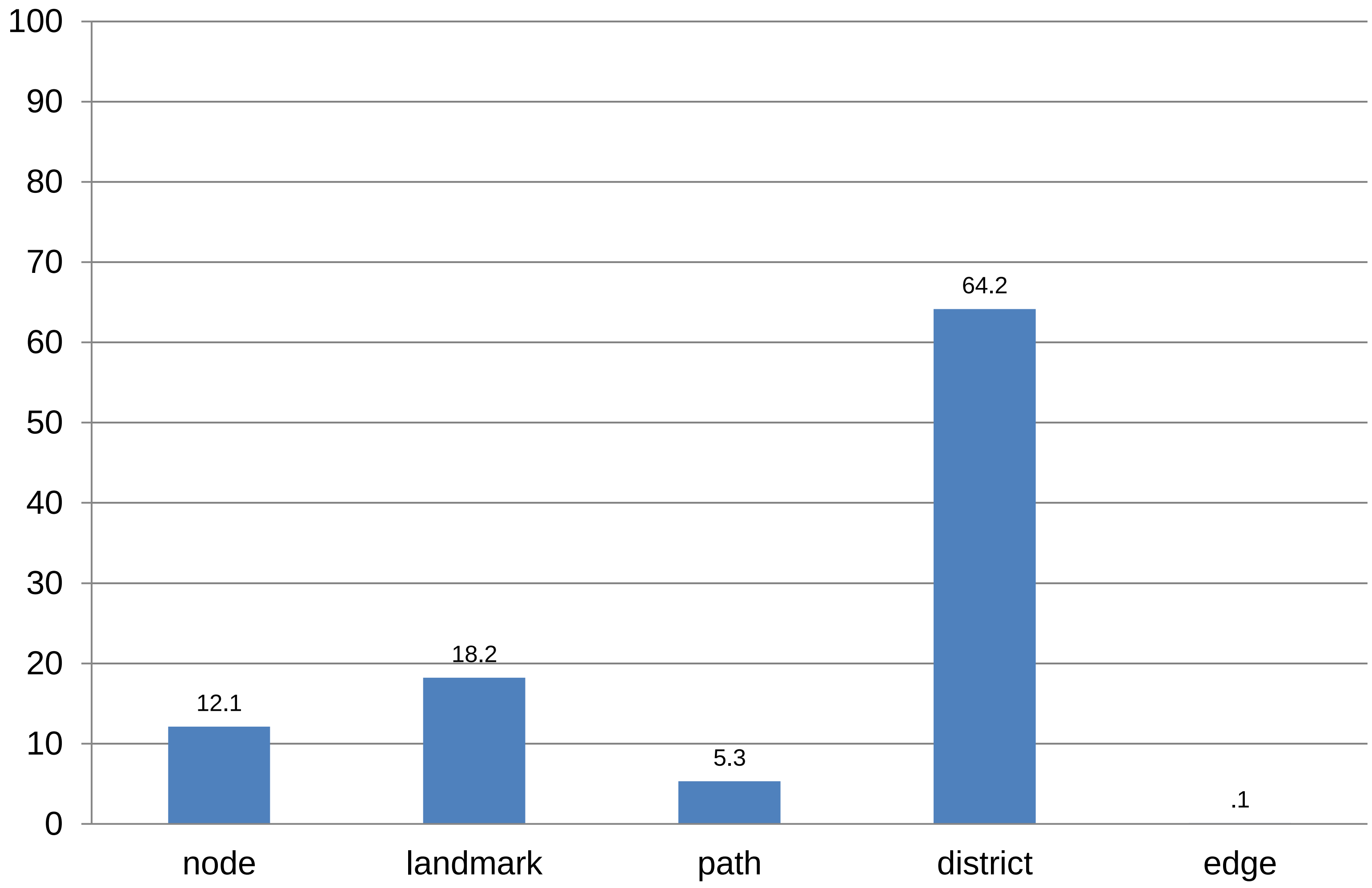


Distribution of average tweets per hour during the observation period



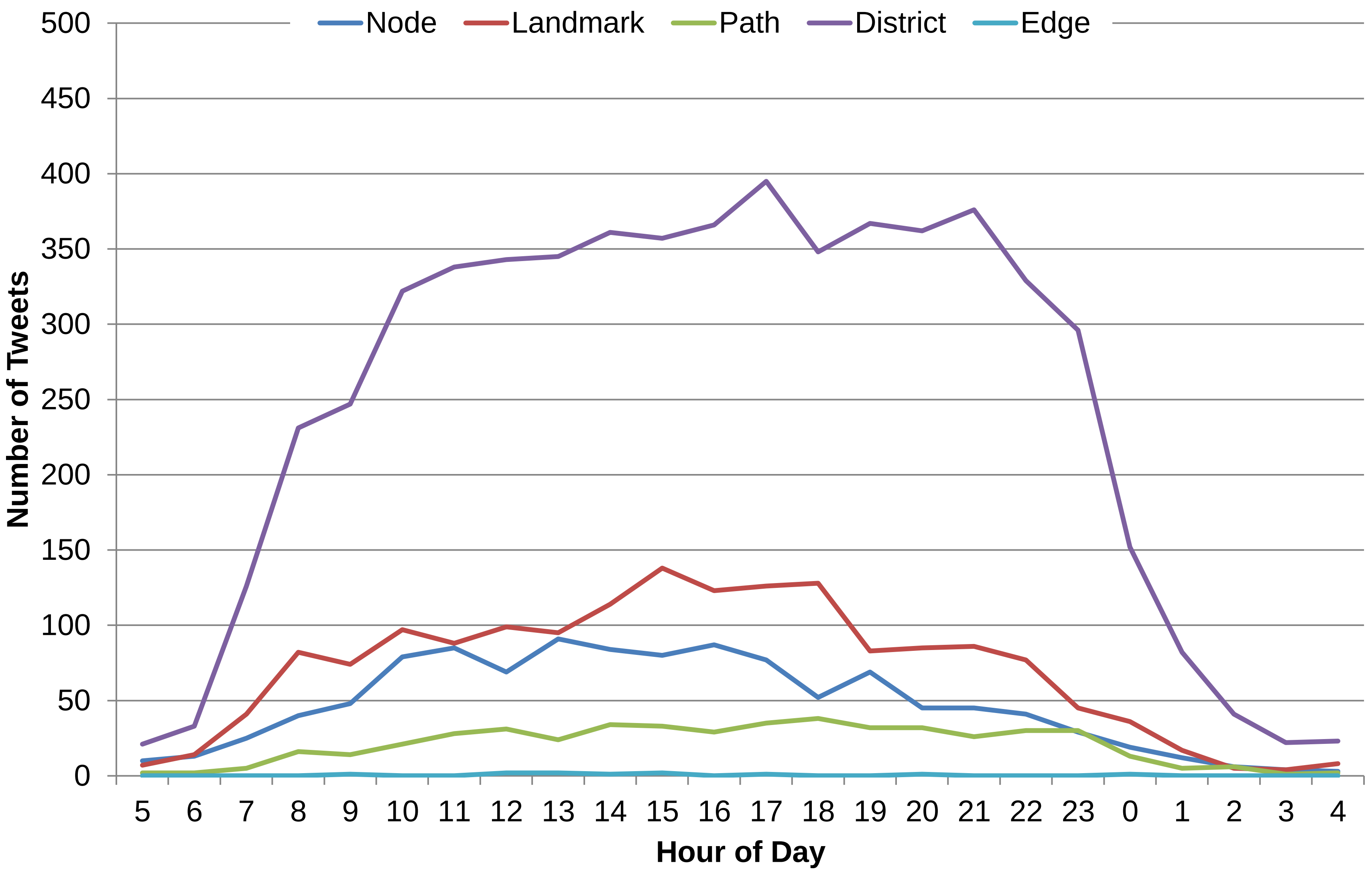
Overall tweets distribution (Lynch)

For the five Image of the City spatial reference categories (n=9103)

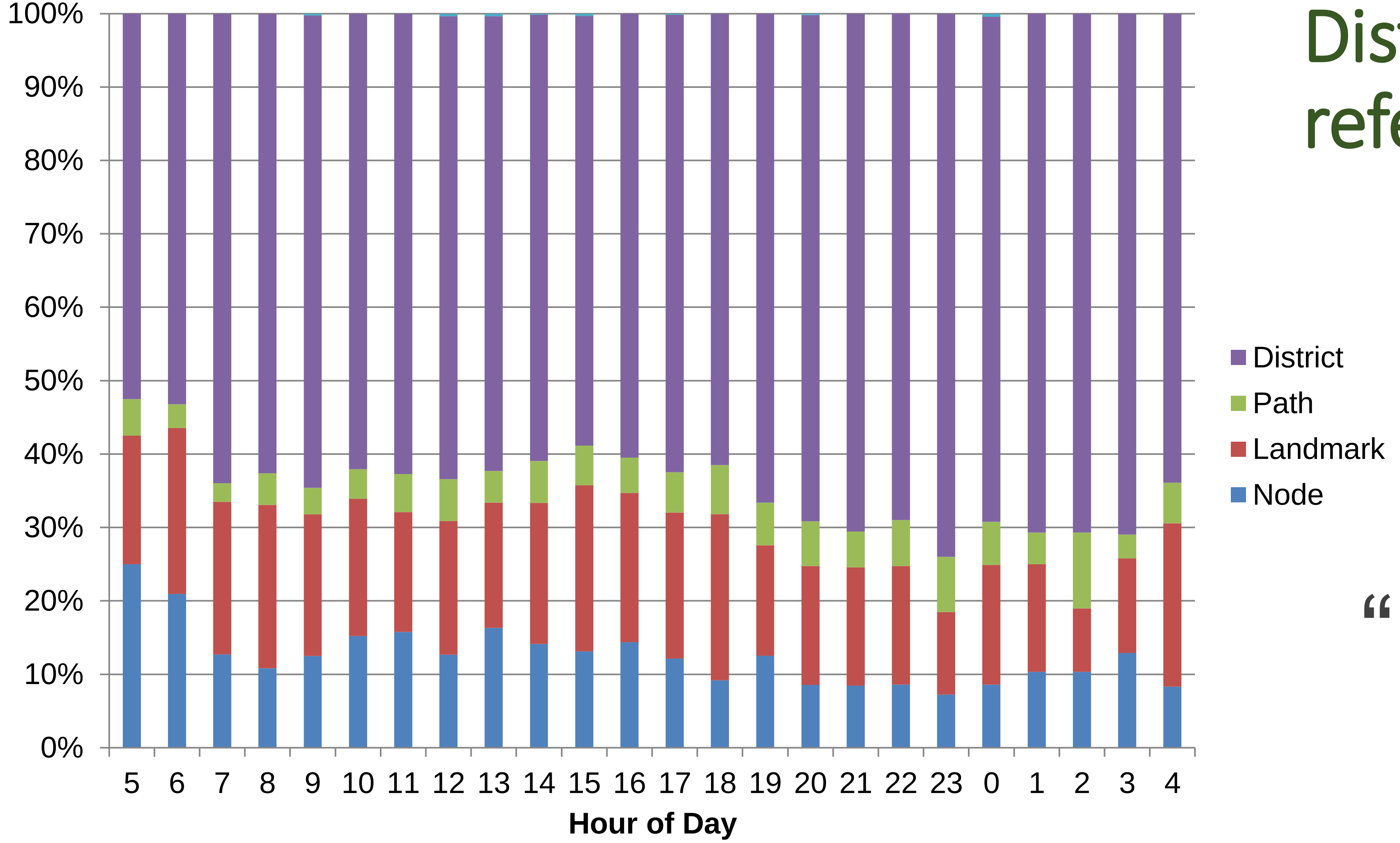


Average tweets per hour

For the five Image of the City spatial reference categories



Distribution of spatial references

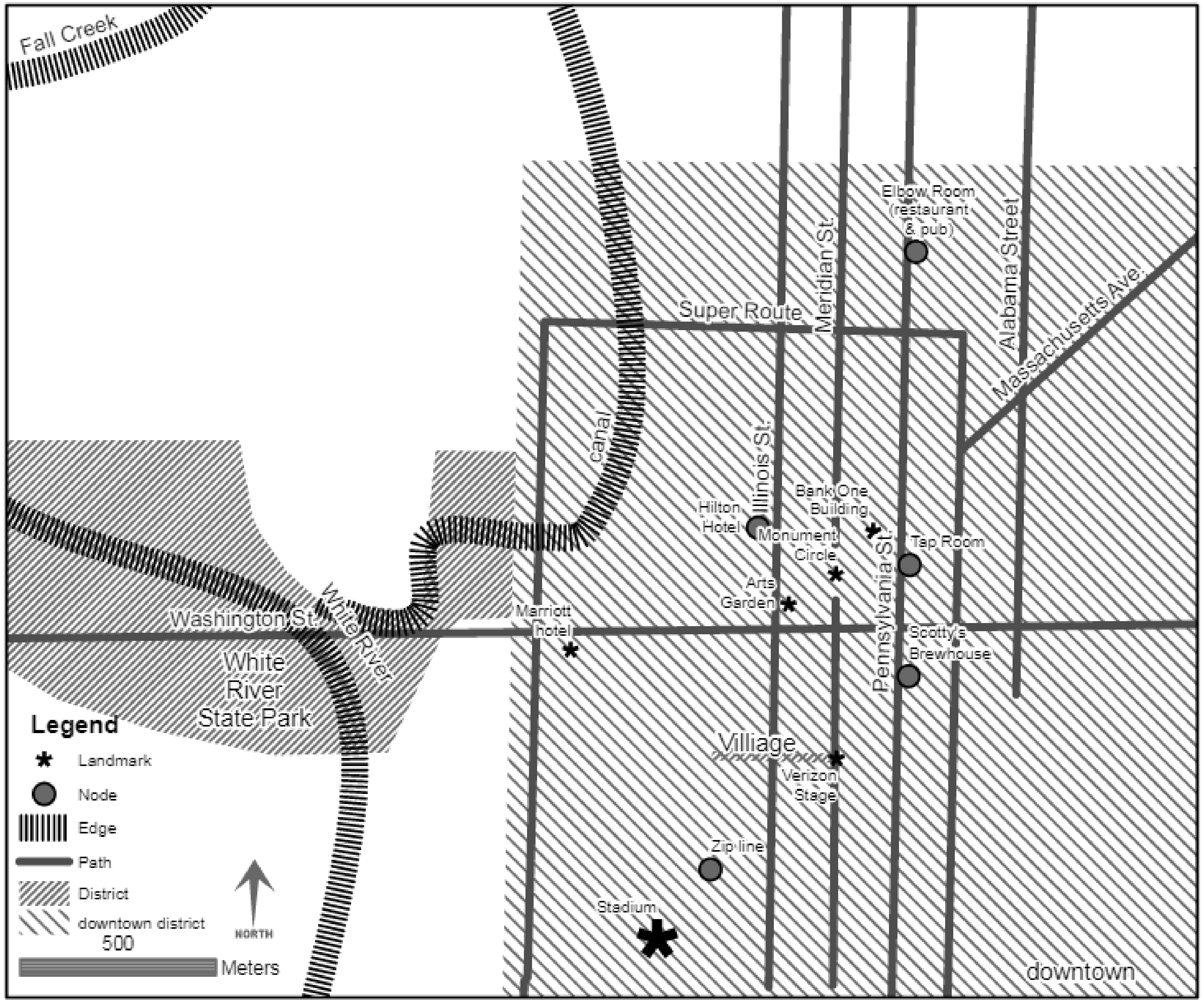


“No Edge”

Table 4. Most frequently occurring spatial terms and their co-occurrence with representatives of other IC categories.

				Frequency in Tweets Containing Other IC Elements				
Category	Rank	Total	Assigned	Node	Landmark	Path	District	Edge
1	Downtown	3261	District	293	191	193	--	4
2	Village	2791	District	305	124	131	--	1
3	Stadium	1218	Landmark	52	--	25	51	0
4	Georgia	768	(multiple)[1]	149	227	192	396	0
5	Avenue	754	(multiple)	158	122	199	447	1
6	Zipline	526	Node	--	12	11	196	1
7	Monument Circle	493	Landmark	80	--	32	133	0
8	Street	378	Path	32	26		142	0
9	Hotel	370	(multiple)	254	158	7	86	0
10	NFL Experience	362	Node	--	15	9	163	0

[1]Assigned category is dependent on semantic context and if used alone or in a phrase

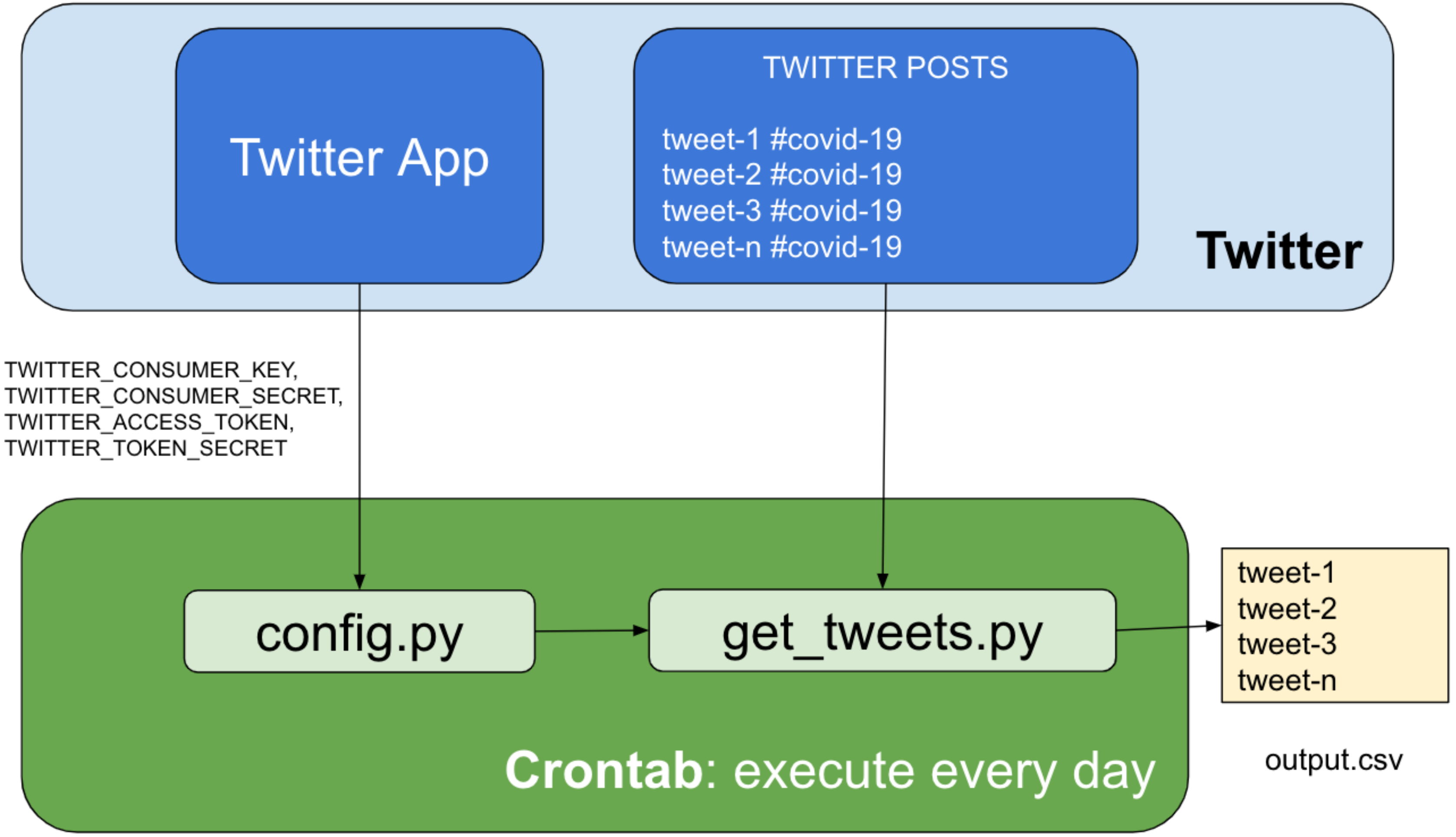


Frequent spatial tweets

Mapped in the Image of the City typology

Conclusions

- Four of the five Lynch's elements were found in the analysis.
- District and Landmark were two most identified categories followed by node and path.
- Edge was almost non-exist in the analysis.



TWITTER POSTS
Twitter App
tweet-1 #covid-19
tweet-2 #covid-19
tweet-3 #covid-19
tweet-n #covid-19
Twitter
TWITTER_CONSUMER_KEY,
TWITTER_CONSUMER_SECRET,
TWITTER_ACCESS_TOKEN,
TWITTER_TOKEN_SECRET
config.py
get_tweets.py
tweet-1
tweet-2
tweet-3
tweet-n
output.csv
Crontab: execute every day

Lab Session : Text Analysis with Twitter
(Source Twitter)

Work Hard

End of Doc.

Thank
You