Natural Language Processing

Riccardo Caprile

October 2022

1 Terminology

1.1 Morpheme

A morpheme is the smallest grammatical unit in a language. In other words, it is the smallest meaningful unit of a language. The linguistics field of study dedicated to morphemes is called **morphology**. A morpheme is not identical to a word, and the principal difference between the two is that a morpheme may or may not stand alone , whereas a word , by definition , is freestanding.

Every morpheme can be classified as either **free** or **bound**.

Free morphemes can function independently as words (dog). Bound morphemes appear only as parts of words, always in conjunction with a root and sometimes with other bound morphemes (unbreakable).

Bound morphemes can be further classified as **derivational** or **inflectional**.

Derivational morphemes , when combined with a root , change either meaning or part of speech of the affected word. For example, in the word happiness, the addition of the bound morpheme -ness to the root happy changes the word from an adjective to a noun.

Inflectional morphemes modify a verb's tense , aspect , mood , person , or number , or noun's , without affecting the word's meaning or class. Adding -s to the root to form dogs.

1.2 Word

A word is the smallest element that can be uttered in isolation with objective or practical meaning. This contrast deeply with a morpheme , which is the smallest unit of meaning but will not necessarily stand on its own.

1.3 Corpus

A text corpus is a large and structured set of texts. They are used to do statistical analysis and hypothesis testing , checking occurrences or validating rules within a specific language territory

1.4 Genre

A style or category of art,music,or literature.

1.5 Genre of a corpus

Usually defined on the basis of text-external features,such as medium,function and format.

1.6 Collocation

A collocation is a sequence of words or terms that co-occur more often than would be expected by chance. ("Pay attention")

1.7 Compound

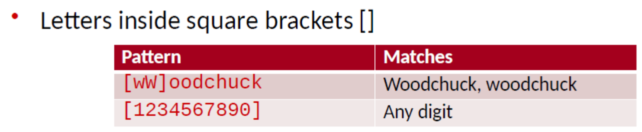
A compound is a word composed of more than one free morpheme. (Blackboard , adjective and noun)

1.8 Noun-phrase

A noun phrase or nominal phrase is a phrase which has a noun as its head word , or which performs the same grammatical function as such a phrase.(”The election-year politics are annoying for many people”).

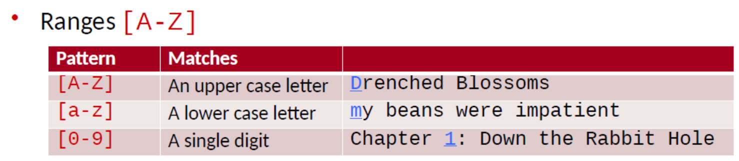
2 RE and Word Syntax

2.1 Regular Expression : Disjunctions



- Letters inside square brackets []

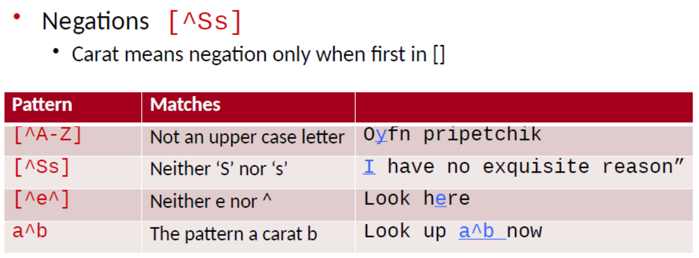
Pattern	Matches
`[wW]oodchuck`	Woodchuck, woodchuck
`[1234567890]`	Any digit



- 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`

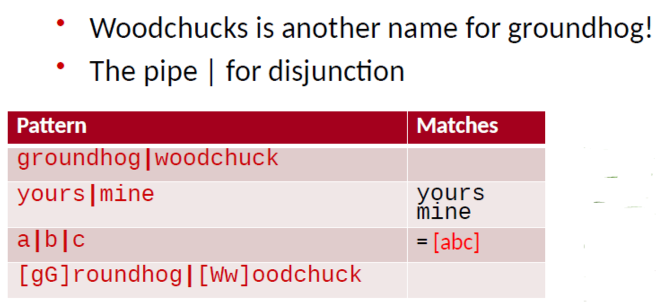
2.2 Regular Expressions : Negation in Disjunction



- Negations `[^Ss]`
  - Carat means negation only when first in []

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

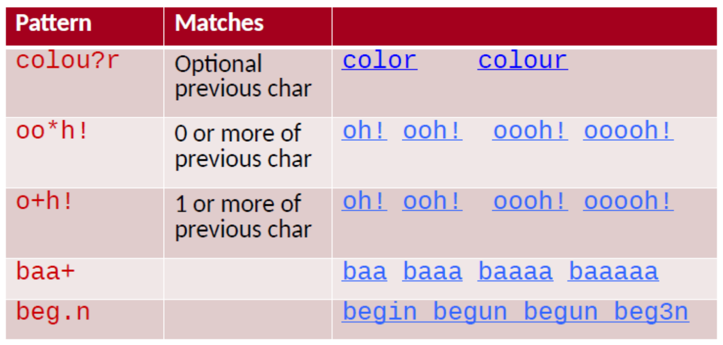
2.3 Regular Expressions : More Disjunction



- Woodchucks is another name for groundhog!
- The pipe | for disjunction

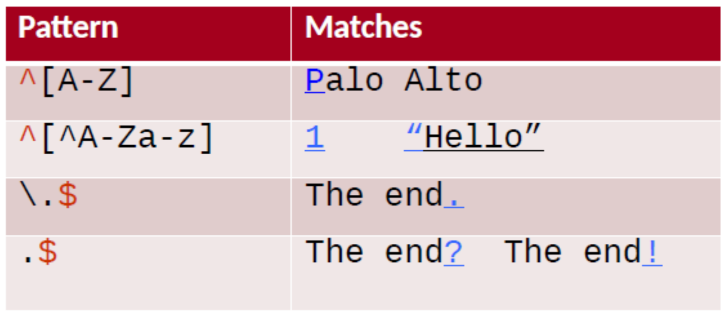
Pattern	Matches
groundhog|woodchuck	
yours|mine	yours mine
a|b|c	= [abc]
[gG]roundhog|[Ww]oodchuck	

2.4 Regular Expressions : Kleene



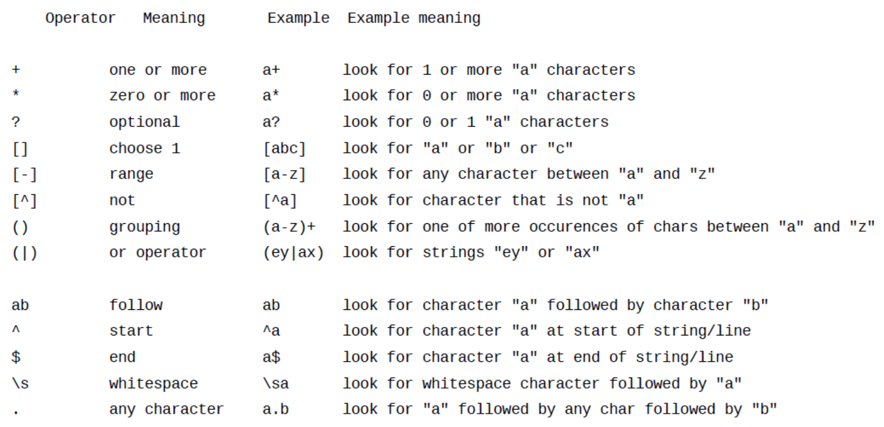
Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n		begin begun begun beg3n

2.5 Regular Expressions : Anchors



Pattern	Matches
^[A-Z]	Palo Alto
^[^A-Za-z]	1 “Hello”
\.$	The end.
.$	The end? The end!

2.6 Summary on Regular Expressions



Operator	Meaning	Example	Example meaning
+	one or more	a+	look for 1 or more "a" characters
*	zero or more	a*	look for 0 or more "a" characters
?	optional	a?	look for 0 or 1 "a" characters
[]	choose 1	[abc]	look for "a" or "b" or "c"
[-]	range	[a-z]	look for any character between "a" and "z"
[^]	not	[^a]	look for character that is not "a"
()	grouping	(a-z)+	look for one of more occurences of chars between "a" and "z"
(|)	or operator	(ey|ax)	look for strings "ey" or "ax"
ab	follow	ab	look for character "a" followed by character "b"
^	start	^a	look for character "a" at start of string/line
$	end	a$	look for character "a" at end of string/line
\s	whitespace	\sa	look for whitespace character followed by "a"
.	any character	a.b	look for "a" followed by any char followed by "b"

3 Syntax

3.0.1 Word Segmentation

Word segmentation is the problem of dividing a string of written language into its component words.(Multi-agent becomes Multi agent).

3.1 Stop Words

Stop words are words which are filtered out before or after processing of natural language data (text). Stop words usually refers to the most common words in a language processing tools , and indeed not all tools use such a list.

The general strategy for determining a stop list is to sort the terms by collection frequency , and then to take the most frequent terms , often hand-filtered for their semantic content relative to the domain of the documents being indexed, as a stop list , the members which are then discarded during indexing.

How can we discriminate between words which are frequent in a document **d** because they are related with the document's topic , and words which are frequent in d because they are frequent in a language?

Our simple corpus C1 consists of Doc1 (length 54) . It has 6 ”the” , 6 ”cat” and 0 car(s).

Corpus C2 consists of Doc1 and Doc2. Doc2 (length 42). It has 4 ”the”, 0 ”cat” , car(s).

So 10 ”the” , 6 ”cat” , 4 car(s) in C2.

3.1.1 Term Frequency

Term frequency tf(t,d) , is the raw count of a term t in a document d , the number of times that term t occurs in document d.

tf(t,d) = number of t ind d / length(d)

3.1.2 Inverse document frequency

Is the logarithmically scaled inverse fraction of the documents in the corpus C that contain the term t. Obtained by dividing the total number of documents in C by the number of documents containing t , and then taking the logarithm of that quotient.

$idf(t, C) = log(|C|/(|documentsinCwhichcontaint|)$

$|C|$ is the dimension of the corpus.

$tf{-}idf(''the'', Doc1, C2) = tf(''the'', Doc1) * idf(''the'', C2) = 6/54 * log(2/2) = 0$

The does not characterized document 1

$tf{-}idf(''cat'', Doc1, C2) = tf(''the'', Doc1) * idf(''the'', C2) = 6/54 * log(2/1) = 0.1 * 0.3 = 0.03$

cat characterized the document because tf-idf is greater than 0.

3.2 Stemming and Lemmatization

Documents are going to use different forms of words , such organize , organizes and organizing.

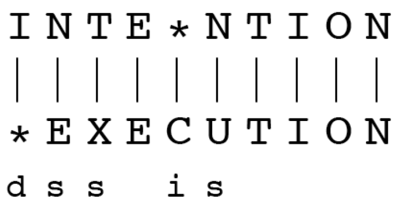
The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form (am,are,is - be)

Stemming and lemmatization differ in their flavor. **Stemming** usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.

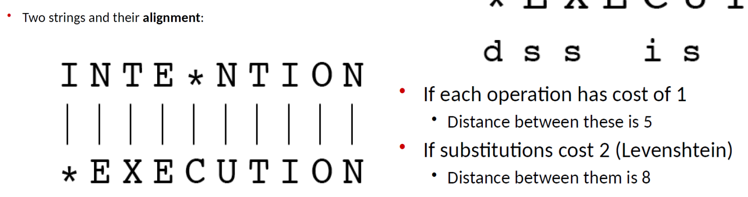
Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words , normally aiming to remove inflectional endings only and to return the base or dictionary form of a word , which is known as the lemma

3.3 Minimum Edit Distance

The minimum edit distance between two strings. We have as operations : Insertion , Deletion , Substitution



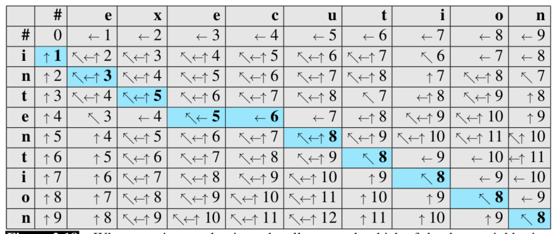
```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```



- Two strings and their **alignment**:

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

- If each operation has cost of 1
  - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
  - Distance between them is 8



	#	e	x	e	c	u	t	i	o	n
#	0	← 1	← 2	← 3	← 4	← 5	← 6	← 7	← 8	← 9
i	↑ **1**	↖←↑ 2	↖←↑ 3	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖ 6	← 7	← 8
n	↑ 2	↖←↑ **3**	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↑ 7	↖←↑ 8	↖ 7
t	↑ 3	↖←↑ 4	↖←↑ **5**	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖ 7	←↑ 8	↖←↑ 9	↑ 8
e	↑ 4	↖ 3	← 4	↖← **5**	← **6**	← 7	←↑ 8	↖←↑ 9	↖←↑ 10	↑ 9
n	↑ 5	↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ **8**	↖←↑ 9	↖←↑ 10	↖←↑ 11	↖↑ 10
t	↑ 6	↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖←↑ 9	↖ **8**	← 9	← 10	←↑ 11
i	↑ 7	↑ 6	↖←↑ 7	↖←↑ 8	↖←↑ 9	↖←↑ 10	↑ 9	↖ **8**	← 9	← 10
o	↑ 8	↑ 7	↖←↑ 8	↖←↑ 9	↖←↑ 10	↖←↑ 11	↑ 10	↑ 9	↖ **8**	← 9
n	↑ 9	↑ 8	↖←↑ 9	↖←↑ 10	↖←↑ 11	↖←↑ 12	↑ 11	↑ 10	↑ 9	↖ **8**

Figure 2.18 When entering a value in each cell, we mark which of the three neighboring cells we came from with up to three arrows. After the table is full we compute an **alignment** (minimum edit path) by using a **backtrace**, starting at the **8** in the lower-right corner and following the arrows back. The sequence of bold cells represents one possible minimum cost alignment between the two strings. Diagram design after Gusfield (1997).

If the letters are the same for row and column , i'll add 0 to the value in the diagonal cell , if they are different i'll add 2.

4 Syntax (Sentence Level)

4.1 Sentence Breaking

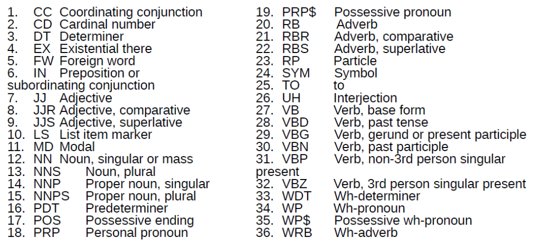
Sentence boundary disambiguation (SBD) , also known as sentence breaking or sentence boundary detection , **is the problem in natural language processing of deciding where sentences begin and end**. Often , natural language processing tools require their input to be divided into sentences for a number of reasons; however, sentence boundary identification is challenging because punctuation marks are often ambiguous. For example , a period may denote an abbreviation , decimal point , or an email address.

The standard approach to locate the end of a sentence:

- If it's a period , it ends a sentence.
- If the preceding token is in the hand-compiled list of abbreviations , then it does not end a sentence
- If the next token is capitalized , then it

4.2 Part of Speech Tagging

Part of Speech Tagging (POS), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text as corresponding to a particular part speech , based on both its definition and its context, its relationship with adjacent and related words in a phrase, sentence , or paragraph.



1.	CC	Coordinating conjunction	19.	PRP$	Possessive pronoun
2.	CD	Cardinal number	20.	RB	Adverb
3.	DT	Determiner	21.	RBR	Adverb, comparative
4.	EX	Existential there	22.	RBS	Adverb, superlative
5.	FW	Foreign word	23.	RP	Particle
6.	IN	Preposition or subordinating conjunction	24.	SYM	Symbol
7.	JJ	Adjective	25.	TO	to
8.	JJR	Adjective, comparative	26.	UH	Interjection
9.	JJS	Adjective, superlative	27.	VB	Verb, base form
10.	LS	List item marker	28.	VBD	Verb, past tense
11.	MD	Modal	29.	VBG	Verb, gerund or present participle
12.	NN	Noun, singular or mass	30.	VBN	Verb, past participle
13.	NNS	Noun, plural	31.	VBP	Verb, non-3rd person singular present
14.	NNP	Proper noun, singular	32.	VBZ	Verb, 3rd person singular present
15.	NNPS	Proper noun, plural	33.	WDT	Wh-determiner
16.	PDT	Predeterminer	34.	WP	Wh-pronoun
17.	POS	Possessive ending	35.	WP$	Possessive wh-pronoun
18.	PRP	Personal pronoun	36.	WRB	Wh-adverb

4.3 Parsing

Within computation linguistics , the term parsing is used to refer to the formal analysis by a computer of a sentence or other string of words into its constituents , resulting in a parse tree showing their syntactic relation to each other, which may also contain semantic and other information. Parsing is one of the possible ways to address the PoS problem.

Establishment of the Chomsky - Schutzenberger hierarchy , a classification of normal languages in terms of their generative power.

Recursively enumerable (Context sensitive (context free (regular))))

4.3.1 Grammar

Grammar specifies the compositional structure of complex messages. A **formal language** is a set of strings of **terminal symbols**.

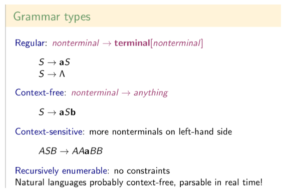
Each string in the language can be analyzed/generated by the grammar.

The grammar is a set of rewrite rules , for example :

S →NP VP

Article →The | a | an ...

Here S is the sentence symbol , NP and VP are nonterminals.



Grammar types

Regular: *nonterminal* → **terminal**[*nonterminal*]

$S \rightarrow \mathbf{a}S$
$S \rightarrow \Lambda$

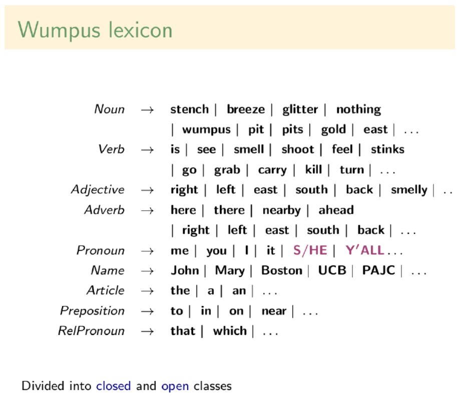
Context-free: *nonterminal* → *anything*

$S \rightarrow \mathbf{a}S\mathbf{b}$

Context-sensitive: more nonterminals on left-hand side

$ASB \rightarrow AA\mathbf{a}BB$

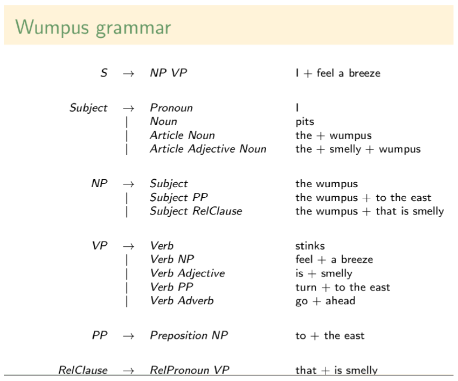
Recursively enumerable: no constraints
Natural languages probably context-free, parsable in real time!



Wumpus lexicon

Noun	→	**stench** | **breeze** | **glitter** | **nothing**
		| **wumpus** | **pit** | **pits** | **gold** | **east** | ...
Verb	→	**is** | **see** | **smell** | **shoot** | **feel** | **stinks**
		| **go** | **grab** | **carry** | **kill** | **turn** | ...
Adjective	→	**right** | **left** | **east** | **south** | **back** | **smelly** | ..
Adverb	→	**here** | **there** | **nearby** | **ahead**
		| **right** | **left** | **east** | **south** | **back** | ...
Pronoun	→	**me** | **you** | **I** | **it** | **S/HE** | **Y'ALL** ...
Name	→	**John** | **Mary** | **Boston** | **UCB** | **PAJC** | ...
Article	→	**the** | **a** | **an** | ...
Preposition	→	**to** | **in** | **on** | **near** | ...
RelPronoun	→	**that** | **which** | ...

Divided into closed and open classes



Wumpus grammar

S	→	*NP VP*	I + feel a breeze
Subject	→	*Pronoun*	I
	|	*Noun*	pits
	|	*Article Noun*	the + wumpus
	|	*Article Adjective Noun*	the + smelly + wumpus
NP	→	*Subject*	the wumpus
	|	*Subject PP*	the wumpus + to the east
	|	*Subject RelClause*	the wumpus + that is smelly
VP	→	*Verb*	stinks
	|	*Verb NP*	feel + a breeze
	|	*Verb Adjective*	is + smelly
	|	*Verb PP*	turn + to the east
	|	*Verb Adverb*	go + ahead
PP	→	*Preposition NP*	to + the east
RelClause	→	*RelPronoun VP*	that + is smelly

5 Hidden Markov Model

Used to model processes that move between states. States are hidden , but we see state outputs.

Our goal : find the sequence of states.

5.1 Example

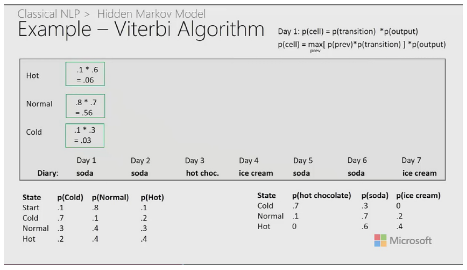
Interested in the weather in a small town , week of Oct 5 , 1900.

Weather records not kept , but we have diary of a teenager and his once a day snacks : hot chocolate , soda , or ice cream.

For our purposes , weather is : cold , normal or hot.

If we are given :

- Probabilities of a given snack for a given weather day , p(soda| hot)
- Weather day transition probabilities , p(hot| cold)
- Diary : soda , soda , hot chocolate , ice cream , soda , soda , ice cream



Classical NLP > Hidden Markov Model

Example – Viterbi Algorithm

Day 1: p(cell) = p(transition) *p(output)

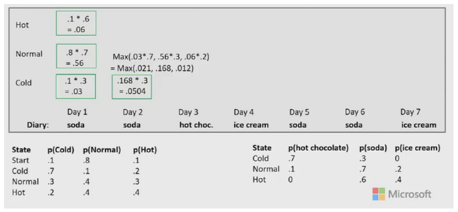
p(cell) = $\max_{prev}$[p(prev)*p(transition)] *p(output)

Hot	.1 * .6 = .06						
Normal	.8 * .7 = .56						
Cold	.1 * .3 = .03						
	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Diary:	**soda**	**soda**	**hot choc.**	**ice cream**	**soda**	**soda**	**ice cream**

State	**p(Cold)**	**p(Normal)**	**p(Hot)**
Start	.1	.8	.1
Cold	.7	.1	.2
Normal	.3	.4	.3
Hot	.2	.4	.4

State	**p(hot chocolate)**	**p(soda)**	**p(ice cream)**
Cold	.7	.3	0
Normal	.1	.7	.2
Hot	0	.6	.4

Microsoft

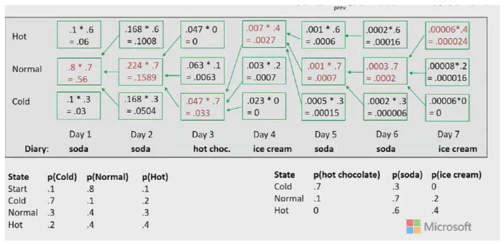


Hot	.1 * .6 = .06						
Normal	.8 * .7 = .56	Max(.03*.7, .56*.3, .06*.2) = Max(.021, .168, .012)					
Cold	.1 * .3 = .03	.168 * .3 = .0504					
	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Diary:	**soda**	**soda**	**hot choc.**	**ice cream**	**soda**	**soda**	**ice cream**

State	**p(Cold)**	**p(Normal)**	**p(Hot)**
Start	.1	.8	.1
Cold	.7	.1	.2
Normal	.3	.4	.3
Hot	.2	.4	.4

State	**p(hot chocolate)**	**p(soda)**	**p(ice cream)**
Cold	.7	.3	0
Normal	.1	.7	.2
Hot	0	.6	.4

Microsoft



State	p(Cold)	p(Normal)	p(Hot)
Start	.1	.8	.1
Cold	.7	.1	.2
Normal	.3	.4	.3
Hot	.2	.4	.4

State	p(hot chocolate)	p(soda)	p(ice cream)
Cold	.7	.3	0
Normal	.1	.7	.2
Hot	0	.6	.4

5.2 Maximum Entropy Markov Models (MEMM)

While an HMM can achieve very high accuracy in POS tagging , it requires a number of architectural innovations to deal with unknown words , suffixes , and so on.

MEMM integrate the ideas of Hidden Markov Models and logistic regression , a probabilistic classifier that makes use of supervised machine.

6 Automatic Text Summarization

Text summarization is commonly used by several websites and applications to create news feed and article summaries. It has become very essential for us due to our busy schedules. We prefer short summaries with all the important points over reading a whole report and summarizing it ourselves.

6.1 What is summarization ?

Summarization is a technique to shorten long texts such that the summary has all the important points of the actual document.

There are mainly four types of summaries :

1. Single Document Summary : Summary of a Single Document
2. Multi-Document Summary : Summary from multiple documents
3. Query Focused Summary : Summary of a Specific query
4. Informative Summary : It includes a summary of the full information

6.2 Approaches to Automatic Summarization

There are mainly two types of summarization :

- Extraction-based Summarization : The extractive approach involves picking up the most important phrases and lines from the documents. It then combines all the important to create the summary. So , in this case , every line and word of the count summary actually belongs to the original document which is summarized.

- Abstraction-based Summarization : The abstractive approach involves summarization based on deep learning. So , it uses new phrases and terms, different from the actual document , keeping the points the same , just like how we actually summarize.

6.3 Evaluation Methods

There are two types of evaluations :

- Human evaluation : Scores are assigned by human experts based on how well the summary covers the points , answer the queries, and other factors like grammatically and non-redundancy.
- Automatic evaluation : ROUGE

6.3.1 Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

It is the method that determines the quality of the summary by comparing it to other summaries made by humans as a reference. To evaluate the model , there are a number of references created by humans and generated candidate summary by machine. The intuition behind this is if a model creates a good summary, then it must have common overlapping portions with the human references.

6.3.2 Rouge-n

It is measure on the comparison between the machine-generated output and the reference output based on n-grams.

An **n-gram** is a sequence of n words.

Original sentence : ”The quick brown for jumps over”.

Unigrams : [”the”,”quick”,”brown”,”fox”,”jumps”,”over”]

6.3.3 Rouge-L

It states that the longer the longest common subsequence in two texts , the similar they are.

So , it is more flexible than n-gram.

It assign scores based on how long can be a sequence , which is common to the machine-generated candidate and the human reference.

6.4 Extractive Summarisations

The Extractive Summarisers first create an intermediate representation that has the main task of highlighting or taking out the most important information of the text to be summarized based on the representations.

6.5 Topic Representation

It focuses on representing the **topics** represented in the texts.

There are several kinds of approaches to get this representation.

6.6 Frequency Driven Approaches

In this approach , we assign **weights** to the words.

If the word is related to the topic we assign 1 or else 0.

Word Probability : It simply uses the frequency of words as indicator of the importance of the word. The probability of a word w is given by the frequency of occurrences of the word, f(W) , divided by all words in the input which has a total of N words.

$P(w) = f(w)/N$

TFIDF : This method is devised as an advancement to the word probability method. TFIDF is a method that assigns low weights to the words that occur very frequently in most of the documents under the intuitions that they are stopwords. Otherwise , due to the term frequency if a word appears in a document uniquely with a high frequency it is given high weight.

6.7 Topic Word Approaches

This method calculates the word frequencies and uses a frequency threshold to find the word that can potentially describe a topic.

It classifies the importance of a sentence as the function of the number of topic words it contains.

6.8 Indicator Representation

This type of representation depends on the features of the sentences and rank them on the basis of such features. So , the importance of the sentence is not dependent on the words it contains as we have seen in the Topic representations but directly on the sentence features.

There are two methods for this type of representation.

6.9 Graph-Based Methods

This is based on the **Page Rank Algorithm**

It represents text documents as connected graphs.

The sentences are represented as the nodes of the graphs and edges between the nodes are a measure of similarity between the two sentences.

6.10 Machine Learning Methods

The machine learning methods approach the summarisation problem as a classification problem.

The models try to classify sentences based on their features into , summary or non summary sentences.

For training the models , we have a training set of documents and their corresponding human reference extractive summaries.

6.11 Scoring and Sentences Selection

Now , once we get the intermediate representations , we move to assign some scores to each sentence to specify their importance.

For **topic representations** , a score to a sentence depends one the topic words it contains. For an **indicator representation** , the score depends on the features of the sentences. Finally, the sentences having top scores , are picked and used to generate a summary.

6.12 Text Rank Algorithm

This algorithm primarily tries to find the importance of a vertex in a given graph.

How the algorithm works?

Each sentence is represented as a vertex. An edge joining two vertices denotes that the two sentences are similar. If the similarity of any two sentences is greater than a particular threshold , the nodes representing the sentences are joined by an edge

When two vertices are joined , it portrays that , one vertex is casting a vote to the other one.

More the number of votes to a particular node , more important is that node and apparently the sentence represented.

Now , the votes are also kind of weighted , each vote is not of the same weight or importance.

The importance of the vote also depends on the importance of the node or sentence casting the vote ì, higher the importance of the node casting the vote higher is the importance of the vote.

This is the same idea behind the google page rank algorithm , and how it decides and ranks webpages.

If we have a paragraph , we will decompose it into a set of sentences.

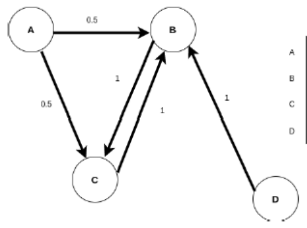
We represent each sentence as a vertex v_i so , we obtain a set of vertices V.

As discussed, an edge joins a vertex with another vertex of the same set , so an edge E can be represented as a subset of (V x V).

$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

$S(v_i)$ is the score of the subject node under consideration. $S(v_j)$ represents all the nodes that have outgoing edges to v_i.

The score of v_j is divided by the out-degree of v_j, which is the consideration of the probability that the user will choose that particular webpage.



$$\begin{array}{c|cccc|} & A & B & C & D \\ A & 0 & 0.5 & 0.5 & 0 \\ B & 0 & 0 & 1 & 0 \\ C & 0 & 1 & 0 & 0 \\ D & 0 & 1 & 0 & 0 \end{array}$$

The factor d is called the damping factor. In the original rank algorithm , factor d incorporates randomness. (1 - d) denotes that the user will move to a random webpage, not doing to the connected ones.

First we assign random scores to all the vertices , say [0.8,0.9,0.9,0.9].

Then , probability scores are assigned to the edges.

It can be observed that the values of adjacent matrices are the probability values, 1/outdegree of that node or vertex.

The whole equation becomes:

$$\begin{vmatrix} S(a) \\ S(b) \\ S(c) \\ S(d) \end{vmatrix} = (1-0.85) + 0.85 \left\langle \begin{array}{c|cccc|} & A & B & C & D \\ A & 0 & 0.5 & 0.5 & 0 \\ B & 0 & 0 & 1 & 0 \\ C & 0 & 1 & 0 & 0 \\ D & 0 & 1 & 0 & 0 \end{array} \cdot \begin{vmatrix} 0.8 \\ 0.9 \\ 0.9 \\ 0.9 \end{vmatrix} \right\rangle$$

6.13 Text Rank Algorithm implementations

The implementation of text rank consists of two different natural language processes :

- A keyword extraction task , which selects keywords and phrases. The natural language is tokenised and parts of speech are tagged, and single words are added to the word graph as nodes. If two words are similar , then corresponding nodes are connected using an edge. The similarity is measured using the co-occurences of words. If two words occur in a window of N words, N varying from 2 to 10 , the two words are considered similar. The words with the maximum number of important incident edges are selected as the most important keywords

- A sentence extraction task, this identifies the most important sentences. Differently from the keyword extraction , where the nodes represented keywords , here they represent entire sentences. For the formation of the graph for sentence ranking, the algorithm creates a vertex for each sentence in the text and adds to the graph. Since the sentences might be too large , co-occcurence measures can not be applied.

7 Semantics

Human language consists of two parts : a **lexicon**, essentially a catalogue of a language's words; and a **grammar** , a system of rules which allow for the combination of those words into meaningful sentences.

The unit of analysis in lexical semantics are lexical units which include not only words but also sub-words or sub-units such as affixes. Lexical unit make up the lexicon.

7.1 Lexical Relations

7.1.1 Synonymy

Similarity of meaning : two expressions are synonymous if the substitution of one for the other never changes the truth value of a sentence in which the substitution is made

7.1.2 Antonymy

Antonym of a word x is sometimes not-x , but not always. Rich and Poor are antonyms, but not rich does not imply not poor

7.1.3 Hyponymy

It is a semantic relation between word meanings. Maple is a hyponym of tree. Tree is a hypernym of maple.

7.1.4 Meronymy

A concept C1 is a meronym of a concept C2 in language L if native speakers of L accept sentences constructed from such frames as "A C2 has a C1

Car : engine , door , wheel ....

7.2 Word Sense Disambiguation WSD

It is an open problem of natural language processing and ontology. WSD is identifying which sense of word is used in a sentence, when the word has multiple meanings.

To give a hint all this works , let us consider the senses of the word "bass", and the two sentences: "I went fishing for some sea bass", "The bass line of the song is too weak".

To a human, it is obvious that the first sentence is using the word (fish) and the second sentence as instrument. Developing algorithms to replicate this human ability can often be a difficult task.

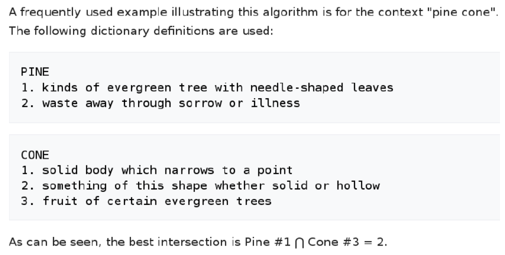
7.2.1 Lesk Algorithm

It is based on the assumption that words in a given ”neighborhood” (section of the text) will tend to share a common topic. A simplified version of the Lesk algorithm is to compare the dictionary definition of an ambiguous word with the terms contained in its neighborhood.

An implementation might look like this :

For every sense of the word being disambiguated one should count the amount of words that are in both neighborhood of that word and in dictionary definition of that sense.

The sense that is to be chosen is the sense which has the biggest number of this count.



A frequently used example illustrating this algorithm is for the context "pine cone".
The following dictionary definitions are used:

```
PINE
1. kinds of evergreen tree with needle-shaped leaves
2. waste away through sorrow or illness
```

```
CONE
1. solid body which narrows to a point
2. something of this shape whether solid or hollow
3. fruit of certain evergreen trees
```

As can be seen, the best intersection is Pine #1 $\cap$ Cone #3 = 2.

8 Ontologies

In the context of computer and information sciences , an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes , attributes (or properties) , and relationships. The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application.

“In the context of database systems, ontology can be viewed as a level of abstraction of data models, analogous to hierarchical and relational models, but intended for modeling knowledge about individuals, their attributes, and their relationships to other individuals. Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases. For this reason, ontologies are said to be at the “semantic” level, whereas database schema are models of data at the “logical” or “physical” level.”

8.1 RDF - RDF Model and Syntax

RDF is a W3C reccomendation and provides a general , flexible , method to decompose any knowledge into small pieces , called triples , with some rules about the semantics (meanings) of those pieces.

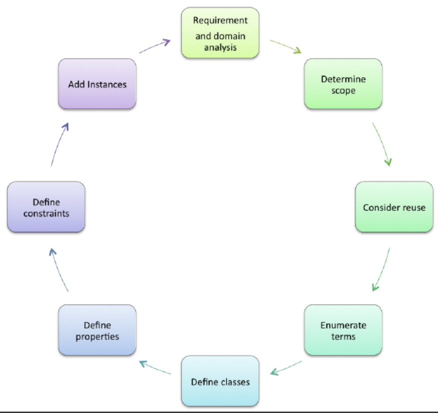
Basic Idea : triple (O,A,V) , an object O has an attribute A with value V.

RDF schemas are used to define the structure of the metadata that are used to describe WWW resources.

8.2 OWL (Ontology web Language)

OWL is an extension of RDF schema, in the sense that OWL would use the RDF meaning of classes and properties, and would add language primitives to support a richer expressiveness.

9 Ontology Design Process



9.1 Requirement and domain analysis

9.1.1 Application Requirements

Application requirements can be acquired by :

- Identifying any controlled vocabulary used in the application
- Identifying any hierarchical or taxonomic structures intrinsic in the domain that might be used for query expansion : Vegeterian pizza such as margherita , funghi ...
- Analysing structured queries and the knowledge they require
- Computational tractability

9.1.2 Domain Requirements

- Take into account heterogeneity , distribution and autonomy needs
- Open vs Closed World (does lack of information imply negative information?)
- Static vs Dynamic ontology processes
- Limited or incomplete knowledge

9.2 Determine Scope

Addresses straight forward questions such as :

- What is the ontology going to be used for
- How is the ontology ultimately going to be used by the software implementation?
- What do we want the ontology to be aware of, and what is the scope of the knowledge we want to have in the ontology?

9.3 Consider Reuse

We rarely have to start from scratch when defining an ontology , there is always an ontology available from a third party that provides at least a useful starting point for our own ontology

Standard vocabularies are available for most domains , many of which are overlapping.

Identify the set that is most relevant to the problem and application issue.

9.4 Enumerate Design Process

Write down in an unstructured list all the relevant terms that are expected to appear in the ontology (nouns form the basis for class names , verbs forms the basis for property names)

9.5 Define Classes

A class is a concept in the domain : Animal (cow,cat,fish) , A class of properties (father,mother).

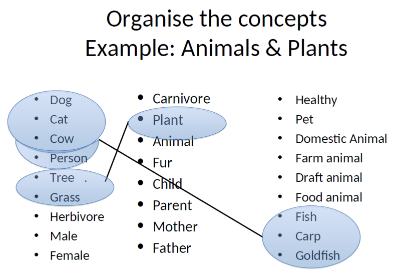
A class is a collection of elements with similar properties. A class contains necessary conditions for membership.

Instances of classes : a particular farm animal, a particular person

9.5.1 Extend the concepts

Take a group of things and ask what they have in common.

For example , Plant and Animal are Living Thing



Organise the concepts
Example: Animals & Plants

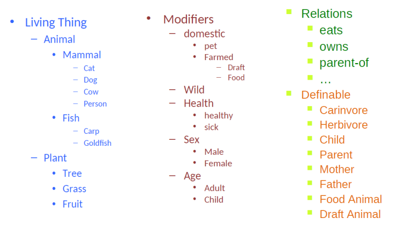
- Dog
- Cat
- Cow
- Person
- Tree .
- Grass
- Herbivore
- Male
- Female

- Carnivore
- Plant
- Animal
- Fur
- Child
- Parent
- Mother
- Father

- Healthy
- Pet
- Domestic Animal
- Farm animal
- Draft animal
- Food animal
- Fish
- Carp
- Goldfish

9.5.2 Choose some main axes

Add abstraction where needed , identify relations , identify definable things



- Living Thing
  - Animal
    - Mammal
      - Cat
      - Dog
      - Cow
      - Person
    - Fish
      - Carp
      - Goldfish
  - Plant
    - Tree
    - Grass
    - Fruit

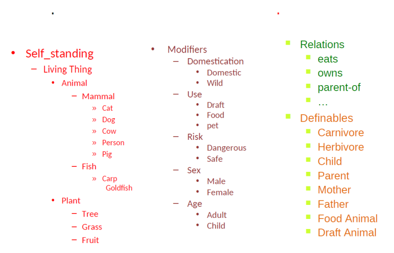
- Modifiers
  - domestic
    - pet
    - Farmed
      - Draft
      - Food
  - Wild
  - Health
    - healthy
    - sick
  - Sex
    - Male
    - Female
  - Age
    - Adult
    - Child

- Relations
  - eats
  - owns
  - parent-of
  - …
- Definable
  - Carinvore
  - Herbivore
  - Child
  - Parent
  - Mother
  - Father
  - Food Animal
  - Draft Animal

9.5.3 Identify self-standing entities

Things that can exist on their own (People, animals , houses , actions , processes)

Modifiers : things that modify in other things (roughly adjectives and adverbs)



- Self_standing
  - Living Thing
    - Animal
      - Mammal
        - Cat
        - Dog
        - Cow
        - Person
        - Pig
      - Fish
        - Carp
          Goldfish
    - Plant
      - Tree
      - Grass
      - Fruit

- Modifiers
  - Domestication
    - Domestic
    - Wild
  - Use
    - Draft
    - Food
    - pet
  - Risk
    - Dangerous
    - Safe
  - Sex
    - Male
    - Female
  - Age
    - Adult
    - Child

- Relations
  - eats
  - owns
  - parent-of
  - …
- Definables
  - Carnivore
  - Herbivore
  - Child
  - Parent
  - Mother
  - Father
  - Food Animal
  - Draft Animal

9.5.4 Class inheritance

Classes are organized into subclass-superclass

Classes are "is-a" related if an instance of the subclass is an instance of the superclass. (Mammal is a sublcass of Animal)

9.6 Define Properties

Often interleaved with the previous step.

Properties describe the attributes of the members of a class.

The semantic of subClassOf demands that whenever A is a subclass of B , every proerty statement that holds for instances of B must also apply to instances of A.

Type of properties :

- Intrinsic properties : flavor and color of wine
- Extrinsic properties : name and price of wine
- Parts : ingredients in a dish
- Relations to other objects : producer of wine

They are represented by data and objcet properties : simple contain primitive valus , complex properties contain other objects

9.7 Define Constraints

9.7.1 Identify the domain and range constraints for properties

Animal eats Living thing (eats domain : Animal , range : Living thing)

9.8 Add Instances

Create an instance of a class : the class becomes a direct type of the instance and any superclass of the direct type is a type of the instance

Filling the ontologies with such instances is a separate step

9.9 N-ary relations

In OWL a property is a binary relations : instances of properties link two individuals.

However , sometimes the most intuitive way to represent certain concepts is to use relations to link an individual to more than just one individual or value (n-ary relations)

9.10 Applications of Ontologies

GeoScience , Medicine , Bioinformatics and organising complex and semi-structured information

10 Word2Vec

10.1 Word Embedding

It is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.. Loosely speaking , they are vector representations of a particular word.

10.1.1 Why do we need word embedding?

Consider the following sentences : "Have a good day" and "Have a great day".

They hardly have different meaning If we construct an exhaustive vocabulary , it would have : V = Have,a,good,day

Now , let us create a one-hot encoded vector for each of these words in V.

Length of our one-hot encoded vector would be equal to the size of V.

We would have a vector of zeros except for the element at the index representing the corresponding word in the vocabulary.

We would end up with the following encoding : Have = [1,0,0,0,0] ; a = [0,1,0,0,0] ; good = [0,0,1,0,0] ; ; great = [0,0,0,1,0] ; day = [0,0,0,0,1]

If we try to visualise these encodings , we can think of a 5 dimensional space, where each word occupies one of the dimensions and has nothing to do with the rest.

This means good and great are as different as day and have , which is not true.

Our objective is to have words with similar context occupy close spatial positions.

The basic idea of word embedding is words that occur in similar context tend to be closer to each other in vector space.

10.1.2 Idea!

Let use generate **distributed representations**.

We introduce some dependence of one word on the other words.

The words in context of this word would get a greater share of this dependence.

In one hot encoding representations , all the words are independent of each other , as mentioned earlier.

10.2 Word2Vec

It is a method to construct such an embedding. It is not a singular algorithm , rather , it is a family of model architectures and optimisations that can be used to learn word embeddings from large datasets. These models are shallow two layer neural networks having one input layer, one hidden layer and one output layer.

Word2vec uses two architectures : CBOW and Skip Gram

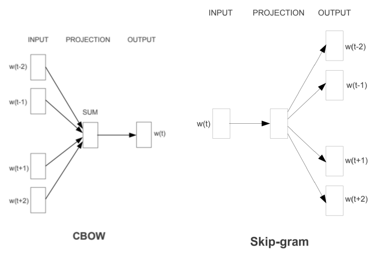
10.2.1 CBOW

CBOW mdoel predicts the current word given context words within specific window.

The input layer contains the context words and the output layer contains the current word.

The hidden layer contains the number of dimensions in which we want to represent current word present at the output layer.

Much simpler , much faster convergence , learn better syntactic relationship.



10.2.2 Skip Gram

Skip gram predicts the surrounding context words within specific window given current word.

The input layer contains the current word and the output layer contains the context words.

The hidden layer contains the number of dimensions in which we want to represent current word present at the input layer.

Learn better semantic relationships, less sensitive to overfit frequent words , more efficient in term of documents required to achieve good performances.

11 Applications

11.1 Machine Translation

Machine translation is a sub-field of computational linguistics that investigates the use of software to translate text or speech from one language to another.

11.2 Information Filtering and Retrieval

An **information filtering** system is a system that removes redundant or unwanted information from an information stream using (semi)automated or computerized methods prior to presentation to a human user. Its main goal is the management of the information overloaded and increment of the semantic signal-to-noise ratio. To do this the user's profile is compared to some reference characteristics. These characteristics may originate from the information item (the content based-approach) or the user's social environment (the collaborative filtering approach).

Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on full-text or other content based indexing. Information retrieval is the science of searching for information in a document, searching for documents themselves, and also searching for metadata that describe data, and for databases of texts,images or sound. Automated information retrieval systems are used to reduce what has been called information overloaded. Many universities and public libraries use IR systems to provide access to books, journals and other documents.

11.3 Question Answering

Question answering is computer science discipline within the fields of information retrieval and natural language processing , which is concerned with building systems that automatically answer questions posed by humans in a natural language. Closed-domain question answering deals with questions under a specific domain , and can be seen as an easier task because NLP systems can exploit domain-specific knowledge frequently formalized in ontologies. Open-domain question answering deals with questions about nearly anything, and can only rely on general ontologies and world knowledge. On the other hand , these systems usually have much more data available from which to extract the answer.

11.4 Automatic Summarization

Automatic summarization is the process of shortening a text document with software , in order to create a summary with the major points of the original document.

11.5 Sentiment Analysis

Sentiment analysis refers to the use of natural language processing, text analysis, computational linguistics , and biometrics to systemically identify , extract, quantify, and study affective states and subjective information

Sentiment analysis aims to determine the attitude of a speaker , writer , or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction or event. The attitude may be a judgement or evaluation, affective state , or the intended emotional communication.

12 Text Features

12.1 Style : n-grams , collocations

Co-occurence of the same n characters or words.

12.2 Style : POS(part of speech)

Playing a particular role (part) in a sentence (speech) is a feature of one word inside a sentence : no need of full documents or corpus. The number of words having a given POS (number of pronouns, number of adjectives) possibly normalized w.r.t the document's length , is a feature of a document.

12.3 Style : FW(Function Word)

In linguistic , function words are words that have little lexical meaning or have ambiguous meaning and express grammatical relationships among other words within a sentence, or specify the attitude or mood of the speaker.

Being a FW is a feature of the world itself : if the word is not ambiguous , there is not even need to have the sentence the word belongs to.

The number of FW , possibly normalized w.r.t the document's length, is a feature of a document.

12.4 Style : Readability measure

Readability is a measure of how easy a piece of text is to read. It can include elements of complexity,familiarity,legibility, and typography. Readability formulas usually look at factors like sentence length, syllable density and word familiarity as part of their calculations.

12.5 Content-based features

Frequency of words belonging to some given , knows semantic category. (TF).

12.6 Vocabulary and Idiosyncrasies

Unique words : number of unique words in the sample.

Capital words : number of words(length greater than 2) containing only capital letters

Spelling errors : number of words that are not found in the language dictionaries

Character flooding : sooooo saaad