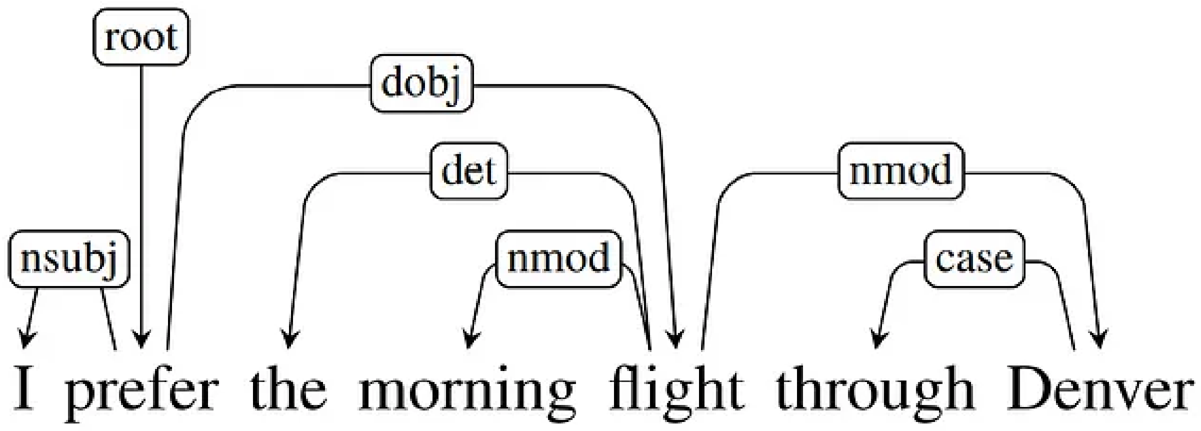
Search Medium Write

Dependency Parsing [NLP, Python]

Yash Jain · Follow
Published in Dev Genius · 4 min read · Mar 9, 2022

37 1



Source: Speech and Language Processing — Daniel, James

Dependency structure shows which word or phrase depends on which other words or phrases. We use dependency-based parsing to analyze and infer both structure and semantic dependencies and relationships between tokens in a sentence.

Relations among the words are illustrated in above figure with directed, *labeled arcs* from **heads to dependents.** It also includes root node that explicitly marks the head of the entire structure of sentence.
(As shown above:-

prefer → is the root node,
flight → is head of → **the** and,
“the” is dependent and is related to flight as ***determiner*** denoted by **`det`**)

Dependency parsing are useful in Information Extraction, Question Answering, Coreference resolution and many more aspects of NLP.

Dependency Formalisms
We are discussing dependency structures that are simply directed graphs. We represent Structures G = (V,A) where V is set of vertices and A is set of arcs- which represents relationship.

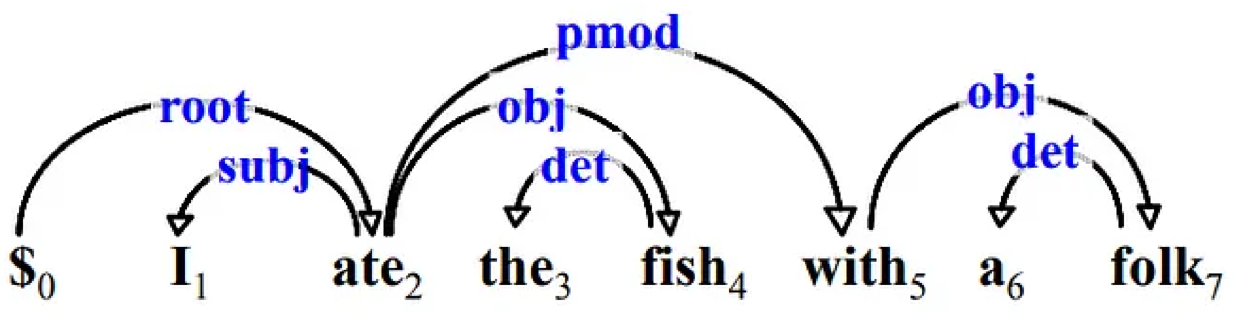
An Dependency tree should meet few constraints:
- Single-head
- Connected
- Acyclic

Taken together, these constraints ensure that each word has a single head, that the dependency structure is connected, and that there is a single root node from which one can follow a unique directed path to each of the words in the sentence

Projectivity

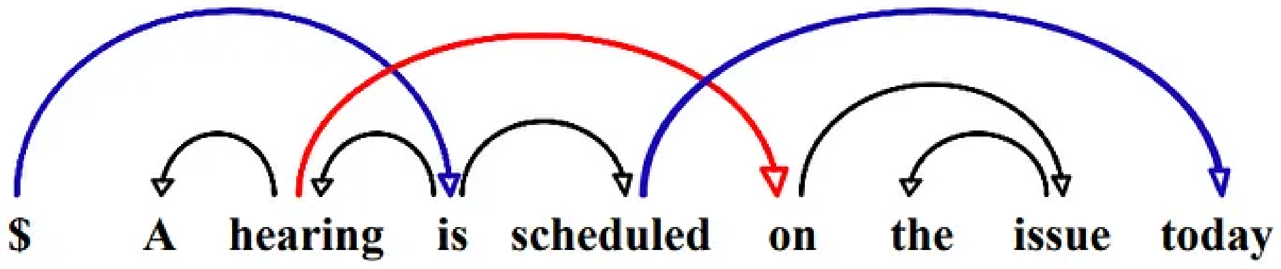
It imposes additional constraint derived from order of words. It is said to be projective if there is a path from head to every word that lies between the head and dependent in sentence. A dependency tree is then said to be projective if all the arcs that make it up are projective.
Informally, “projective” means the tree does not contain any crossing arcs. and A non-projective dependency tree contains crossing arcs.

Projective -



Projective dependency [Source: Dependency Parsing: Past, Present, and Future by Wenliang Chen, Zhenghua Li, Min Zhang]

Non-projective -



Non-projective dependency [Source: Dependency Parsing: Past, Present, and Future by Wenliang Chen, Zhenghua Li, Min Zhang]

If you want to dig deeper I would suggest to look at this document on dependency parsing — link
You should checkout chapter 14 of Speech and Language Processing — by Daniel Jurafsky & James H. Martin.

Dependency Labels

You can checkout label's explanation here.

Implementation

Lets see dependency structure for "The quick brown fox jumping over the lazy dog"

spaCy dependency parsing

```
import spacy
from spacy import displacy
nlp = spacy.load("en_core_web_sm")

sentence = "The quick brown fox jumping over the lazy dog"
doc = nlp(sentence)

print(f"{'Node (from)-->':<15} {'Relation':^10} {'-->Node
(to)':>15}\n")

for token in doc:
    print("{:<15} {:^10} {:>15}".format(str(token.head.text),
str(token.dep_), str(token.text)))

displacy.render(doc, style='dep')
```

`token.dep_` → shows dependency

`token.head.text` → shows head

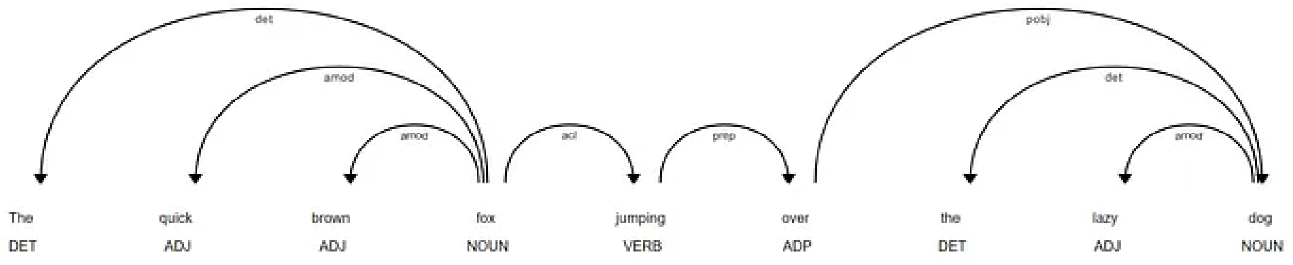
`token.head.text` → shows dependent

`{:<15} {:^10} {:>15}` → its just for formatting string for pretty output. You can checkout how to do this formatting here

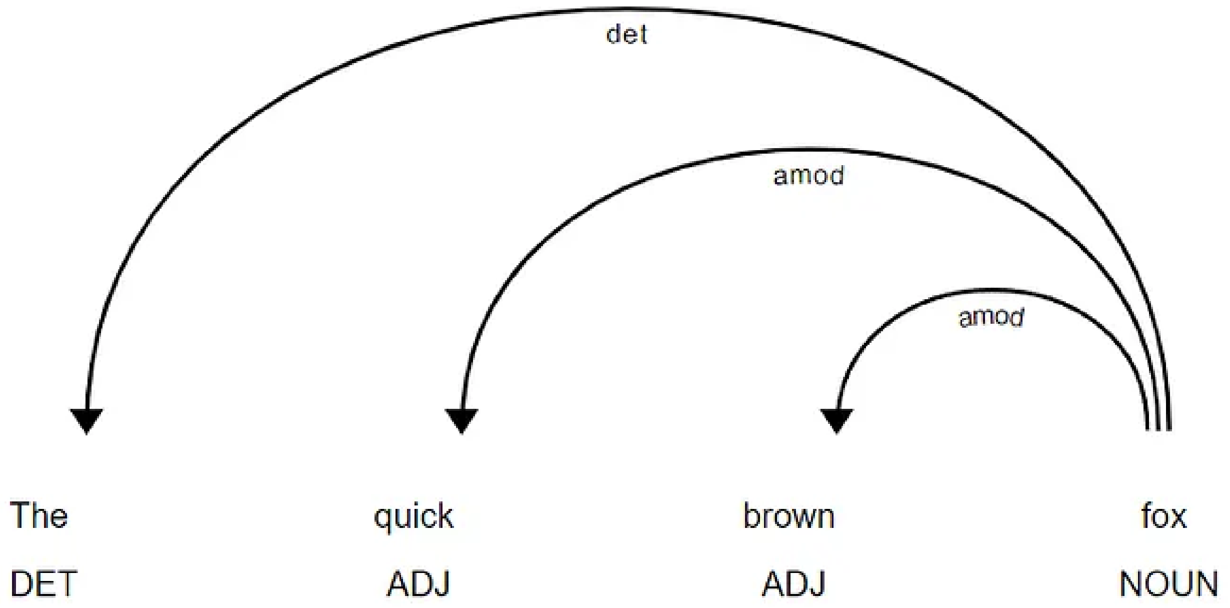
Output:

```
Node (from)-->   Relation     -->Node (to)


fox                det                The
fox                amod             quick
fox                amod             brown
fox                ROOT               fox
fox                acl            jumping
jumping            prep              over
dog                det                the
dog              compound            lazy
over               pobj               dog
```



Let's checkout subpart of this dependency tree



In above figure there is relationship between *fox* and *[the, quick, brown]* as you know **the** is determiner therefore relation directed from "fox" to "The" is named as ***det***(determiner*), and* relation between "fox" to "quick", "brown" is named as ***amod*** *(adjectival modifier* →A word or phrase that modifies meaning of another word, usually a noun).

Stanza dependency parsing

```
import stanza
stanza.download('en')

nlp = stanza.Pipeline(lang='en',
processors='tokenize,mwt,pos,lemma,depparse')

doc = nlp("The quick brown fox jumping over the lazy dog")

for sent in doc.sentences:
    for word in sent.words:
        print(f'id:{word.id}\tword: {word.text}\thead id:
{word.head}\thead: {sent.words[word.head-1].text if word.head > 0
else "root"}\tdeprel: {word.deprel}', sep='\n')
```

`word.deprel` → shows dependency here

Output:

```
id: 1   word: The       head id: 4      head: fox       deprel: det
id: 2   word: quick     head id: 4      head: fox       deprel:
amod
id: 3   word: brown     head id: 4      head: fox       deprel:
amod
id: 4   word: fox       head id: 0      head: root      deprel:
root
id: 5   word: jumping   head id: 4      head: fox       deprel: acl
id: 6   word: over      head id: 9      head: dog       deprel:
case
id: 7   word: the       head id: 9      head: dog       deprel: det
id: 8   word: lazy      head id: 9      head: dog       deprel:
amod
id: 9   word: dog       head id: 5      head: jumping   deprel: obl
```

You can checkout more about stanza dependency parser here.

Spark NLP dependency parser: You can checkout code for this here.

You can try Stanford corenlp dependency parsing online on corenlp.run

I hope this would clear some idea and would help in implementing dependency parsing.

Dependency Parsing NLP Naturallanguageprocessing Python Dependency Graph

More from the list: "NLP"

Curated by Himanshu Birla

Jon Gi... in Towards Data ...

Characteristics of Word Embeddings

11 min read · Sep 4, 2021

Jon Gi... in Towards Data ...

The Word2vec Hyperparameters

6 min read · Sep 3, 2021

Jon Gi... in

The Word2ve

15 min rea

View list

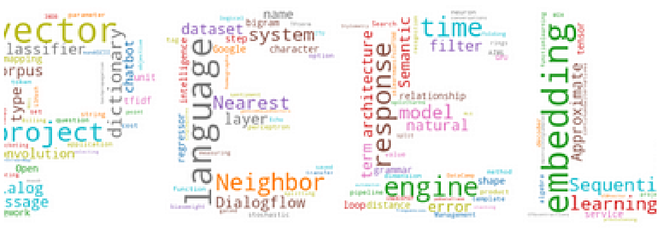
Written by Yash Jain

Follow

93 Followers · Writer for Dev Genius

Data Scientist/ Data Engineer at IBM | Alumnus of @niituniversity | Natural Language Processing | Pronouns: He, Him, His

More from Yash Jain and Dev Genius



Yash Jain

Spell check and correction[NLP, Python]

In Natural Language Processing it's important that spelling errors should be as less as…

4 min read · Feb 19, 2022

21 1

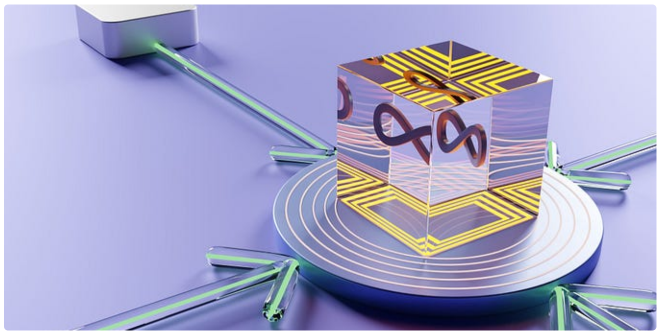
Devansh- Machine Learning Made Si… in Dev Gen…

Amazon Prime Video reduced costs by 90% by ditching…

This article is *not* sponsored by the World Monolith Supremacy Association

6 min read · May 10

1.3K 41



Priyansh Khodiyar in Dev Genius

Yash Jain

Stopwords [NLP, Python]

Python for DevOps — A Definitive Guide

Python Python Python! Apple might have to add the pronunciation of the word Python in...

9 min read · Jul 9

265 6

Stop words are common words in any language that occur with a high frequency b...

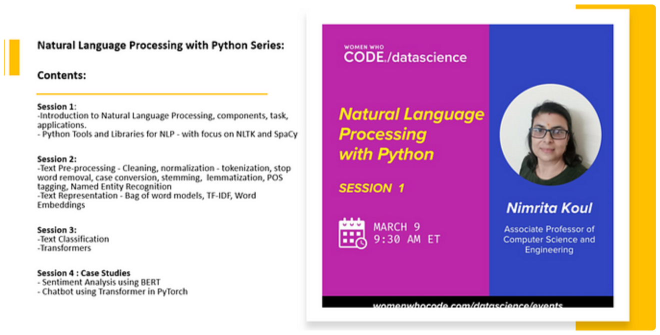
6 min read · Feb 23, 2022

8

See all from Yash Jain

See all from Dev Genius

Recommended from Medium



Natural Language Processing with Python Series:

Contents:

Session 1:
-Introduction to Natural Language Processing, components, task, applications.
- Python Tools and Libraries for NLP - with focus on NLTK and SpaCy

Session 2:
-Text Pre-processing - Cleaning, normalization - tokenization, stop word removal, case conversion, stemming, lemmatization, POS tagging, Named Entity Recognition
-Text Representation - Bag of word models, TF-IDF, Word Embeddings

Session 3:
-Text Classification
-Transformers

Session 4 : Case Studies
- Sentiment Analysis using BERT
- Chatbot using Transformer in PyTorch

WOMEN WHO CODE /datascience
Natural Language Processing with Python
SESSION 1
MARCH 9 9:30 AM ET
Nimrita Koul
Associate Professor of Computer Science and Engineering
womenwhocode.com/datascience/events

Nimrita Koul

NLP with Python Part 2 NLTK

This is the second article in the series of articles on Natural Language Processing…

5 min read · Apr 5

2

Ebo Jackson in Level Up Coding

Comparing Concurrency and Parallelism Techniques in Python:…

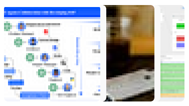
Introduction

5 min read · Jun 13

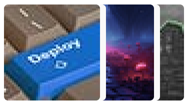
70

Lists

Coding & Development
11 stories · 200 saves

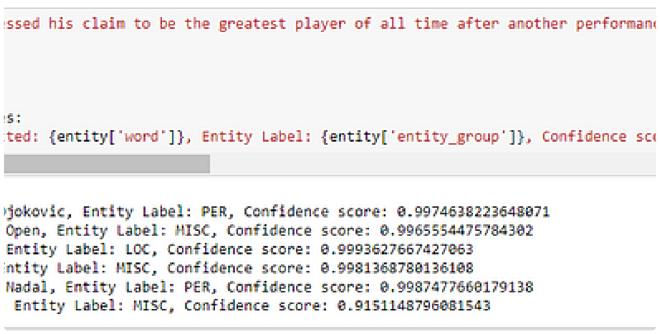


Natural Language Processing
669 stories · 283 saves



Predictive Modeling w/ Python
20 stories · 452 saves

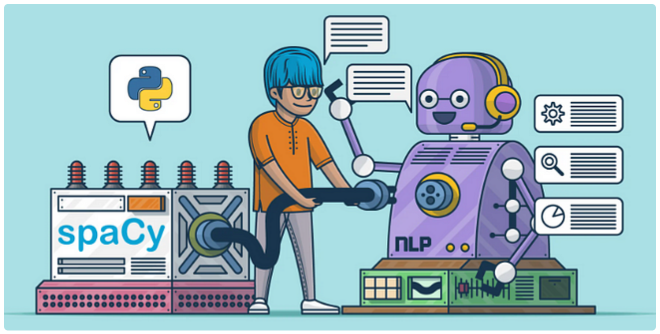
Practical Guides to Machine Learning
10 stories · 519 saves



```
ssed his claim to be the greatest player of all time after another performanc


s:
ted: {entity['word']}, Entity Label: {entity['entity_group']}, Confidence sc

jokovic, Entity Label: PER, Confidence score: 0.9974638223648071
 Open, Entity Label: MISC, Confidence score: 0.9965554475784302
 Entity Label: LOC, Confidence score: 0.9993627667427063
ntity Label: MISC, Confidence score: 0.9981368780136108
 Nadal, Entity Label: PER, Confidence score: 0.9987477660179138
  Entity Label: MISC, Confidence score: 0.9151148796081543
```



Seffa B

Named Entity Recognition with Transformers: Extracting Metadata

3 min read · Jun 12

7

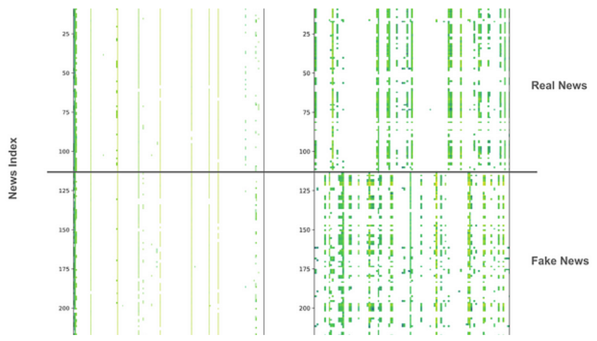
HasancanÇakıcıoğlu

Comprehensive Text Preprocessing NLP (Natural Language...

Text preprocessing plays a crucial role in Natural Language Processing (NLP) by...

12 min read · Jul 9

12



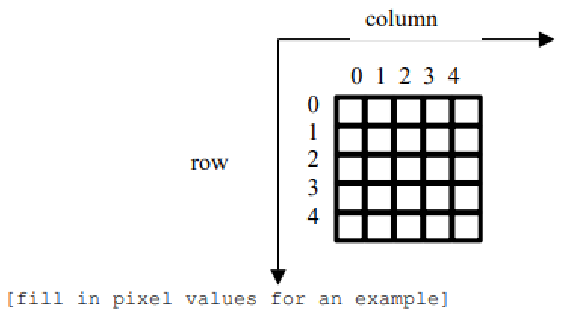
Sherry Wu in Stanford CS224W GraphML Tutorials

Spread No More: Twitter Fake News Detection with GNN

By Li Tian, Sherry Wu, Yifei Zheng as part of the Stanford CS224W course project.

11 min read · May 16

1



Coursesteach

Computer Vision (Part 7)

Understanding How Computers "See" Images

6 min read · Sep 16

See more recommendations