# Spacy-kirjaston peruskomennot

Tämä dokumentti sisältää kattavan listan Spacy-kirjaston peruskomennoista jaoteltuna eri kategorioihin. Jokaisen komennon yhteydessä on lyhyt esimerkki ja selitys sen toiminnasta.

## Tutkimuksen vaiheet

Spacy-kirjaston peruskomentojen muodostamiseksi suoritettiin seuraavat tutkimusvaiheet:

1. Haettiin Spacy-kirjaston dokumentaatiosta kaikki peruskomennot.
2. Jaettiin peruskomennot kategorioihin tekstin lataamisen ja esikäsittelyn, kielimallien lataamisen ja käytön, sanaluokkien ja entiteettien tunnistamisen, syntaktisen analyysin suorittamisen ja tekstin visualisoinnin mukaan.
3. Muodostettiin jokaisesta peruskomennosta lyhyt esimerkki, joka havainnollistaa sen käyttöä.
4. Selitettiin lyhyesti jokaisen peruskomennon tarkoitus ja miten se toimii.
5. Järjestettiin peruskomennot loogiseen järjestykseen helpottamaan niiden oppimista ja käyttöä.

## Tekstin lataaminen ja esikäsittely

Spacy tarjoaa tehokkaita työkaluja tekstin lataamiseen ja esikäsittelyyn. Seuraavassa on listattu joitakin keskeisiä komentoja:

**1. spacy.load()**

- Tarkoitus: Lataa kielimallin, joka sisältää tarvittavat komponentit tekstin käsittelyyn.
- Toiminta: Ottaa argumenttina kielimallin nimen ja palauttaa Language-olion.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Tämä on esimerkkilause.")
```

1

**2. spacy.blank()**

- Tarkoitus: Luo tyhjän kielimallin.
- Toiminta: Ottaa argumenttina kielikoodin (esim. "en" englannille) ja palauttaa Language-olion, joka sisältää vain tokenizerin.

- Käyttötarkoitukset:
  - Kun tarvitset vain tokenizeria.
  - Kun haluat rakentaa mukautetun pipeline-rakenteen alusta alkaen.
  - Testaustarkoituksiin.
- Esimerkki:

Python

```
import spacy

nlp = spacy.blank("en")
doc = nlp("Tämä on esimerkkilause.")
```

2

**3. Doc.sents**

- Tarkoitus: Jakaa tekstin lauseisiin.
- Toiminta: Luo generaattorin, joka iteroi Doc-objektin lauseiden yli.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Tämä on ensimmäinen lause. Tämä on toinen lause.")
for sent in doc.sents:
    print(sent.text)
```

3

**4. Token.text**

- Tarkoitus: Palauttaa tokenin tekstisisällön.
- Toiminta: Antaa token-objektin tekstimuotoisen esityksen.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Tämä on esimerkkilause.")
for token in doc:
    print(token.text)
```

3

**5. Token.is_alpha, Token.is_punct, Token.is_space**

- Tarkoitus: Tarkistaa, onko token aakkonen, välimerkki vai välilyönti.
- Toiminta: Palauttaa totuusarvon (True/False).
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Tämä on esimerkkilause.")
for token in doc:
    print(token.text, token.is_alpha, token.is_punct, token.is_space)
```

3

**6. nlp.pipe()**

- Tarkoitus: Käsittelee tekstiä tehokkaasti erissä.
- Toiminta: Ottaa argumenttina listan tekstejä ja palauttaa generaattorin, joka iteroi Doc-objektien yli.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
texts = ["Tämä on ensimmäinen teksti.", "Tämä on toinen teksti."]
for doc in nlp.pipe(texts):
    print(doc.text)
```

5

### 7. Tokenizerin mukauttaminen

Spacyn tokenizeria voidaan mukauttaa lisäämällä poikkeussääntöjä ja määrittämällä etu-, jälki- ja sisäliitteiden jakamista. Tämä on hyödyllistä, jos haluat käsitellä tekstiä, joka sisältää esimerkiksi erikoismerkkejä tai sanoja, joita ei ole Spacyn sanastossa.

- Poikkeussääntöjen lisääminen: Voit lisätä poikkeussääntöjä tokenizerille nlp.tokenizer.add_special_case() -komennolla.
- Etu-, jälki- ja sisäliitteiden jakaminen: Voit määrittää etu-, jälki- ja sisäliitteiden jakamista tokenizerin asetuksissa.
- Esimerkki:

Python

```
import spacy
import re
from spacy.language import Language

nlp = spacy.load("en_core_web_sm")

# Lisätään poikkeussääntö sanalle "U.K."
nlp.tokenizer.add_special_case("U.K.", [{"ORTH": "U.K."}])

# Määritetään, että pistettä ei jaeta erilliseksi tokeniksi
@Language.component("prevent_period_splitting")
def prevent_period_splitting(doc):
    for token in doc[:-1]:
        if token.text == "." and boundary.match(doc[token.i +
1].text):
            token.is_sent_start = False
    return doc

nlp.add_pipe("prevent_period_splitting", before="parser")

doc = nlp("Tämä on lause, jossa on U.K. ja piste.")
print()
```

3

### 8. DocBin

DocBin-objektia käytetään Doc-objektien tehokkaaseen binääriseen serialisointiin. Tämä on

hyödyllistä suurten tietomäärien tehokkaaseen tallentamiseen ja käsittelyyn.

- Tarkoitus: Tallentaa ja ladata Doc-objekteja tehokkaasti.
- Käyttö: Voit luoda DocBin-objektin ja lisätä siihen Doc-objekteja. Voit sitten serialisoida DocBin-objektin tavuiksi tai tiedostoon.
- Esimerkki:

Python

```
import spacy
from spacy.tokens import DocBin

nlp = spacy.load("en_core_web_sm")
doc_bin = DocBin()
docs = [nlp("Tämä on esimerkkilause.") for _ in range(10)]
for doc in docs:
    doc_bin.add(doc)
bytes_data = doc_bin.to_bytes()

# Lataa Doc-objektit myöhemmin
new_doc_bin = DocBin().from_bytes(bytes_data)
new_docs = list(new_doc_bin.get_docs(nlp.vocab))
```

6

### 9. Lookups

Lookups-objektia käytetään suurien hakutaulukoiden ja sanakirjojen käyttämiseen. Tämä on hyödyllistä esimerkiksi lemmatizerin ja muiden kieliominaisuuksien datan lataamiseen.

- Tarkoitus: Tehokas tapa käyttää suuria hakutaulukoita.
- Käyttö: Voit luoda Lookups-objektin ja ladata siihen dataa tiedostosta tai URL-osoitteesta.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
lookups = nlp.vocab.lookups
lookups.add_table("my_table", {"key1": "value1", "key2": "value2"})
```

[6]

Spacyn tokenizer on erittäin tehokas ja tarkka. Se on suunniteltu käsittelemään suuria tietomääriä nopeasti ja tehokkaasti[7].

# Kielimallien lataaminen ja käyttö

Spacy tarjoaa valmiiksi koulutettuja kielimalleja eri kielille ja eri käyttötarkoituksiin. Kielimalli ladataan spacy.load()-komennolla. Kielimallit sisältävät tyypillisesti seuraavia komponentteja:

- Binääripainot osien puheen tunnistajalle, riippuvuusjäsennimelle ja nimettyjen entiteettien tunnistajalle kontekstissa olevien merkintöjen ennustamiseksi.
- Leksikaaliset merkinnät sanastossa, eli sanat ja niiden kontekstista riippumattomat ominaisuudet, kuten muoto tai oikeinkirjoitus.
- Datatiedostot, kuten lemmatizerin säännöt ja hakutaulukot.
- Sanavektorit, eli sanojen moniulotteiset merkitysesitykset, joiden avulla voit määrittää, kuinka samanlaisia ne ovat toisiinsa nähden.
- Määritysvaihtoehdot, kuten kieli- ja käsittelyputken asetukset ja käytettävät mallien toteutukset, jotta Spacy saadaan oikeaan tilaan putkea ladattaessa. [1]

Esimerkiksi englannin kielen perusmalli ladataan seuraavasti:

Python

```
import spacy

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

Voit ladata myös muita kielimalleja, kuten saksan kielen mallin:

Python

```
import spacy

nlp = spacy.load("de_core_news_sm")
```

[8]

On tärkeää valita oikea kielimalli tiettyyn tehtävään. Spacy tarjoaa erilaisia kielimalleja, joilla on erilaiset ominaisuudet ja koot. Pienemmät mallit ovat nopeampia, mutta niillä on rajoitetumpi sanasto ja ne ovat vähemmän tarkkoja. Suuremmat mallit ovat tarkempia, mutta ne ovat myös

hitaampia ja vaativat enemmän muistia[2].

**Example-objekti**

Example-objektia käytetään Spacy-mallien kouluttamiseen. Se edustaa yhtä koulutusdataa ja sisältää kaksi Doc-objektia: yhden viitetiedolle ja toisen putken ennusteille. Example-objekteja voidaan luoda Example.from_dict()-metodilla[10].

**Corpus-objekti**

Corpus-objektia käytetään merkittyjen korpusten hallintaan koulutus- ja arviointidataa varten. Se tarjoaa tehokkaan tavan ladata ja käsitellä suuria tietomääriä[6].

# Sanaluokkien ja entiteettien tunnistaminen

Spacy tarjoaa useita komentoja sanaluokkien ja entiteettien tunnistamiseen tekstistä. Kieliopilliset merkinnät, kuten osien puheen tunnisteet ja riippuvuussuhteet, voivat parantaa tehtävien, kuten mielipideanalyysin, tekstin luokittelun ja tiedonlouhinnan, tarkkuutta[1].

**1. Token.pos_**

- Tarkoitus: Palauttaa tokenin sanaluokan.
- Toiminta: Määrittää tokenille sanaluokan (esim. substantiivi, verbi, adjektiivi). Spacy käyttää kahdenlaisia osien puheen tunnisteita: karkeajakoisia ja hienojakoisia. Karkeajakoinen tunniste (token.pos_) antaa yleiskuvan sanaluokasta, kun taas hienojakoinen tunniste (token.tag_) antaa tarkemman kuvauksen.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Tämä on esimerkkilause.")
for token in doc:
    print(token.text, token.pos_)
```

1

**2. Token.tag_**

- Tarkoitus: Palauttaa tokenin tarkan sanaluokan.
- Toiminta: Määrittää tokenille tarkan sanaluokan (esim. yksikkösubstantiivi, preesensissä oleva verbi).
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Tämä on esimerkkilause.")
for token in doc:
    print(token.text, token.tag_)
```

1

**3. spacy.explain()**

- Tarkoitus: Selittää sanaluokan tai entiteetin tunnisteen merkityksen.
- Toiminta: Ottaa argumenttina tunnisteen ja palauttaa sen selityksen.
- Esimerkki:

Python

```
import spacy

print(spacy.explain("NN"))  # Tulostaa: noun, singular or mass
print(spacy.explain("ORG")) # Tulostaa: Companies, agencies,
institutions, etc.
```

1

**4. Doc.ents**

- Tarkoitus: Palauttaa Doc-objektin tunnistetut entiteetit.
- Toiminta: Luo generaattorin, joka iteroi Doc-objektin entiteettien yli.
- Esimerkki:

Python

```
import spacy

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

```
doc = nlp("Apple on yritys Kaliforniassa.")
for ent in doc.ents:
    print(ent.text, ent.label_)
```

14

**5. Token.ent_iob_, Token.ent_type_**

- Tarkoitus: Palauttaa tokenin entiteettityypin ja sen IOB-koodauksen.
- Toiminta: Määrittää tokenille entiteettityypin (esim. PERSON, ORG, LOC) ja sen roolin entiteetin sisällä (B-alku, I-sisällä, O-ulkopuolella).
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple on yritys Kaliforniassa.")
for token in doc:
    print(token.text, token.ent_iob_, token.ent_type_)
```

15

**6. Doc.vector ja Span.vector**

Doc.vector ja Span.vector -attribuutit tarjoavat vektorimuotoisia esityksiä dokumenteille ja lausekkeille. Näitä vektoreita voidaan käyttää dokumenttien ja lausekkeiden samankaltaisuuden laskemiseen[1].

- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc1 = nlp("Tämä on ensimmäinen dokumentti.")
doc2 = nlp("Tämä on toinen dokumentti.")
similarity = doc1.similarity(doc2)
print(similarity)
```

### 7. KnowledgeBase

KnowledgeBase-objektia käytetään entiteettien linkittämiseen. Se tarjoaa tavan yhdistää tekstissä esiintyvät entiteetit tietämyskannan tietueisiin[6].

- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
kb = nlp.vocab.lookups.get_table("knowledge_base")
doc = nlp("Pariisi on Ranskan pääkaupunki.")
for ent in doc.ents:
    if ent.text in kb:
        print(ent.text, kb)
```

### 8. spacy-lookups-data

spacy-lookups-data-paketti tarjoaa dataa lemmatizerille ja muille kieliominaisuuksille. Se sisältää esimerkiksi taulukoita sanojen perusmuodoista ja morfologisista ominaisuuksista[5].

## Syntaktisen analyysin suorittaminen

### 1. Token.dep_

- Tarkoitus: Palauttaa tokenin syntaktisen riippuvuussuhteen.
- Toiminta: Määrittää tokenille sen roolin lauseen syntaktisessa rakenteessa (esim. subjekti, objekti, predikatiivi).
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Kissa istuu matolla.")
for token in doc:
    print(token.text, token.dep_)
```

11

### 2. Token.head

- Tarkoitus: Palauttaa tokenin päätokenin.
- Toiminta: Määrittää tokenille sen päätokenin, josta se on syntaktisesti riippuvainen.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Kissa istuu matolla.")
for token in doc:
    print(token.text, token.head.text)
```

11

### 3. Doc.noun_chunks

- Tarkoitus: Palauttaa Doc-objektin substantiivilausekkeet.
- Toiminta: Luo generaattorin, joka iteroi Doc-objektin substantiivilausekkeiden yli.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Suuri punainen auto ajaa tiellä.")
for chunk in doc.noun_chunks:
    print(chunk.text)
```

1

### 4. Token.children, Token.lefts, Token.rights

- Tarkoitus: Palauttaa tokenin lapsi-, vasen- tai oikeapuoleiset tokenit syntaktisessa puussa.
- Toiminta: Luo generaattorin, joka iteroi tokenin lapsi-, vasen- tai oikeapuoleisten tokenien yli.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Kissa istuu matolla.")
for token in doc:
    print(token.text,)
```

11

**5. Token.subtree, Token.ancestors**

- Tarkoitus: Palauttaa tokenin alipuun tai esivanhemmat syntaktisessa puussa.
- Toiminta: Luo generaattorin, joka iteroi tokenin alipuun tai esivanhempien yli.
- Esimerkki:

Python

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Kissa istuu matolla.")
for token in doc:
    print(token.text,)
```

11

# Rule-based Matching

Matcher-objektia käytetään sääntöpohjaiseen hakuun. Voit luoda malleja ja käyttää Matcher-objektia löytääksesi vastaavuuksia tekstistä[6].

- Esimerkki:

Python

```python
import spacy
from spacy.matcher import Matcher

nlp = spacy.load("en_core_web_sm")
matcher = Matcher(nlp.vocab)
pattern = [{"LOWER": "hello"}, {"IS_PUNCT": True}, {"LOWER": "world"}]
matcher.add("HelloWorld",)
doc = nlp("Hello, world!")
matches = matcher(doc)
for match_id, start, end in matches:
    string_id = nlp.vocab.strings  # Get string representation
    span = doc[start:end]  # The matched span
    print(match_id, string_id, start, end, span.text)
```

# Suuret kielimallit (LLM)

spacy-llm-paketti mahdollistaa suurten kielimallien (LLM) integroinnin Spacy-putkiin. Se tarjoaa modulaarisen järjestelmän nopeaan prototyyppien luomiseen ja kehotteiden muodostamiseen, ja se muuntaa jäsentämättömiä vastauksia vankkoiksi tuloksiksi erilaisiin NLP-tehtäviin ilman koulutusdataa[17].

spacy-llm:n keskeisiä ominaisuuksia ovat:

- Serialisoitava LLM-komponentti kehotteiden integroimiseksi putkeen.
- Modulaariset funktiot tehtävän (kehotteiden muodostaminen ja jäsentäminen) ja mallin (käytettävä malli) määrittelemiseksi.
- Tuki isännöidyille API:ille ja itse isännöidyille avoimen lähdekoodin malleille.
- Integrointi LangChainiin.
- Pääsy OpenAI API:iin, mukaan lukien GPT-4 ja useita GPT-3-malleja.
- Sisäänrakennettu tuki useille Hugging Facessa isännöidyille avoimen lähdekoodin malleille.
- Käyttöesimerkkejä tavallisille NLP-tehtäville, kuten nimettyjen entiteettien tunnistaminen ja tekstin luokittelu.
- Oman funktion helppo toteutus rekisterin kautta mukautettuja kehotteita, jäsentämistä ja mallien integrointia varten.

Voit lisätä LLM:n Spacy-putkeen seuraavasti:

Python

```python
import spacy
import spacy_llm

nlp = spacy.load("en_core_web_sm")
nlp.add_pipe("llm", config={"model": "gpt-3"})
```

# Tekstin visualisointi

Spacy tarjoaa työkaluja tekstin visualisointiin. Nämä työkalut voivat olla hyödyllisiä virheenkorjauksessa ja kehitysprosessin nopeuttamisessa.

**1. displacy.serve()**

- Tarkoitus: Visualisoi Doc-objektin riippuvuuspuun tai entiteetit selaimessa.
- Toiminta: Käynnistää web-palvelimen, joka näyttää visualisoinnin.
- Esimerkki:

Python

```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Tämä on esimerkkilause.")
displacy.serve(doc, style="dep")
```

18

**2. displacy.render()**

- Tarkoitus: Luo HTML-merkintää Doc-objektin riippuvuuspuusta tai entiteeteistä.
- Toiminta: Palauttaa merkkijonon, joka sisältää HTML-merkintää.
- Esimerkki:

Python

```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Tämä on esimerkkilause.")
html = displacy.render(doc, style="dep")
print(html)
```

18

# Yhteenveto

Spacy-kirjasto tarjoaa laajan valikoiman työkaluja tekstin analysointiin ja käsittelyyn. Yllä oleva lista peruskomennoista auttaa sinua alkuun Spacyn käytössä. Muista tutustua myös Spacyn dokumentaatioon saadaksesi lisätietoja komennoista ja niiden käyttötavoista. Spacy on tehokas ja monipuolinen työkalu, joka soveltuu monenlaisiin NLP-tehtäviin. Sen avulla voit esimerkiksi jäsentää tekstiä, tunnistaa entiteettejä ja suorittaa syntaktista analyysia.

### Lähdeartikkelit


1. spaCy 101: Everything you need to know, avattu maaliskuuta 4, 2025, https://spacy.io/usage/spacy-101
2. Models & Languages · spaCy Usage Documentation, avattu maaliskuuta 4, 2025, https://spacy.io/usage/models
3. SpaCy Package - Text Analysis - Guides at Penn Libraries, avattu maaliskuuta 4, 2025, https://guides.library.upenn.edu/penntdm/python/spacy
4. DependencyParser · spaCy API Documentation, avattu maaliskuuta 4, 2025, https://spacy.io/api/dependencyparser
5. Saving and Loading · spaCy Usage Documentation, avattu maaliskuuta 4, 2025, https://spacy.io/usage/saving-loading
6. Library Architecture · spaCy API Documentation, avattu maaliskuuta 4, 2025, https://spacy.io/api
7. NLP Preprocessing using Spacy - Soshace, avattu maaliskuuta 4, 2025, https://soshace.com/2023/04/05/nlp-preprocessing-using-spacy/
8. How to install a language model - Stack Overflow, avattu maaliskuuta 4, 2025, https://stackoverflow.com/questions/57837315/how-to-install-a-language-model
9. explosion/spacy-models: Models for the spaCy Natural Language Processing (NLP) library - GitHub, avattu maaliskuuta 4, 2025, https://github.com/explosion/spacy-models
10. Data formats · spaCy API Documentation, avattu maaliskuuta 4, 2025, https://spacy.io/api/data-formats
11. Linguistic Features · spaCy Usage Documentation, avattu maaliskuuta 4, 2025, https://spacy.io/usage/linguistic-features
12. Parts of Speech Tagging and Dependency Parsing using spaCy | NLP | Part 3, avattu maaliskuuta 4, 2025, https://ashutoshtripathi.com/2020/04/13/parts-of-speech-tagging-and-dependency-parsing-using-spacy-nlp/
13. Named Entity Recognition (NER) in Python with Spacy - Analytics Vidhya, avattu maaliskuuta 4, 2025, https://www.analyticsvidhya.com/blog/2021/06/nlp-application-named-entity-recognition-ner-in-python-with-spacy/
14. Entity Extraction and Classification using SpaCy - Kaggle, avattu maaliskuuta 4, 2025, https://www.kaggle.com/code/curiousprogrammer/entity-extraction-and-classification-using-spacy
15. EntityRecognizer · spaCy API Documentation, avattu maaliskuuta 4, 2025, https://spacy.io/api/entityrecognizer
16. Processing texts using spaCy - Applied Language Technology, avattu maaliskuuta 4, 2025, https://applied-language-technology.mooc.fi/html/notebooks/part_ii/03_basic_nlp.html
17. Large Language Models · spaCy Usage Documentation, avattu maaliskuuta 4, 2025,

https://spacy.io/usage/large-language-models
18. spaCy - Visualization Function - TutorialsPoint, avattu maaliskuuta 4, 2025, https://www.tutorialspoint.com/spacy/spacy_visualization_function.htm
19. Visualizers · spaCy Usage Documentation, avattu maaliskuuta 4, 2025, https://spacy.io/usage/visualizers