01 Spacy Tokenization

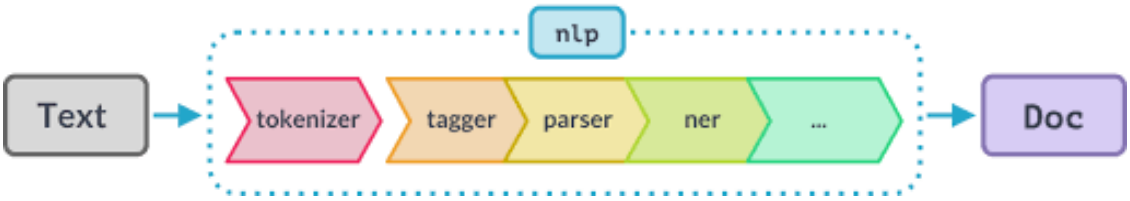
December 20, 2025

1 Tokenization

[1]:
```
import spacy
from prettytable import PrettyTable
```

[2]:
```
nlp = spacy.load('en_core_web_sm')
```

[3]:
```
text = "Hello Uditya Narayan Tiwari! let's learn NLP together i am 21"
```



[4]:
```
doc = nlp(text)
doc
```

[4]:
```
Hello Uditya Narayan Tiwari! let's learn NLP together i am 21
```

[5]:
```
doc.to_dict()
```

[5]:
```
{'text': "Hello Uditya Narayan Tiwari! let's learn NLP together i am 21",
 'array_head': (71, 81, 65, 67, 75, 79, 76, 77, 78, 452, 454, 73, 453, 74, 80),
 'array_body': array([[                   5,                    1,
1577730570815003155 1,
          5983625672228268878,  3252815442139690129,                    0,
           8206900633647566924,                    2,                    0,
                             0,                    0,  5983625672228268878,
                           456,                   91,                    1],
        [                    6,                    1,   621519759253969662,
         12044956042206692584, 15794550382381185553,                    2,
           7037928807040764755,                    3,                  380,
                             0,                    0,   621519759253969662,
         11292551915497242671,                   96, 18446744073709551615],
        [                    7,                    1, 17937943263439891957,
           1332317125561210957, 15794550382381185553,                    1,
           7037928807040764755,                    1,                  380,
                             0,                    0, 17937943263439891957,
         11292551915497242671,                   96, 18446744073709551615],
        [                    6,                    0,  7708786325424957310,
         17610643503028872449, 15794550382381185553, 18446744073709551613,
                           428,                    1,                  380,
                             0,                    0,  7708786325424957310,
         11292551915497242671,                   96, 18446744073709551615],
        [                    1,                    1, 17494803046312582752,
         17494803046312582752, 12646065887601541794, 18446744073709551612,
                           445,                    2,                    0,
                             0,                    0, 17494803046312582752,
           6739740606194143788,                   97, 18446744073709551615],
        [                    3,                    0,   278066919066513387,
            278066919066513387, 14200088355797579614,                    0,
           8206900633647566924,                    2,                    0,
                             0,                    0,   278066919066513387,
           4068996703163926224,                  100,                    1],
```

```
       [                   2,                    1, 16428057658620181782,
         4950757572332304006, 13656873538139661788,                    1,
                         429,                    2,                    0,
                           0,                    0,  4950757572332304006,
        12523944338091500347,                   95, 18446744073709551615],
       [                   5,                    1,  9664905639869093544,
         9664905639869093544, 14200088355797579614, 18446744073709551614,
                         408,                    2,                    0,
                           0,                    0,  9664905639869093544,
         4068996703163926224,                  100, 18446744073709551615],
       [                   3,                    1, 15832915187156881108,
        11273594034978133401, 15794550382381185553, 18446744073709551615,
                         416,                    3,                  383,
                           0,                    0, 15832915187156881108,
        11292551915497242671,                   96, 18446744073709551615],
       [                   8,                    1, 12060003407050460571,
        12060003407050460571,   164681854541413346, 18446744073709551614,
                         400,                    2,                    0,
                           0,                    0, 12060003407050460571,
                         456,                   86, 18446744073709551615],
       [                   1,                    1,  5097672513440128799,
         5097672513440128799, 13656873538139661788,                    1,
                         429,                    2,                    0,
                           0,                    0,  4690420944186131903,
         8572492957087256302,                   95, 18446744073709551615],
       [                   2,                    1,   959164148857638496,
          959164148857638496,  9188597074677201817, 18446744073709551612,
                         408,                    2,                    0,
                           0,                    0, 10382539506755952630,
         1447802835980306976,                   87, 18446744073709551615],
       [                   2,                    0,  4686009691886217934,
         4686009691886217934,  8427216679587749980, 18446744073709551615,
                         404,                    2,                    0,
                           0,                    0,  4686009691886217934,
         1599824077107694006,                   93, 18446744073709551615]],
      dtype=uint64),
'sentiment': 0.0,
'tensor': array([[ 0.05445875, -0.59416246,  0.818173  , …,  0.38977188,
        -0.40392596,  1.4487071 ],
       [-0.04225922, -1.1231196 ,  1.6962935 , …,  0.59242487,
         0.6001835 ,  0.16076504],
       [ 0.24459338, -1.140665  ,  0.28151155, …,  0.08815472,
         0.8826299 , -0.05909336],
       …,
       [-1.0578656 , -0.7674625 , -0.77184594, …,  0.47284043,
        -0.36860317,  0.25863093],
       [-0.13910657, -0.273593  , -0.9564745 , …,  0.04795459,
```

```
        -0.00504667,  2.2988346 ],
       [-0.23384394,  0.4510045 ,  0.6332393 , …,  0.61563087,
        -0.33255428, -0.26758942]], dtype=float32),
'cats': {},
'spans': b'\x90',
'strings': ['',
 'PERSON',
 'narayan',
 'ROOT',
 'PronType=Prs',
 '21',
 'npadvmod',
 'tiwari',
 'hello',
 'attr',
 'VB',
 'ORG',
 'Number=Sing',
 'dobj',
 'VBP',
 'Mood=Ind|Number=Sing|Person=1|Tense=Pres|VerbForm=Fin',
 'Narayan',
 'learn',
 '.',
 'Uditya',
 'NumType=Card',
 'NNP',
 'Tiwari',
 'I',
 'punct',
 'advmod',
 'NLP',
 '!',
 'let',
 'compound',
 'be',
 'CD',
 'VerbForm=Inf',
 'us',
 'nlp',
 'PunctType=Peri',
 'UH',
 'together',
 'i',
 'am',
 'uditya',
 'ccomp',
```

```
  'RB',
  'nsubj',
  'Case=Nom|Number=Sing|Person=1|PronType=Prs',
  'PRP'],
 'has_unknown_spaces': False}
```

[6]:
```
for token in doc:
        print(token.text, token.is_alpha, token.is_punct, token.like_num)
```

```
Hello True False False
Uditya True False False
Narayan True False False
Tiwari True False False
! False True False
let True False False
's False False False
learn True False False
NLP True False False
together True False False
i True False False
am True False False
21 False False True
```

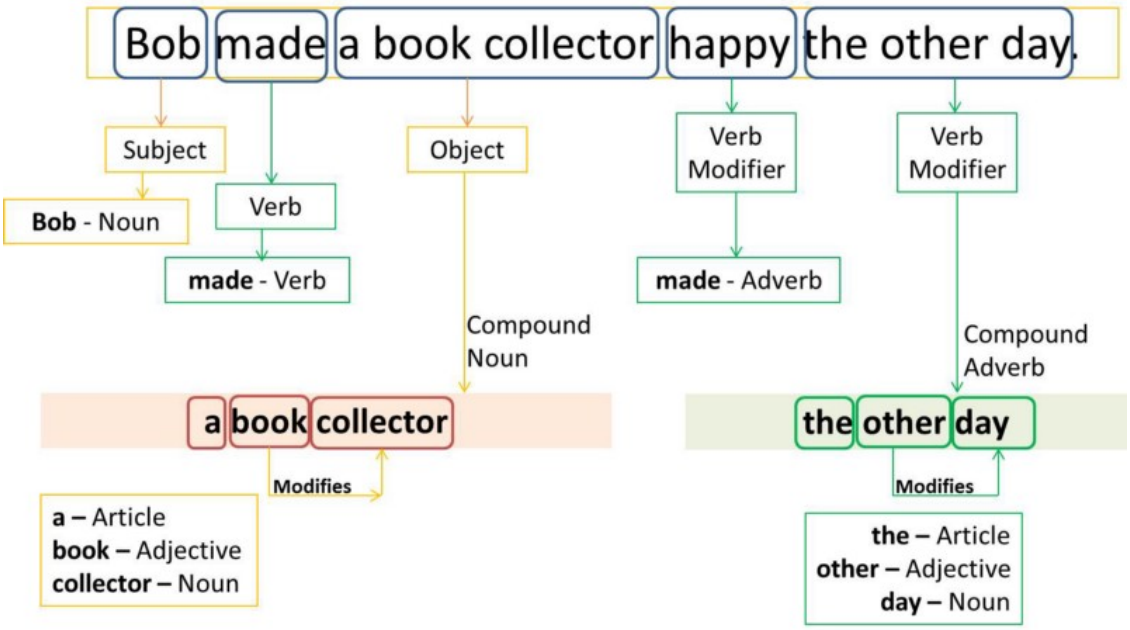
[7]:
```
table = PrettyTable()
table.field_names = ['token', 'is alpha', 'is punct', 'is number']
for token in doc:
        table.add_row([token.text, token.is_alpha, token.is_punct, token.
 ↪like_num])
```

[8]:
```
print(table)  # so using pretty table we can see the things proper in the␣
 ↪formatted way
```

```
+----------+----------+----------+-----------+
|  token   | is alpha | is punct | is number |
+----------+----------+----------+-----------+
|  Hello   |   True   |  False   |   False   |
|  Uditya  |   True   |  False   |   False   |
| Narayan  |   True   |  False   |   False   |
|  Tiwari  |   True   |  False   |   False   |
|    !     |  False   |   True   |   False   |
|   let    |   True   |  False   |   False   |
|    's    |  False   |  False   |   False   |
|  learn   |   True   |  False   |   False   |
|   NLP    |   True   |  False   |   False   |
| together |   True   |  False   |   False   |
|    i     |   True   |  False   |   False   |
|    am    |   True   |  False   |   False   |
|    21    |  False   |  False   |    True   |
```

```
+----------+----------+----------+-----------+
```

2 Part Of Speech(POS) Tagging



[9]:
```
doc
```

[9]:
```
Hello Uditya Narayan Tiwari! let's learn NLP together i am 21
```

[10]:
```
table = PrettyTable()
table.field_names = ['token', 'pos', 'details', 'explanation']

for token in doc:
        table.add_row([token.text, token.pos_, token.tag_, spacy.explain(token.
 ↪tag_)])

print(table)
```

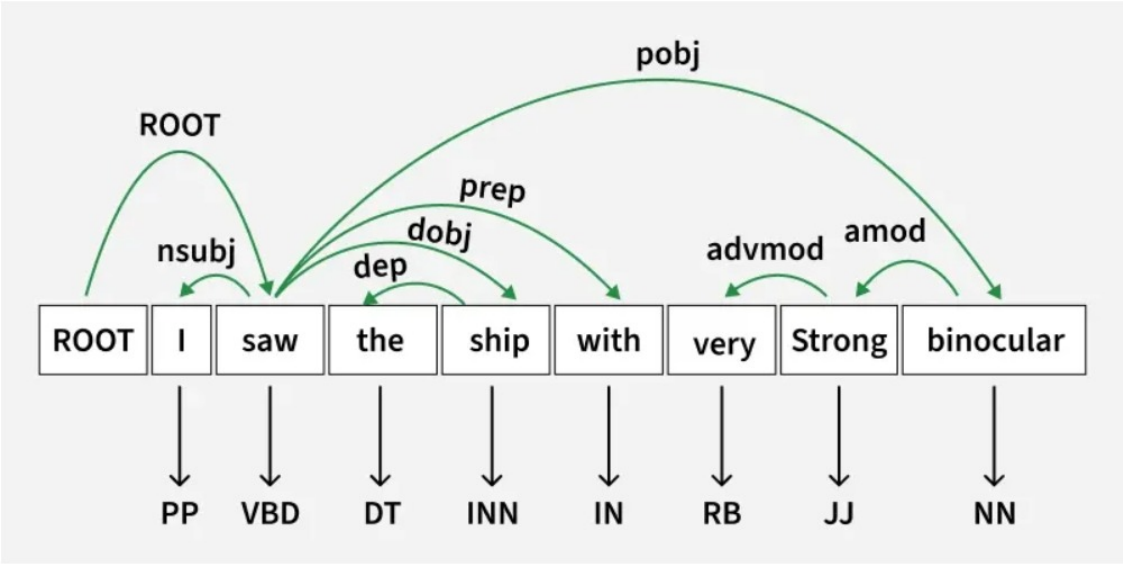
```
+----------+-------+---------+---------------------------------------+
|  token   |  pos  | details |              explanation              |
+----------+-------+---------+---------------------------------------+
|  Hello   |  INTJ |    UH   |              interjection             |
|  Uditya  | PROPN |   NNP   |         noun, proper singular         |
| Narayan  | PROPN |   NNP   |         noun, proper singular         |
|  Tiwari  | PROPN |   NNP   |         noun, proper singular         |
|    !     | PUNCT |    .    |   punctuation mark, sentence closer   |
|   let    |  VERB |    VB   |           verb, base form             |
|    's    |  PRON |   PRP   |          pronoun, personal            |
|  learn   |  VERB |    VB   |           verb, base form             |
```

NLP	PROPN	NNP	noun, proper singular
together	ADV	RB	adverb
i	PRON	PRP	pronoun, personal
am	AUX	VBP	verb, non-3rd person singular present
21	NUM	CD	cardinal number

02 Dependency Parsing

December 20, 2025

1 Dependency Parsing



[1]:
```
import spacy
from prettytable import PrettyTable
```

[2]:
```
nlp = spacy.load('en_core_web_sm')
```

[3]:
```
text = "Spot Intelligence is a NLP company that builds custom models"
```

[4]:
```
doc = nlp(text)
doc
```

[4]:
```
Spot Intelligence is a NLP company that builds custom models
```

[5]:
```
table = PrettyTable()
table.field_names = ['token', 'dep', 'head text', 'head pos', 'children']

for token in doc:
        children = [child.text for child in token.children]
```

```
        table.add_row([token.text, token.dep_, token.head.text, token.head.
 ↪pos_, children])

print(table)
```

```
+--------------+----------+--------------+----------+---------------------------
--+
|    token     |   dep    |  head text   | head pos |           children
|
+--------------+----------+--------------+----------+---------------------------
--+
|     Spot     | compound | Intelligence |  PROPN   |              []
|
| Intelligence |  nsubj   |      is      |   AUX    |           ['Spot']
|
|      is      |   ROOT   |      is      |   AUX    | ['Intelligence',
'company'] |
|      a       |   det    |   company    |   NOUN   |              []
|
|     NLP      | compound |   company    |   NOUN   |              []
|
|   company    |   attr   |      is      |   AUX    |    ['a', 'NLP', 'builds']
|
|     that     |  nsubj   |    builds    |   VERB   |              []
|
|    builds    |  relcl   |   company    |   NOUN   |      ['that', 'models']
|
|    custom    | compound |    models    |   NOUN   |              []
|
|    models    |   dobj   |    builds    |   VERB   |          ['custom']
|
+--------------+----------+--------------+----------+---------------------------
--+
```

[6]:
```
from spacy import displacy
displacy.render(doc, style = 'dep', jupyter=True, options={})
```

```
<IPython.core.display.HTML object>
```

[7]:
```
from spacy import displacy
displacy.render(doc, style = 'dep', jupyter=True, options={'distance':85,␣
 ↪'compact':True, 'bg':'#09a3d3', 'color': '#ffffff'})
```

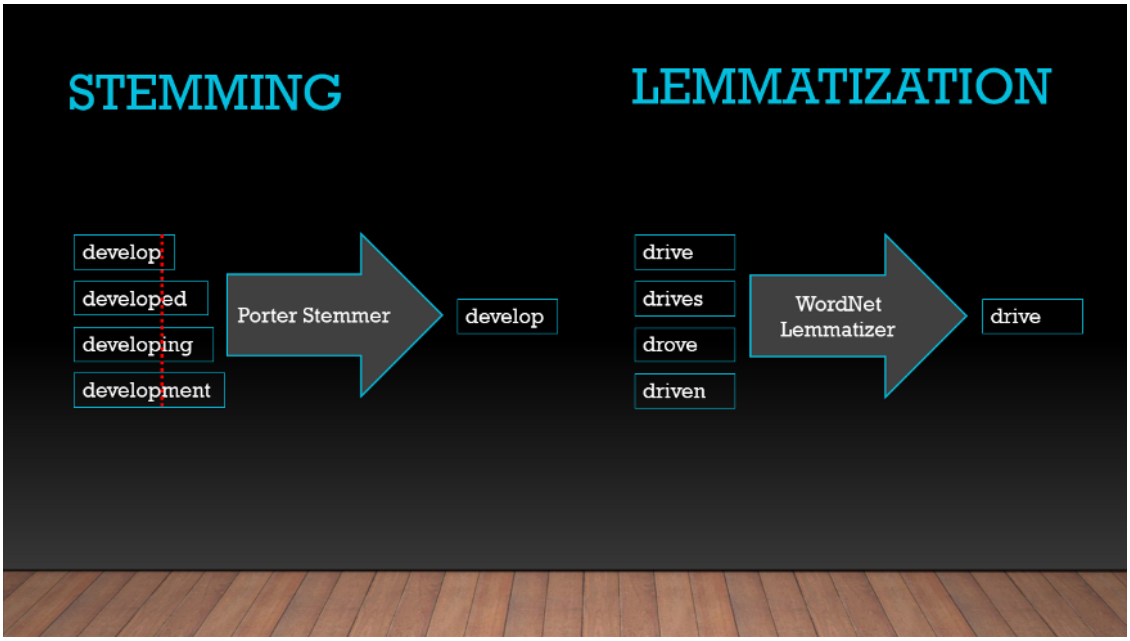
```
<IPython.core.display.HTML object>
```

03 Lemmatization

December 20, 2025

1 Lemmatization

- **Base Form Reduction**: Lemmatization reduces words to their meaningful base form(e.g., "running" to "run")
- **Enhances NLP Accuracy**: It normalizes words variations, improving the accuracy of NLP tasks by treating different word forms as the same.



[1]:
```
import spacy
from prettytable import PrettyTable
nlp = spacy.load('en_core_web_sm')
```

[2]:
```
text = 'run ran running'
doc = nlp(text)

table = PrettyTable(field_names=['token', 'lemma'])  # here lemma means root␣
 ↪words

for token in doc:
        table.add_row([token.text, token.lemma_])
```

```
print(table)
```

```
+---------+-------+
|  token  | lemma |
+---------+-------+
|   run   |  run  |
|   ran   |  run  |
| running |  run  |
+---------+-------+
```

[3]:
```
text = 'The scientists discover new species every year, Last year, they␣
 ↪discovered an ancient artifact. they are discovering new techniques with␣
 ↪their recent discovery'
doc = nlp(text)

table = PrettyTable(field_names=['token', 'lemma', 'pos'])  # here lemma means␣
 ↪root words

for token in doc:
        table.add_row([token.text, token.lemma_, token.pos_])

print(table)
```

```
+-------------+-----------+-------+
|    token    |   lemma   |  pos  |
+-------------+-----------+-------+
|     The     |    the    |  DET  |
|  scientists | scientist |  NOUN |
|   discover  |  discover |  VERB |
|     new     |    new    |  ADJ  |
|   species   |   specie  |  NOUN |
|    every    |   every   |  DET  |
|     year    |    year   |  NOUN |
|      ,      |     ,     | PUNCT |
|     Last    |    last   |  ADJ  |
|     year    |    year   |  NOUN |
|      ,      |     ,     | PUNCT |
|     they    |    they   |  PRON |
|  discovered |  discover |  VERB |
|      an     |     an    |  DET  |
|   ancient   |  ancient  |  ADJ  |
|   artifact  |  artifact |  NOUN |
|      .      |     .     | PUNCT |
|     they    |    they   |  PRON |
|     are     |     be    |  AUX  |
| discovering |  discover |  VERB |
|     new     |    new    |  ADJ  |
```

techniques	technique	NOUN
with	with	ADP
their	their	PRON
recent	recent	ADJ
discovery	discovery	NOUN

04 Named Entity Recognition

December 20, 2025

1 Named Entity Recognition(NER)

- **Entity Classification**: NER identifies and categorizes entities like names, locations, and dates in text, helping to extract key information.
- **Broad Application**: It enhances tasks like information extraction , question answering, and summarization by highlighting important entities in the text.

[1]:
```
import spacy
from prettytable import PrettyTable

nlp = spacy.load('en_core_web_sm')
```

[2]:
```
nlp.get_pipe('ner').labels # pre-trained model --> ner
```

[2]:
```
('CARDINAL',
 'DATE',
 'EVENT',
 'FAC',
```

```
 'GPE',
 'LANGUAGE',
 'LAW',
 'LOC',
 'MONEY',
 'NORP',
 'ORDINAL',
 'ORG',
 'PERCENT',
 'PERSON',
 'PRODUCT',
 'QUANTITY',
 'TIME',
 'WORK_OF_ART')
```

[3]:
```
text = """
On December 20, 2025, Satya Nadella announced that Microsoft would invest $5␣
 ↪billion to build a new research hub in London. This facility, located near␣
 ↪Kings Cross, aims to rival the innovation seen at Google in Mountain View,␣
 ↪California.
"""
```

[4]:
```
from spacy import displacy
doc = nlp(text = text)
displacy.render(doc, style = 'ent', jupyter = True)
```

```
<IPython.core.display.HTML object>
```

[5]:
```
table = PrettyTable()
table.field_names=['token', 'label', 'explanation', 'Star Char', 'End Char']
for ent in doc.ents:
        table.add_row([ent.text, ent.label_, spacy.explain(ent.label_), ent.
 ↪start_char, ent.end_char])

print(table)
```

```
+------------------+--------+-----------------------------------------+--------
---+----------+
|       token      | label  |               explanation               | Star
Char | End Char |
+------------------+--------+-----------------------------------------+--------
---+----------+
| December 20, 2025 |  DATE  |  Absolute or relative dates or periods  |     4
|    21    |
|   Satya Nadella   | PERSON |       People, including fictional       |     23
|    36    |
|     Microsoft     |  ORG   | Companies, agencies, institutions, etc. |     52
|    61    |
```

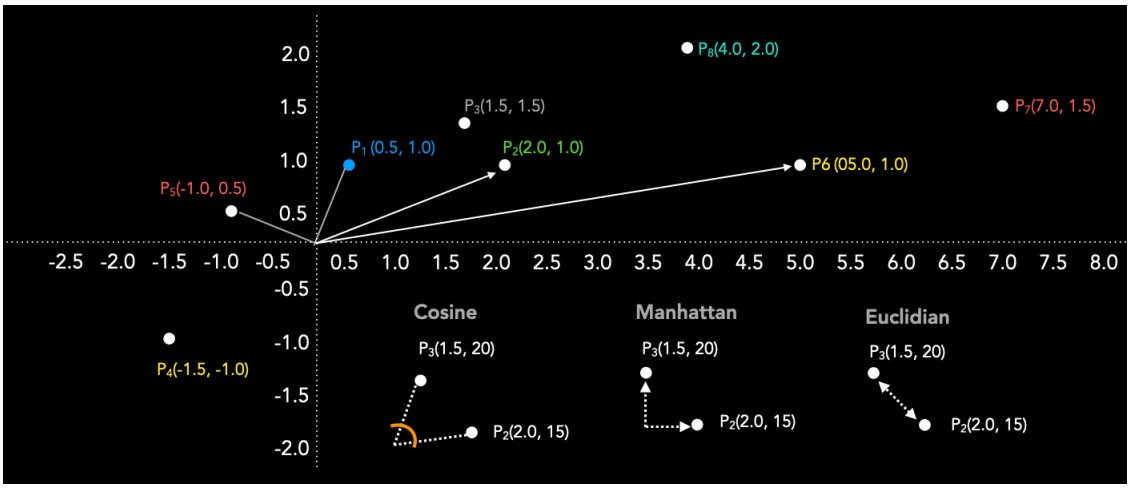
$5 billion	MONEY	Monetary values, including unit	75	85
London	GPE	Countries, cities, states	117	123
Kings Cross	ORG	Companies, agencies, institutions, etc.	153	164
Google	ORG	Companies, agencies, institutions, etc.	203	209
Mountain View	GPE	Countries, cities, states	213	226
California	GPE	Countries, cities, states	228	238

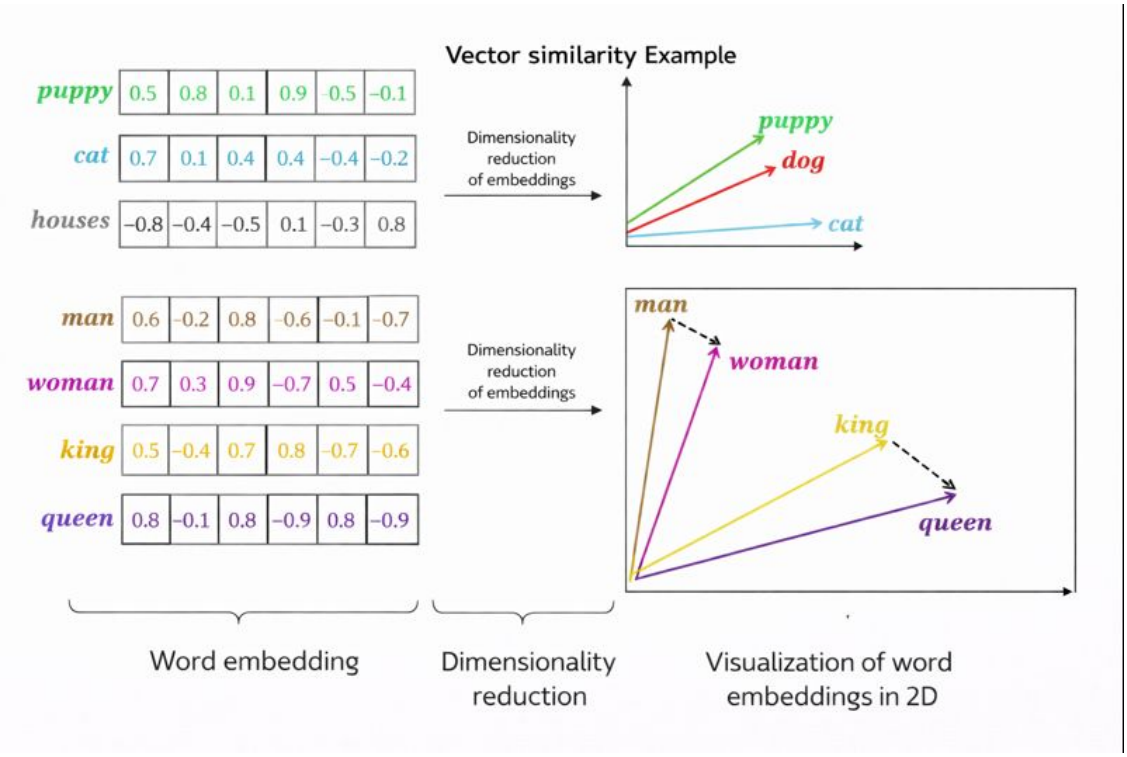
05 Vector Similarity

December 20, 2025

1 Vector Similarity

- **Measures Textual Relatedness**: Similarity in NLP quantifies how closely related two pieces of text are, based on factors like words choice, context, and meaning.
- **Used in Various Applications**: It's Crucial for tasks such as document clustering, recommendations systems, and paraphrase detection, where understanding the degree of similarity between texts is essential.
- **Different Methods**: Similarity can be measured using various approaches, from simple techniques like cosine similarity on word vectors to more advanced methods involving contextual embeddings from models like BERT.





[1]:
```
import spacy
from prettytable import PrettyTable
from spacy import displacy
```

[2]:
```
nlp = spacy.load('en_core_web_lg')
```

[3]:
```
doc = nlp('cat')
vec1 = doc.vector
```

[4]:
```
doc.vector.shape
```

[4]: (300,)

[5]:
```
# doc.vector
```

[6]:
```
doc = nlp('dog')
vec2 = doc.vector
```

[7]:
```
doc.vector.shape
```

[7]: (300,)

[8]:
```
from sklearn.metrics.pairwise import cosine_similarity

sim = cosine_similarity([vec1], [vec2])
print(sim)
```

```
[[0.80168545]]
```

[9]:
```
words = ['dog', 'cat', 'man', 'woman', 'king', 'queen', 'houses']
table = PrettyTable(field_names=['word 1', 'word 2', 'score'])
```

[10]:
```
from itertools import combinations

list(combinations(words, 2))
```

[10]:
```
[('dog', 'cat'),
 ('dog', 'man'),
 ('dog', 'woman'),
 ('dog', 'king'),
 ('dog', 'queen'),
 ('dog', 'houses'),
 ('cat', 'man'),
 ('cat', 'woman'),
 ('cat', 'king'),
 ('cat', 'queen'),
 ('cat', 'houses'),
 ('man', 'woman'),
 ('man', 'king'),
 ('man', 'queen'),
 ('man', 'houses'),
 ('woman', 'king'),
 ('woman', 'queen'),
 ('woman', 'houses'),
 ('king', 'queen'),
 ('king', 'houses'),
 ('queen', 'houses')]
```

[11]:
```
words = ['dog', 'cat', 'man', 'woman', 'king', 'queen', 'houses']
table = PrettyTable(field_names=['word 1', 'word 2', 'score'])
for word1, word2 in combinations(words, 2):
        token1 = nlp(word1)
        token2 = nlp(word2)

        sim = token1.similarity(token2)
        table.add_row([word1, word2, f"{sim:.2f}"])

print(table)
```

```
+--------+--------+-------+
| word 1 | word 2 | score |
+--------+--------+-------+
|  dog   |  cat   |  0.80 |
|  dog   |  man   |  0.42 |
|  dog   | woman  |  0.39 |
```

dog	king	0.25
dog	queen	0.23
dog	houses	0.23
cat	man	0.36
cat	woman	0.36
cat	king	0.28
cat	queen	0.31
cat	houses	0.20
man	woman	0.74
man	king	0.41
man	queen	0.27
man	houses	0.24
woman	king	0.27
woman	queen	0.41
woman	houses	0.21
king	queen	0.73
king	houses	0.21
queen	houses	0.21