Unit 3

Accessing Text Corpora and Word Level Analysis

Text Corpora

- What is Corpora: It is the plural form of corpus
- Corpus is collection of linguistics data or computer readable collection of text or speech (entire work of a particular author)
- It is a structured collection of texts used for language analysis and training language models
- Corpora can be organized in various ways, including categorized, overlapping, and temporal structures.
- They can be simple, like collections of documents with no further organization, or highly structured, with annotations for grammatical information
- Example: collection of writing or speeches of a person

Text Corpora

- A text corpus is a large body of text
- Many corpora are designed to contain a careful balance of material in one or more genres
- Example of various corpus available in nltk
  - Gutenberg corpus
  - Web and Chat corpus
  - Brown corpus
  - Reuters corpus
  - Inaugural Address corpus
  - Annotated text corpus
  - Corpora in other languages

Data Acquisition

- Data is the heart of any ML system
- In most industrial projects, it is often the data that becomes the bottleneck
- Example
  - We have to develop an NLP system to identify whether an incoming customer query is a **sales inquiry** or a **customer care inquiry**
  - We use a chat interface
  - Depending on the type of query, it should be automatically routed to the right team
  - How can one go about building such a system?
  - Answer depends on the type and amount of data we have to work with

Data Acquisition

Ideal setting

- We have the required datasets with thousands or even millions of data points
- In such cases, we don't have to worry about data acquisition
- Example Scenario
    - We have historic queries from previous years, which sales and support teams responded to
    - Further, the teams tagged these queries as sales, support, or other
    - So, not only do we have the data, but we also have the labels

Data Acquisition

Less than ideal scenario

- We have little or no data
- We can start by **looking at patterns** in the data that indicate if the incoming message is a sales or support query
- We can then **use regular expressions** and other **heuristics** to match these patterns to separate sales queries from support queries
- We evaluate this solution by **collecting a set of queries** from both categories and calculating what percentage of the messages were correctly identified by our system
- Say we get OK-ish numbers
- We would like to improve the system performance
- Simultaneously we will start building our data set

Ways of Acquiring Data

Use a public Dataset

- We could see if there are any public datasets available that we can leverage

Scrape Data

- We could find a source of relevant data on the internet
- For example, a consumer or discussion forum where people hove posted queries (sales or support)
- Scrape the data from there and get it labeled by human annotators.
- For many industrial settings, gathering data from external sources does not suffice because the data doesn't contain nuances like product names or product specific user behavior and thus might be very different from the data seen in production environments
- This is when we'll have to start looking for data inside the organization.

Ways of Acquiring Data

Product Intervention

- Instead of collecting data passively or through backend logs alone, product teams often introduce interactive elements that encourage users to voluntarily provide useful information.
- product intervention for acquiring data is a strategic approach where product features and interactions are crafted to elicit valuable information from users.
- In the tech world, this is called product intervention
- Product intervention is often the best way to collect data for building intelligent applications in industrial settings
- Tech giants like Google, Facebook, Microsoft, Netflix, etc., have known this for a long time *and* have tried to collect as much data as possible from as many users as possible
- For example,
    - during user onboarding, a product might ask questions about a user's preferences, goals, or background. This not only helps the system personalize content but also provides labeled data that can be used to improve algorithms.
    - Another common form of product intervention is feedback prompts — such as asking users to rate an article, tag a photo, or answer "Was this helpful?" after reading a help article.
    - Gamification is another powerful form of product intervention.

Ways of Acquiring Data

Data Augmentation

- While instrumenting products is a great way to collect data, it takes time
- Even if you instrument the product today, it can take anywhere between three to six months to collect a decent-sized, comprehensive dataset
- Data Augmentation techniques can be used to create more data
- They try to **exploit language properties** to create text that is syntactically similar to source text data
- They may appear as hacks, but they work very well in practice
- Some techniques are
    - Synonym Replacement, Back Translation, TFIDF Based Word Replacement, Bigram Flipping, Replacing Emilies, Adding Noise to Data

Data Augmentation

Synonym Replacement

Randomly choose “k" words in a sentence that are not stop words

Replace these words with their synonyms

For synonyms, we can use **Synsets** in Wordnet

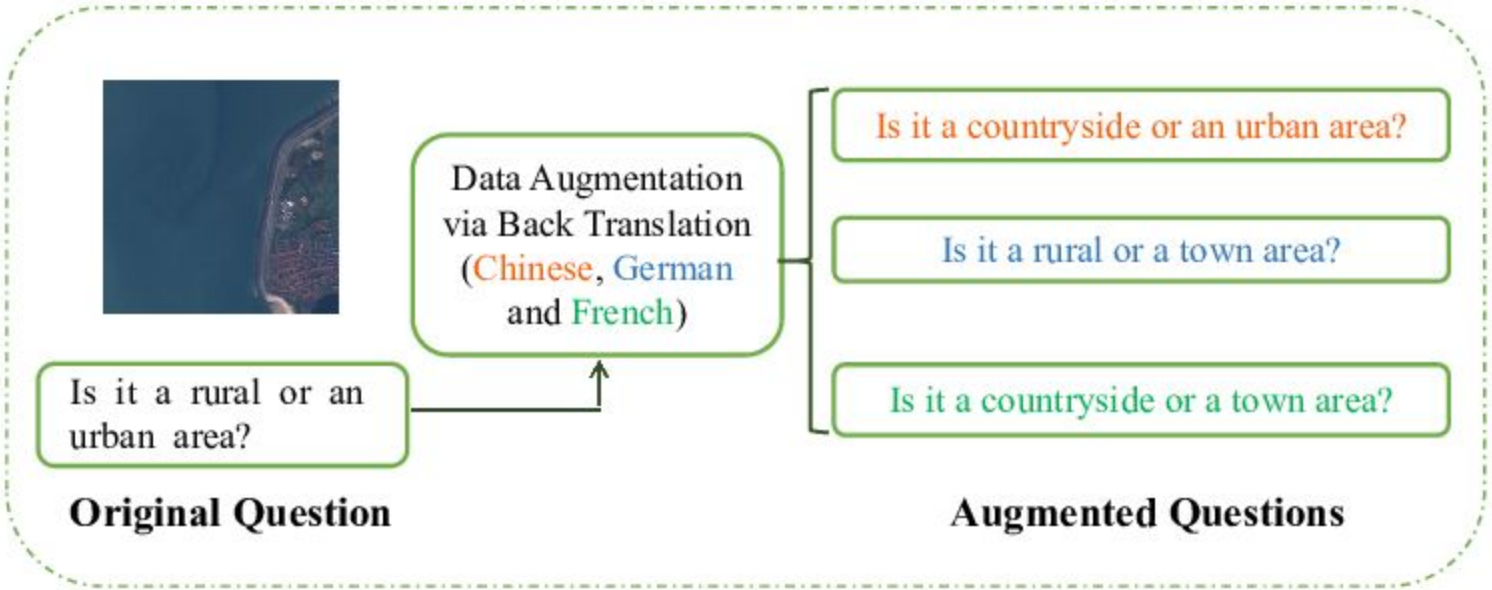
Data Augmentation

Back Translation

- Say we have a sentence, S1, in English
- We use a machine-translation library like Google Translate to translate it into some other language say German
- Let the corresponding sentence in German be S2
- Now, we'll use the machine-translation library again to translate back to English
- Let the output sentence be S3
- We'll find that S1 and S3 are very similar in meaning but are slight variations of each other
- Now we can add S3 to our dataset

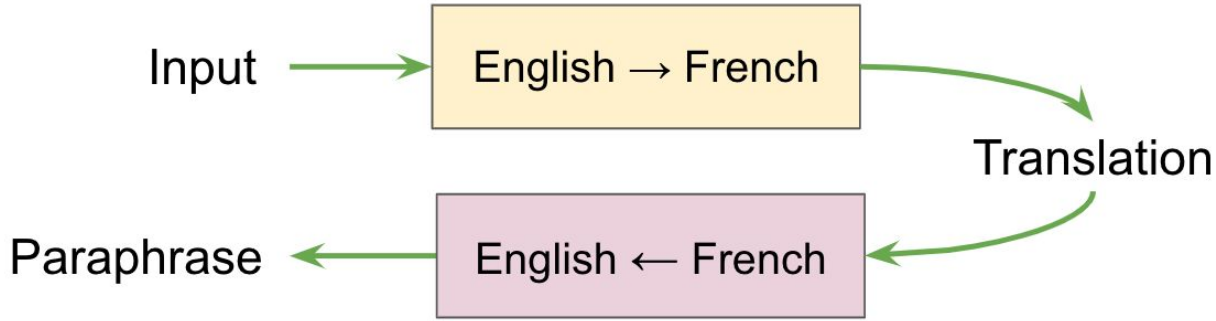
This trick works beautifully for text classification

Back Translation



Back Translation

Previously, tea had been used primarily for Buddhist monks to stay awake during meditation.



Autrefois, le thé avait été utilisé surtout pour les moines bouddhistes pour rester éveillé pendant la méditation.

In the past, tea was used mostly for Buddhist monks to stay awake during the meditation.

Data Augmentation

TFIDF Based Replacement

- Back translation can lose certain words that are crucial to the sentence.
- TF-IDF, can be used to handle this

DIY- Find out more on how this is done

Data Augmentation

Bigram Flipping

- Divide the sentence into bigrams
- Take one bigram at random and flip it
- For examples:
  - “I am going to the supermarket."
  - Here, we take the bigram “going to" and replace it with the flipped one: “to going."
  - **Original Sentence:** "The quick brown fox jumps over the lazy dog."
  - **Bigram Flipped Sentence:** "The quick fox brown jumps over the lazy dog."
- Bigram flipping can **slightly distort grammar** but is often useful for training models to be more robust to variations in word order.
- It works better for tasks where **small syntactic changes don't greatly affect** the outcome, like sentiment analysis.
- Avoid using this technique in tasks highly sensitive to word order

Data Augmentation

Replacing entities

- Replace entities like person name, location, organization, etc., with other entities in the same category
- That is, replace person name with another person name, city with another city, etc
- For example, in “I live in California,” replace “California” with “London.”

Data Augmentation

Adding noise to data

- In many NLP applications, the incoming data contains spelling mistakes
- This is primarily due to characteristics of the platform where the data is being generated (for example, Twitter)
- In such cases, we can add a bit of noise to dato to train robust models
- For example, randomly choose a word in a sentence and replace it with another word that's closer in spelling to the first word
- Another source of noise is the “fat finger” problem on mobile keyboards
- Simulate a QWERTY keyboard error by replacing a few characters with their neighboring characters on the QWERTY keyboard

Text corpus for NLP

- In NLP there is a need to train models on lots of data
- For this purpose, researchers have assembled many text corpora
- Common corpus is also useful for benchmarking various models
- Corpus can be used for tasks such as word sense disambiguation, coreference resolution, machine translation, part of speech tagging

Ref: Devopedlo. 2020. "Text Cc›rpus fc'r NLP." Version J, December 20. https://devc'pedTo.org/text-corpus-for-nip

Some traits of a good corpus

- **Depth**: A wordlist, for instance, should include the top 60K words and not just the top 3K words.
- **Recent**: Corpus based on outdated texts is not going to suit today's tasks.
- **Metadata**: Metadata should indicate the sources, assumptions, limitations and what's included in the corpus.
- **Genre**: Unless corpus has been collected for specific tasks, it should include different genres such as newspapers, magazines, blogs, academic journals, etc.
- **Size**: A corpus of half a million words or more ensures that low frequency words ore also adequately represented.
- **Clean**: A wordlist giving word forms of the same word can be messy to process. A better corpus would include only the lemma and part of speech
- **Balance**: Balance of material in one or more genres

Challenges to create a good corpus

- Deciding the type of data needed to solve the problem statement
- Availability of data
- Quality of the data
- Adequacy of the data in terms of the amount

Text corpus structure

- Simplest kind Lacks any structure. It is just a collection of texts
- Often, texts are grouped into categories that might correspond to genre, source, author, language, etc.
- Sometimes these categories overlap, notably in the case of topical categories as a text can be relevant to more than one topic
- Occasionally, text collections have temporal structure, news collections being the most common Example

Text corpus structure

Raw Text Corpus

- Just plain text files or a single large text file.

Document 1: "Age is just a number."

Document 2: "A journey of a thousand miles begins with a single step."

Document 3: "To be or not to be, that is the question."

Structured Corpus (Document-based)

- Organized as multiple documents, often stored in folders or with metadata.
- Each document is an entry.
- May include metadata such as author, date, title, labels, etc.


```
[
  {
    "doc_id": "001",
    "title": "Fox Story",
    "author": "John Doe",
    "text": "The quick brown fox jumps over the lazy dog."
  },
  {
    "doc_id": "002",
    "title": "Proverb",
    "author": "Anonymous",
    "text": "A journey of a thousand miles begins with a single step."
  }
]
```

Text corpus structure

Tokenized Corpus

- Corpus where text is already split into tokens (words, punctuation).

```
[
  ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."],
  ["A", "journey", "of", "a", "thousand", "miles", "begins", "with", "a", "single", "step", "."]
]
```

Annotated Corpus

Includes linguistic annotations like POS tags, syntactic parses, or semantic labels.

Example (word-level with POS tags):

```
[
  [("The", "DT"), ("quick", "JJ"), ("brown", "JJ"), ("fox", "NN"), ("jumps", "VBZ"), ...],
  [("A", "DT"), ("journey", "NN"), ("of", "IN"), ("a", "DT"), ...]
]
```

Parallel Corpus

Used in machine translation, includes sentence pairs in two or more languages.

English	Spanish
The quick brown fox jumps.	El rápido zorro marrón salta.
A journey of a thousand miles.	Un viaje de mil millas.

Summary Table:

Corpus Type	Structure	Use Cases
Raw Text	Plain text files	Language modeling, unsupervised NLP
Document-based	JSON, XML, folders with docs	Information retrieval, topic modeling
Tokenized	List of token lists	Text classification, embeddings
Annotated	Tokens + labels (POS, NER, etc.)	Linguistic research, supervised learning
Parallel	Sentence pairs across languages	Machine translation

Gutenberg Corpus

- NLTK includes a small selection of texts from the Project Gutenberg electronic text archive
- Project Gutenberg contains some 70,000 free electronic books
- About Project Gutenberg
  - Project Gutenberg is an online library of free eBooks.
  - Project Gutenberg was the first provider of free electronic books, or eBooks
  - Michael Hart, founder of Project Gutenberg, invented eBooks in 1971 *and* his memory continues to inspire the creation of eBooks and related content today

Project Gutenberg is a library of over 75,000 free eBooks

Choose among free epub and Kindle eBooks, download them or read them online. You will find the world's great literature here, with focus on older works for which U.S. copyright has expired. Thousands of volunteers digitiz proofread the eBooks, for you to enjoy.

Our Newest Releases click here for more

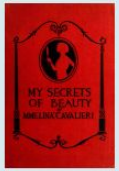
A blighted life by Lady Rosina Lytton

Meijeri by Artturi Järviluoma

Ihinen ja peto by Jr. Samuel Scoville and

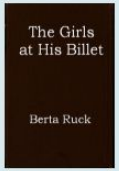
Der Witz und seine Beziehung

The symbolism of colour by Ellen



My secrets of beauty by Lina Cavalieri

The wooing of Rosamond Fayre by Berta



The girls at his billet by Berta Ruck

The tomb of Ts'in by Edgar Wallace

The children by Edith Wharton

Find Free eBooks

- Search Options. By author, title, subject, language, type, popularity, and more.
- Main Categories. The ones you'd find in any large bookstore.
- Reading Lists. Hand-curated by volunteers.
- Frequently downloaded: Top 100, or ranked by popularity.

Quick Summary of Fields in NLTK's Gutenberg Corpus

Field	Description	Example Usage
`raw()`	Raw, unprocessed text (string format)	`gutenberg.raw('austen-emma.txt')`
`words()`	List of word tokens (words + punctuation)	`gutenberg.words('austen-emma.txt')`
`sents()`	List of sentences, each sentence is a list of word tokens	`gutenberg.sents('austen-emma.txt')`
`fileids()`	List of all available text files in the corpus	`gutenberg.fileids()`
`metadata()`	Metadata associated with the text (usually empty)	`gutenberg.metadata('austen-emma.txt')`

```
from nltk.corpus import gutenberg

# List all available texts in Gutenberg corpus
print("Available texts:", gutenberg.fileids())

# Access raw text of "Emma"
emma_raw = gutenberg.raw('austen-emma.txt')
print("Raw text preview:", emma_raw[:500])  # Preview the first 500 characters

# Tokenize words in "Emma"
emma_words = gutenberg.words('austen-emma.txt')
print("First 10 words:", emma_words[:10])

# Tokenize sentences in "Emma"
emma_sents = gutenberg.sents('austen-emma.txt')
print("First 2 sentences:", emma_sents[:2])

# Access metadata (if available)
emma_metadata = gutenberg.metadata('austen-emma.txt')
print("Metadata:", emma_metadata)
```

Output

Available texts: ['austen-emma.txt', 'austen-persuasion.txt', 'bible-kjv.txt', 'blake-poems.txt', ...]
Raw text preview: [First 500 characters of the raw text]
First 10 words: ['[', 'The', 'Project', 'Gutenberg', 'EBook', 'of', 'Emma', ',', 'by', 'Jane']
First 2 sentences: [['[', 'The', 'Project', 'Gutenberg', 'EBook', 'of', 'Emma', ',', 'by', 'Jane', 'Austen', 'EBook', 'No', '.'], ['Emma', 'by', 'Jane', 'Austen', 'Chapter', '1', '.', 'Emma', 'Woodhouse', 'has', 'been', 'too', 'long', 'the', 'mistress', 'of', 'her', 'father's', 'house', 'to', 'allow', 'her', 'to', 'feel', 'the', 'importance', 'of', 'her', 'own', 'conduct', '.']]
Metadata: None

Web and Chat text

- Although Project Gutenberg contains thousands of books, it represents established literature
- It is important to consider less formal language as well

The NLTK webtext corpus includes the following files:

1. **Firefox.txt**
   - Mozilla Firefox web browser discussions **from a newsgroup.**
2. **grail.txt**
   - Excerpt from the *Monty Python and the Holy Grail* **movie script**.
3. **overheard.txt**
   - **Short quotes** from the "Overheard in New York" website (humorous snippets of real conversations).
4. **pirates.txt**
   - Dialog from the movie *Pirates of the Caribbean*.
5. **singles.txt**
   - **Online dating ads** (presumably collected from a website or forum).
6. **wine.txt**
   - A collection of **wine reviews** (brief descriptions and evaluations of different wines).

You can access this corpus in Python using:

```
import nltk
nltk.download('webtext')
from nltk.corpus import webtext
# List file names
print(webtext.fileids())
```

Output: ['firefox.txt', 'grail.txt', 'overheard.txt', 'pirates.txt', 'singles.txt', 'wine.txt']

```
# Read content from a file

print(webtext.raw('wine.txt')[:500])  # Show first 500 characters of wine reviews
```

Output:

Lovely delicate, fragrant Rhone wine, a bit short on the finish

Delicate almost fragile wine with a nice aroma, though a bit short in the mouth

Very fruity, fresh, and vivid. Stylish and uncomplicated

Fresh, lively, grapey, and clean. A nice picnic wine

Fresh and grapey, with a good finish, some complexity

Rather old-fashioned style. Charming and honest

Super-ripe and a bit raisiny. A wine to appreciate on its own

Very ripe fruit. A hint of leafiness in the nose

Soft and round with good fruit and a hint of spice. A delicious wine

Web and Chat text

- There is also a corpus of **instant messaging chat sessions**, originally collected by the Naval Postgraduate School for research on **automatic detection of Internet Predators**
- Corpus contains over 10,000 posts, anonymized by replacing usernames with generic names of the form"UserNNN”
- It is manually edited to remove any other identifying information
- Corpus is organized into 15 files, where each file contains several hundred posts collected on a given date, for an age-specific chatroom (teens, 20s, 30s, 40s, plus *a* generic adults chat room)
- The filename contains the date chatroom, and number of posts
- It contains
    - 706 posts
    - gathered from the 20s chat room on 10/19/2006

Brown Corpus

- The Brown Corpus was the **first million-word electronic corpus** of English, created in 1961 at Brown University.
- This corpus contains text from 500 sources, and the sources have been categorized by genre, such as news, editorial, and so on
- We can access the corpus as a list of words, or a list of sentences (where each sentence is itself just a list of words)
- We can optionally specify particular categories or files to read

Category	Genre (Code)	# of texts	Total Tokens	%
INFORMATIVE	Learned (J)	80	160,000	16.0%
INFORMATIVE	Belles Lettres, Biography, Memoirs, etc (G)	75	150,000	15.0%
INFORMATIVE	Popular Lore (F)	48	96,000	9.6%
INFORMATIVE	Press: Reportage (A)	44	88,000	8.8%
INFORMATIVE	Skills and Hobbies (E)	36	72,000	7.2%
INFORMATIVE	Miscellaneous (H)	30	60,000	6.0%
IMAGINATIVE	General Fiction (K)	29	58,000	5.8%
IMAGINATIVE	Adventure and Western Fiction (N)	29	58,000	5.8%
IMAGINATIVE	Romance and Love Story (P)	29	58,000	5.8%
INFORMATIVE	Press: Editorial (B)	27	54,000	5.4%
IMAGINATIVE	Mystery and Detective Fiction (L)	24	48,000	4.8%
INFORMATIVE	Press: Reviews (theatre, books, music, dance) (C)	17	34,000	3.4%
INFORMATIVE	Religion (D)	17	34,000	3.4%
IMAGINATIVE	Humor (R)	9	18,000	1.8%
IMAGINATIVE	Science Fiction (M)	6	12,000	1.2%
	TOTAL	500	1,000,000	100.0%

```
import nltk
from nltk.corpus import brown
# Download Brown corpus if not already downloaded
nltk.download('brown')

# List the categories of the Brown corpus
categories = brown.categories()
print("Categories in the Brown corpus:", categories)

# Get a sample from the "news" category (E1)
news_category = brown.words(categories='news')
print(news_category[:50])  # Print the first 50 words from the 'news' category
```

Output:

['adventure', 'belles_lettres', 'bottom', 'cambridge', 'classics', 'commerce', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'news', 'religion', 'science_fiction']

['The', 'F.B.I.', 'said', 'today', 'it', 'had', 'arrested', 'two', 'persons', 'involved', 'in', 'the', 'robbery', 'case', '.', 'The', 'suspects', 'were', 'identified', 'as', 'John', 'Doe', 'and', 'Jane', 'Smith', '.', 'Both', 'have', 'been', 'charged', 'with', 'the', 'theft', 'of', '$', '500', 'from', 'a', 'local', 'bank', '.']

Reuter corpus

- Reuters corpus contains 10,788 **news documents** totaling **1.3 million words.**
- Documents have been classified into 90 topics, and grouped into two sets, called "training" and "test"
- Unlike the Brown Corpus, categories in the Reuters corpus **overlap** with each other, simply because a news story often covers multiple topics
- We can ask for the topics covered by one or more documents, or for the documents included in one or more categories
- For convenience, the corpus methods accept a simple fileid or a list of fileids

```
import nltk
from nltk.corpus import reuters
nltk.download('reuters')
# List of file IDs
print(reuters.fileids()[:5])  # Show first 5 file IDs
```

['test/14826', 'test/14828', 'test/14829', 'test/14832', 'test/14833']

```
# List of categories (topics)
print(reuters.categories()[:10])  # Show first 10 categories
```

['acq', 'alum', 'barley', 'bop', 'carcass', 'castor-oil', 'cocoa', 'coconut', 'coconut-oil', 'coffee']

```
# Categories for a specific document
print(reuters.categories('training/9865'))
```

['barley', 'corn', 'grain', 'wheat']

```
# Raw text of a document
print(reuters.raw('training/9865')[:300])  # First 300 characters
```

FRENCH FARMERS PLAN MORE PROTESTS

French farmers, who have been holding up trains and blocking roads to protest ...

Inaugural Address Corpus

- The **Inaugural Address Corpus** in NLTK is a collection of **U.S. presidential inaugural speeches**, used frequently in NLP and text analysis for studying political language, trends over time, and stylistic patterns.
- Corpus is actually a collection of 55 texts, one for each presidential address
- An interesting property of this collection is its **time dimension**

```
import nltk
from nltk.corpus import inaugural

nltk.download('inaugural')

# List of inaugural speech files
print(inaugural.fileids()[:5])
        ['1789-Washington.txt', '1793-Washington.txt', '1797-Adams.txt', '1801-Jefferson.txt', '1805-Jefferson.txt']

# Words from a specific speech
print(inaugural.words('1789-Washington.txt')[:20])
        ['Fellow-Citizens', 'of', 'the', 'Senate', 'and', 'of', 'the', 'House', 'of', 'Representatives', ':', 'Among', 'the', 'vicissitudes', 'incident', 'to', 'life', ',', 'no', 'event']

# Years of speeches
print([file[:4] for file in inaugural.fileids()])
        ['1789', '1793', '1797', '1801', '1805', '1809', '1813', '1817', '1821', '1825']
```

Example

Words used the most in PM's speech on 76’h Independence day in 2022

While Amrit Kal/Amrit Mahotsav was used 32 times.

India was used 31 times.

Apart from this, the Prime Minister used the word Sankalp 26 times, dream and 75 years 24 times.

World 23 and corruption 17 times.

Annotated text corpora

- **Annotated text corpora** are collections of text data where additional layers of information (annotations) have been added to the raw text.
- These annotations could include various linguistic features, such as:
  - POS tags
  - named entities recognition (NER)
  - syntactic structures
  - semantic roles, and so forth

Table 2 : Examples for Case 1

Post	Severity value	Review
She low life assclown obviously elected voter fraud	0.410707	Harsh
arparkschneider He ugly little bastard he mamas boy	0.410707	Harsh
Them whats dinner tonight bitch videos sitting well	0.410707	Harsh
realDonaldTrump You want less testing less cases What f ing assface dope	0.410397	Harsh
And I SCREW YOU JACK horse your dumbass rode I major problem little privileged asshat thinking going censor go hell	0.410016	Harsh
bug looking ass bitch I even lick your ear loudlycryingface	0.441247	Very Harsh
Shit baby abysmal record Republicans You guys proven you govern Wasn til Trump Pres trouble Thanks bringing dumshit	0.409991	Less Harsh
There you go islamic terrorist way betraying Is Muslims jihadi arsehole betray Is Quran teaches	0.409931	Less Harsh

	Tweet Id	Text	Username	UserID	Verifed	Followers	Friends Co	Media Cou	Reply Cou	Retweet c	Like count	algo label	Severity v	Profanity	Sentence	Semantic	Capital v
2	1.36E+18	BlackBinLi	trollingcla	8.68E+17	FALSE	54	261	791	0	0	6	Severe Bu	0.46340	0.333333	0.814286	1	0.07692
3	1.36E+18	xjasmines	harryrobs	4.78E+09	FALSE	2007	424	14	2	0	0	Nasty	0.194821	0.2	0.885714	0	0.0312
4	1.36E+18	ForumAth	magusnik	2.86E+09	FALSE	734	799	10044	0	0	4	Nasty	0.128263	0.125	0.603571	0	0.02702
5	1.36E+18	So many f	county201	4E+09	FALSE	842	1001	2763	1	0	1	Nasty	0.151851	0.181818	0.546429	0	0.03149
6	1.36E+18	aldaricojr	AnthonyJl	18725885	FALSE	842	3792	376	0	0	1	Light Bully	0.265487	0.034483	0.432143	1	0.02515
7	1.36E+18	Minibeuu	beetlebut	1.27E+18	FALSE	16	98	35	0	0	0	Light Bully	0.555693	0.5	0.939286	1	0.05882
8	1.36E+18	leftistrage	BecauseR	1.28E+18	FALSE	2552	2432	2306	1	0	6	Medium B	0.371619	0.166667	0.642857	1	0.1
9	1.36E+18	hands col	charlotter	98872009	FALSE	1215	2113	4962	0	0	0	Nasty	0.194325	0.222222	0.832143	0	
10	1.36E+18	tweetalat	Michael_N	9.48E+17	FALSE	138	567	64	0	0	0	Nasty	0.182399	0.2	0.721429	0	0.05128
11	1.36E+18	SkyNews l	arachnais	36852483	FALSE	272	738	808	0	0	0	Severe Bu	0.443373	0.3	0.775	1	0.07936
12	1.36E+18	tallsty mr	TheNightF	21768706	FALSE	196	541	667	1	0	0	Nasty	0.127049	0.125	0.592857	0	0.02631
13	1.36E+18	Claire real	sammstor	95061480	FALSE	567	3003	2204	0	0	0	Medium B	0.296949	0.083333	0.417857	1	0.06740
14	1.36E+18	lymndillo	ScoopyCh	1.26E+18	FALSE	49	104	920	0	0	0	Medium B	0.282756	0.107143	0.253571	1	0.01913
15	1.36E+18	When sto	Unwise_T	8.02E+17	FALSE	7163	2659	5917	1	0	20	Nasty	0.108474	0.05	0.478571	0	0.17808
16	1.36E+18	DailyMirr	lisagouch	1.63E+09	FALSE	416	503	1749	0	0	0	Nasty	0.156521	0.176471	0.642857	0	0.0
17	1.36E+18	Imagine p	NoForced'	1.27E+18	FALSE	231	86	704	0	1	1	Medium B	0.336313	0.192308	0.357143	1	0.02222
18	1.36E+18	When one	jayceebro	2.25E+09	FALSE	9363	9382	16822	1	0	1	Nasty	0.256429	0.333333	0.85	0	0.0238
19	1.36E+18	How hell I	scousewe	3.41E+09	FALSE	11050	7340	3764	1	0	1	Medium B	0.309628	0.108108	0.271429	1	0.14215
20	1.36E+18	I hope ex	isitreallyn	4.59E+08	FALSE	11462	490	1513	20	3	168	Nasty	0.278214	0.375	0.857143	0	0.02
21	1.36E+18	mcstonkfa	BadComp	1.31E+18	FALSE	90	387	898	0	0	0	Light Bully	0.385357	0.2	0.853571	1	
22	1.36E+18	I noticed h	hiworld_I	1.04E+18	FALSE	265	633	265	0	0	1	Medium B	0.343187	0.173913	0.517857	1	0.02222
23	1.36E+18	AbhishBar	rai_vkr007	1.35E+18	FALSE	35	129	23	0	0	0	Nasty	0.179167	0.214286	0.625	0	0.04761
24	1.36E+18	TeaRoomI	caseywhe	1.36E+09	FALSE	137	543	55	0	0	0	Severe Bu	0.420746	0.272727	0.735714	1	0.05405
25	1.36E+18	LiamLizar	creddy_fish	4.05E+09	FALSE	3504	3300	3007	0	0	2	Medium B	0.349107	0.2	0.428571	1	0.0312
26	1.36E+18	georgiafa	Taylorand	2.19E+08	FALSE	20454	3293	9066	0	0	1	Nasty	0.220317	0.285714	0.746429	0	0.01408
27	1.36E+18	BettyBow	WandaTa	9.71E+17	FALSE	245	430	18	0	0	0	Nasty	0.353002	0.5	0.867857	0	0.08108

Corpora in other languages

NLTK comes with corporo for many languages, though in some cases you will need to learn how to manipulate character encodings in Python before using these corpora

Wordlist corpora

- **Wordlist corpora** are collections of words, often with associated metadata, used in various **Natural Language Processing (NLP)** tasks.
- These corpora are typically used for **lexical resources**, such as for building dictionaries, spell-checking, and word-related tasks like **synonym identification**, **collocation**, **word frequency analysis**, and **language modeling**.
- NLTK includes some corpora that are nothing more than wordlists
- Words Corpus is the /usr/dict/words file from Unix, used by some spell checker
- We can use it to find unusual or misspelled words in a text corpus
- There are some common **wordlist corpora** available in **NLTK** and elsewhere

Wordlist corpora

Location: nltk.corpus.words

```
from nltk.corpus import words
nltk.download('words')

# Get the first 10 words in the wordlist
wordlist = words.words()
print(wordlist[:10])
        ['a', 'aa', 'aal', 'aalii', 'aam', 'aang', 'aap', 'aaron', 'ab', 'aba']

from nltk.corpus import stopwords
nltk.download('stopwords')
# Get the first 10 stopwords in English
stop_words = stopwords.words('english')
print(stop_words[:10])
        ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your']

from nltk.corpus import names
nltk.download('names')

# Get the first 10 male names
male_names = names.words('male.txt')
print(male_names[:10])
        ['Aabid', 'Aaden', 'Aaron', 'Ab', 'Abb', 'Abdul', 'Abel', 'Abie', 'Abner', 'Abraham']
```

Example of Using Wordlist for Spell-checking

```
from nltk.corpus import words

# Function to check if a word is in the wordlist
def check_word(word):
    wordlist = set(words.words())  # Create a set of words for quick lookup
    if word.lower() in wordlist:
        return True
    else:
        return False

# Test words
print(check_word("hello"))  # True
print(check_word("helo"))   # False
```

Wordlist corpora

Task

Plot the number of female and male names ending with each letter of the alphabet

Pronouncing Dictionary

- A **pronouncing dictionary** is a dictionary that provides the **pronunciation** of words.
- It usually contains mappings of words to their corresponding phonetic representations, often in **IPA (International Phonetic Alphabet)** or another phonetic notation.

Common Use Cases:

- **Speech synthesis (text-to-speech)**: Converting written text into spoken words.
- **Speech recognition**: Mapping recognized words to their correct pronunciations.
- **Pronunciation checking**: For language learners or in NLP applications that deal with phonetic features of text.
- **Phonetic analysis**: Studying how words are pronounced in different dialects or languages.

Pronouncing Dictionary in NLTK:

- NLTK provides a **pronouncing dictionary** in the **CMU Pronouncing Dictionary** (a publicly available dictionary of English words with their phonetic transcriptions).
- **Features:**
  - The dictionary contains over **100,000 words** with their corresponding phonetic transcriptions in **ARPAbet**, a set of phonetic symbols used by the CMU pronunciation dictionary.
- The dictionary can be accessed through **nltk.corpus.cmudict**.

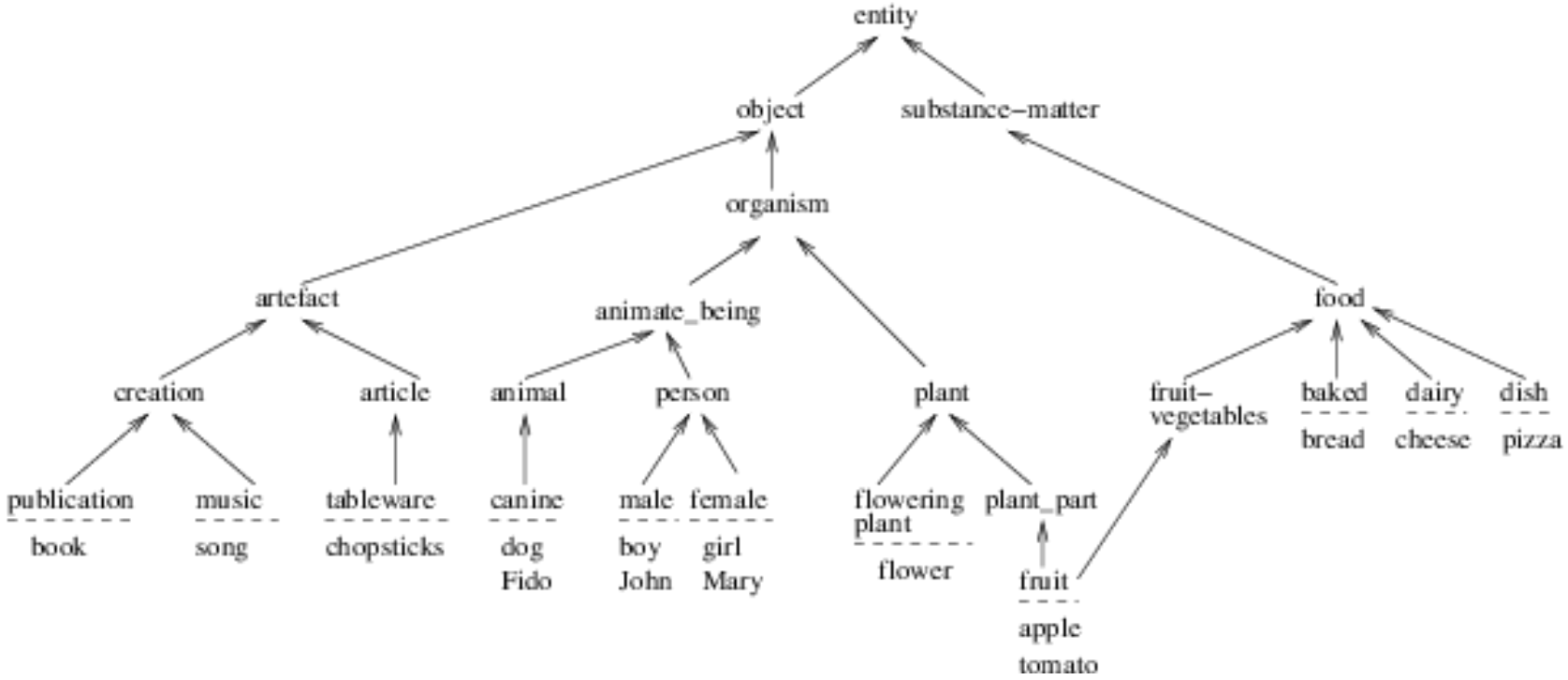
Wordnet

- **WordNet** is a large, **lexical database of English** that groups words into sets of synonyms called **synsets**, and provides **definitions**, **examples**, and **semantic relationships** between them.
- It’s widely used in **Natural Language Processing (NLP)**, **computational linguistics**, and **artificial intelligence** for tasks involving **word meaning**, **semantic similarity**, and **word sense disambiguation**.

Key Features of Wordnet

Component	Description
Synsets	Sets of **synonymous words** that share a common meaning. Each synset has a definition and usage example.
Lemmas	Specific words that are members of a synset.
POS Tags	WordNet includes **nouns**, **verbs**, **adjectives**, and **adverbs**.
Relations	Synsets are connected via **semantic relations** like hypernym (is-a), hyponym (has-a), meronym (part-of), antonym (opposite of), etc.

Wordnet Hierarchy



Wordnet Hierarchy

- One of the most important and useful aspects of WordNet is its **hierarchical structure**
- It helps NLP systems understand how words relate to each other in terms of **generality, specificity, and meaning**.

Entity (top-level synset)

├── **Physical Entity**

│ └── **Object**

│ └── **Animal**

│ └── **Mammal**

│ └── **Dog**

│ └── **Poodle**

in this hierarchy:

- “Poodle” is a hyponym of “Dog”
- “Dog” is a hypernym of “Poodle”
- “Mammal” is a more general category
- “Entity” is the root of all noun hierarchies

Wordnet Hierarchy

Why is WordNet Hierarchy Useful?

- **Word sense disambiguation**: Helps choose the correct meaning of a word in context
- **Semantic similarity**: You can measure how close two words are based on their path in the hierarchy
- **Text classification**: Use generalizations to improve model performance
- **Knowledge graphs**: Helps construct structured knowledge from unstructured text

CO-2,3; SO- 1;BL-	b.	Define WordNet. Describe the wordnet hierarchy.	[5]

How to Use WordNet in NLTK

Import and Download WordNet

```
import nltk
from nltk.corpus import wordnet as wn
nltk.download('wordnet')
```

Examples:

1. Get Synsets for a Word

```
print(wn.synsets('bank'))
```

```
[Synset('bank.n.01'), Synset('bank.n.02'), ..., Synset('bank.v.09')]
```

2. Get Definition and Examples of a Synset

```
syn = wn.synsets('bank')[0]
print("Definition:", syn.definition())
print("Examples:", syn.examples())
```

```
Definition: sloping land (especially the slope beside a body of water)
Examples: ['they pulled the canoe up on the bank']
```

3. Lemmas (Synonyms in a Synset)

```
print(syn.lemmas())
print([lemma.name() for lemma in syn.lemmas()])
```

```
[Lemma('bank.n.01.bank')]
['bank']
```

Summary

Term	Meaning
Synset	A set of words with the same meaning (sense)
Lemma	A word in its base form (e.g., "run" instead of "ran")
Hypernym	A general category (e.g., animal → dog)
Hyponym	A specific type (e.g., dog → poodle)
Meronym	Part of something (e.g., wheel → car)
Antonym	Opposite word (e.g., hot ↔ cold)

Wordnet 3.0 Statistics

Category	Count
Total Synsets	~117,000
• Noun synsets	~82,000
• Verb synsets	~13,500
• Adjective synsets	~18,000
• Adverb synsets	~4,500
Total Word Forms (Lemmas)	~155,000
• Unique word forms	~117,000
• Words per synset (avg.)	~1.03 to 1.1
Relations (Approximate)	
• Hypernym/Hyponym pairs	~75,000
• Meronym/Holonym pairs	~20,000
• Antonym pairs	~3,600
• Derivational links	~15,000
• Attribute links	~2,700
Glosses (definitions)	One per synset
Example sentences	Many synsets include usage examples