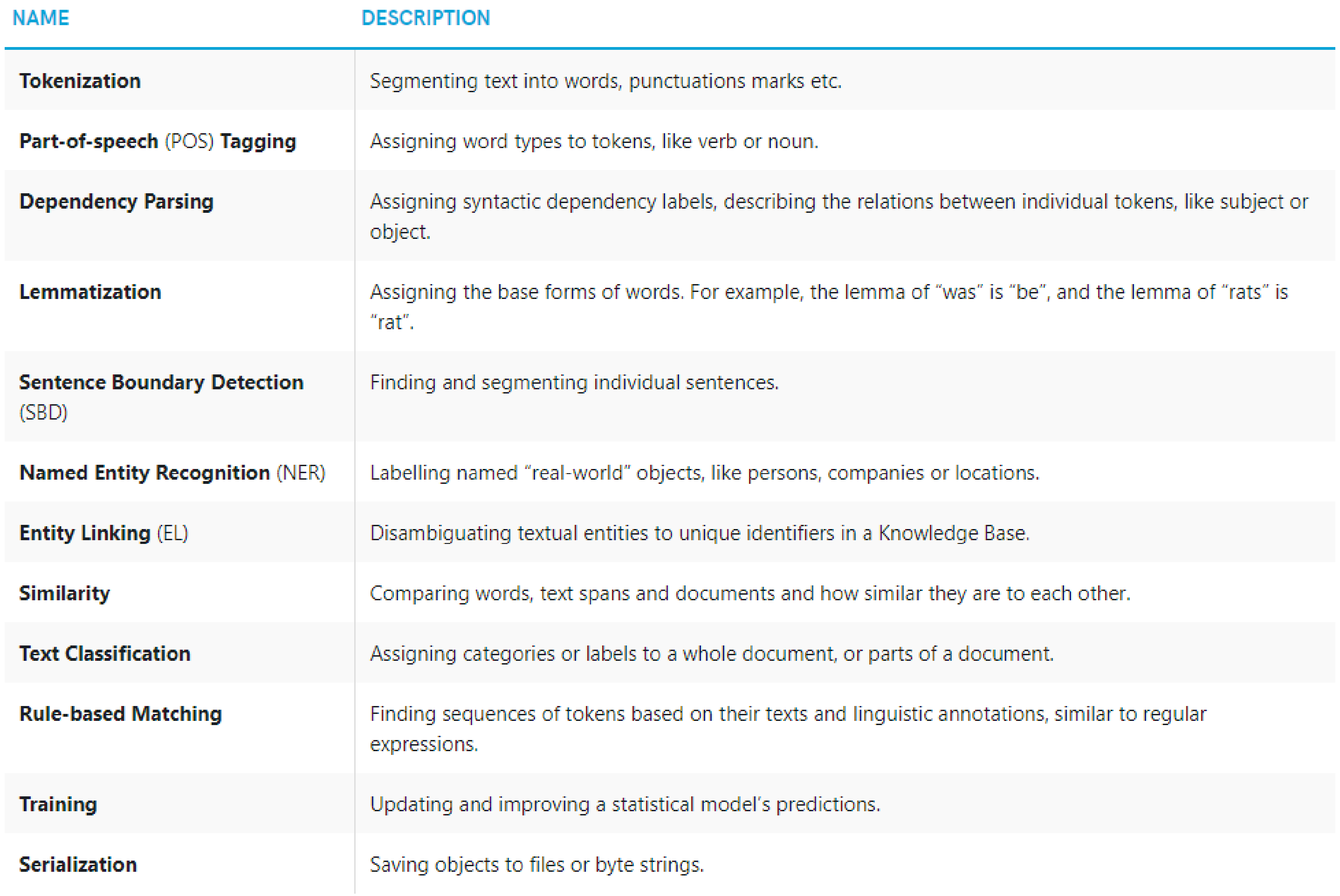
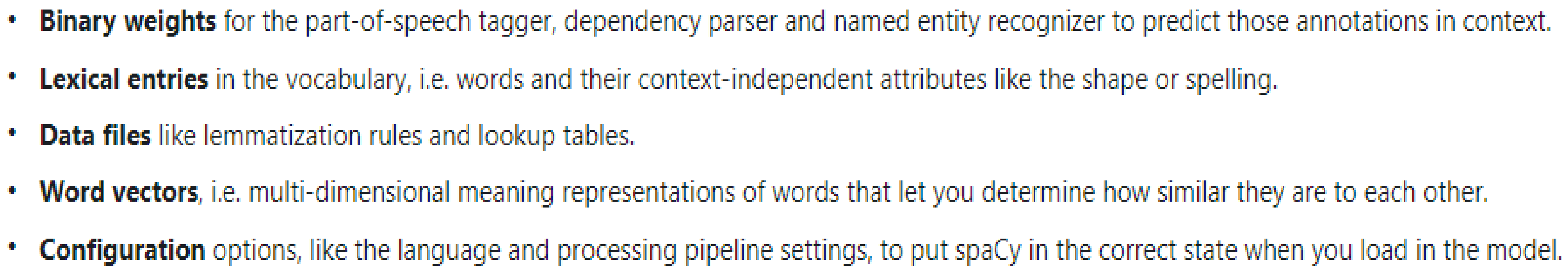
# spaCy

# Features



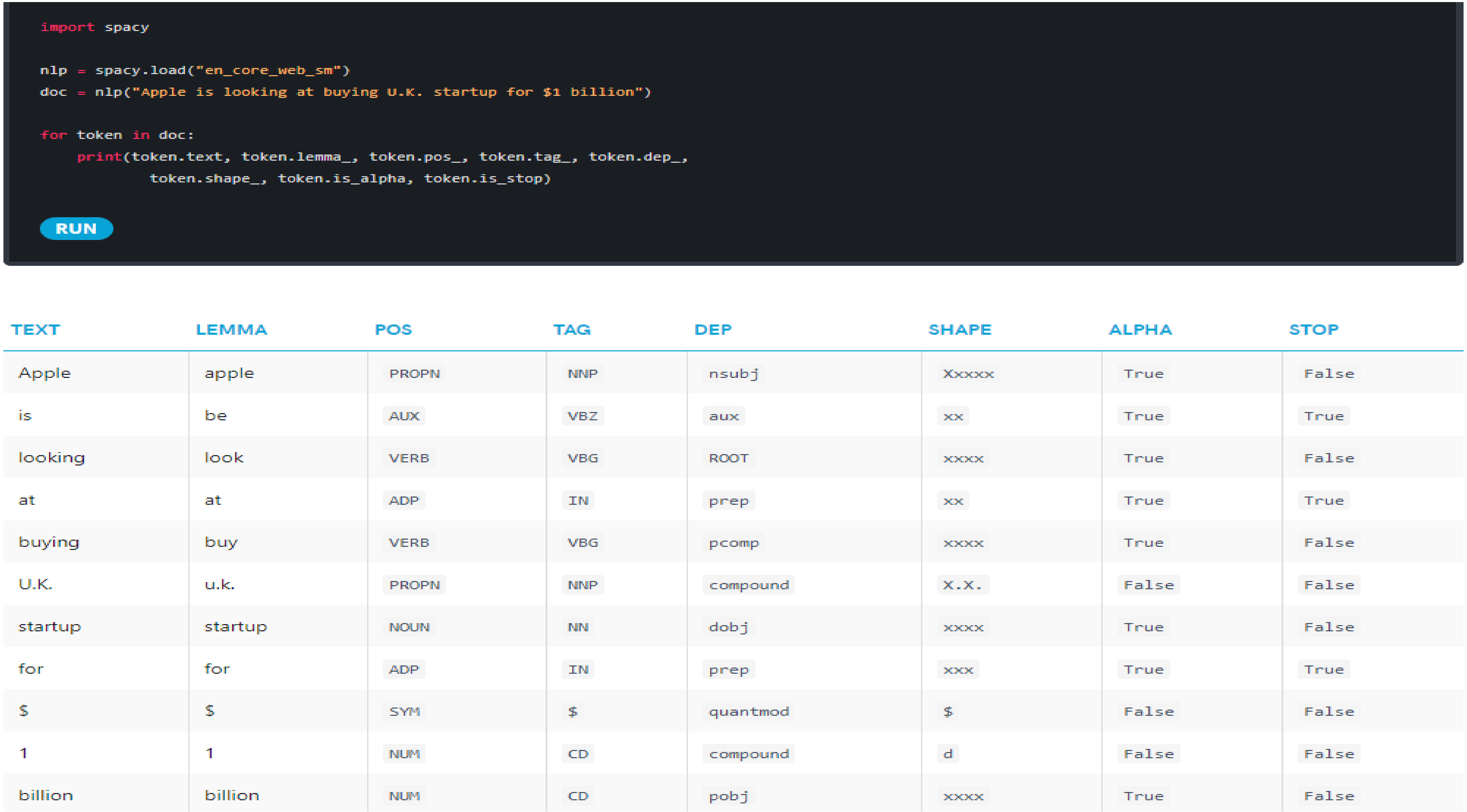
| NAME | DESCRIPTION |
|---|---|
| **Tokenization** | Segmenting text into words, punctuations marks etc. |
| **Part-of-speech** (POS) **Tagging** | Assigning word types to tokens, like verb or noun. |
| **Dependency Parsing** | Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object. |
| **Lemmatization** | Assigning the base forms of words. For example, the lemma of "was" is "be", and the lemma of "rats" is "rat". |
| **Sentence Boundary Detection** (SBD) | Finding and segmenting individual sentences. |
| **Named Entity Recognition** (NER) | Labelling named "real-world" objects, like persons, companies or locations. |
| **Entity Linking** (EL) | Disambiguating textual entities to unique identifiers in a Knowledge Base. |
| **Similarity** | Comparing words, text spans and documents and how similar they are to each other. |
| **Text Classification** | Assigning categories or labels to a whole document, or parts of a document. |
| **Rule-based Matching** | Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions. |
| **Training** | Updating and improving a statistical model's predictions. |
| **Serialization** | Saving objects to files or byte strings. |

# Model (ex: en_core_web_sm)



- **Binary weights** for the part-of-speech tagger, dependency parser and named entity recognizer to predict those annotations in context.
- **Lexical entries** in the vocabulary, i.e. words and their context-independent attributes like the shape or spelling.
- **Data files** like lemmatization rules and lookup tables.
- **Word vectors**, i.e. multi-dimensional meaning representations of words that let you determine how similar they are to each other.
- **Configuration** options, like the language and processing pipeline settings, to put spaCy in the correct state when you load in the model.

# POS tags



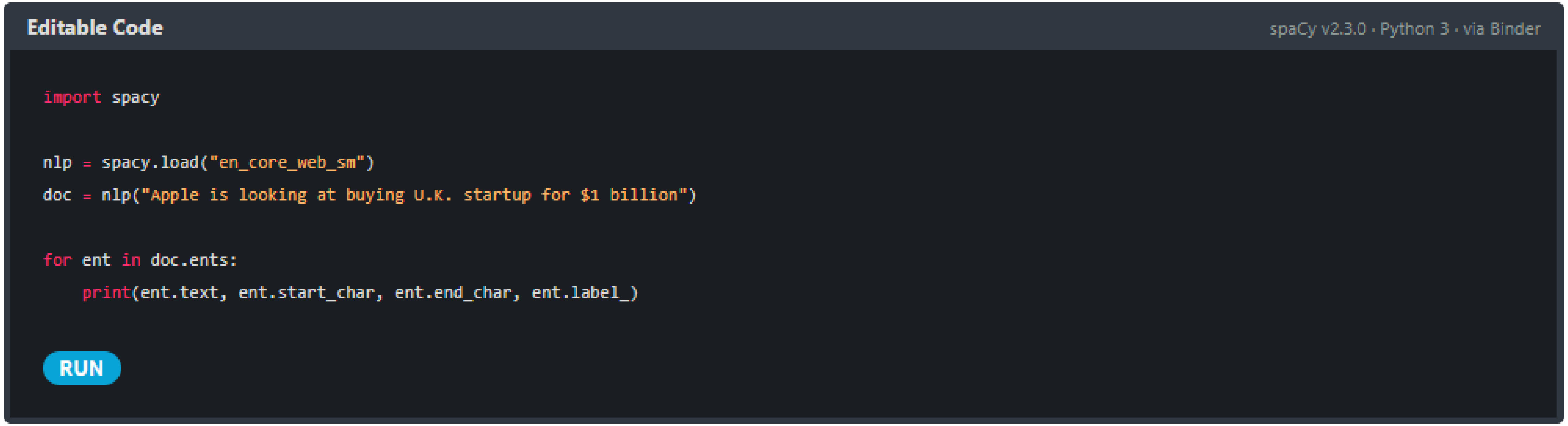
```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
            token.shape_, token.is_alpha, token.is_stop)
```

| TEXT | LEMMA | POS | TAG | DEP | SHAPE | ALPHA | STOP |
|---|---|---|---|---|---|---|---|
| Apple | apple | PROPN | NNP | nsubj | Xxxxx | True | False |
| is | be | AUX | VBZ | aux | xx | True | True |
| looking | look | VERB | VBG | ROOT | xxxx | True | False |
| at | at | ADP | IN | prep | xx | True | True |
| buying | buy | VERB | VBG | pcomp | xxxx | True | False |
| U.K. | u.k. | PROPN | NNP | compound | X.X. | False | False |
| startup | startup | NOUN | NN | dobj | xxxx | True | False |
| for | for | ADP | IN | prep | xxx | True | True |
| $ | $ | SYM | $ | quantmod | $ | False | False |
| 1 | 1 | NUM | CD | compound | d | False | False |
| billion | billion | NUM | CD | pobj | xxxx | True | False |

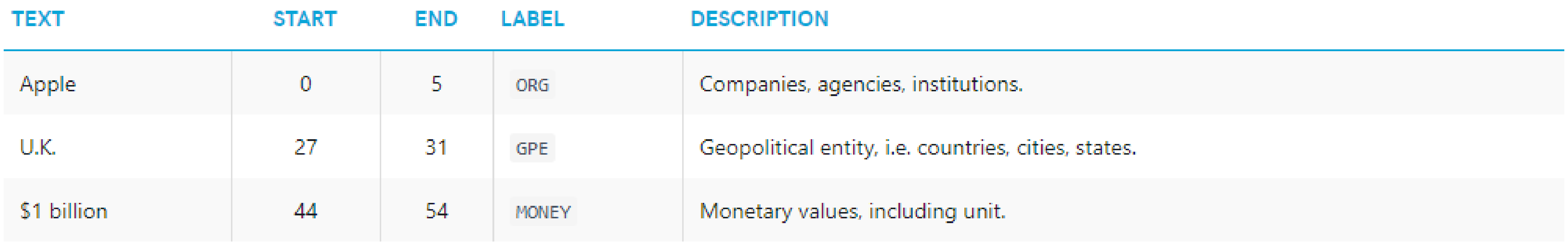
# Named Entities (real-world objects)



```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```



| TEXT | START | END | LABEL | DESCRIPTION |
|---|---|---|---|---|
| Apple | 0 | 5 | ORG | Companies, agencies, institutions. |
| U.K. | 27 | 31 | GPE | Geopolitical entity, i.e. countries, cities, states. |
| $1 billion | 44 | 54 | MONEY | Monetary values, including unit. |