Theory: Introduction to Stanza

19 minutes 0 / 5 problems solved

We have covered two libraries, NLTK and SpaCy. Of course, there are many other libraries for text processing. **Stanza** is one of them. This is a collection of tools for 66 natural languages: Russian, English, Bulgarian, Czech, etc. It allows users to split texts into sentences and words, extract base word forms and their morphological features, analyze the syntactic structure of sentences, or train new models. In this topic, we are going to cover the basic features of Stanza.

§1. Installation

To use Stanza, you need to install it with *pip*.

```
pip install stanza
```

You can import it afterward.

```
import stanza
```

It is essential to download a pre-trained Stanza language model before commencing NLP tasks. You can do it using the `stanza.download()` command. You can either specify a full language name (for instance, "English") or write down a short code (for instance, "en").

```
stanza.download("en")
```

You can find more information on models and languages in the Stanza Available Models & Languages official documentation section.

Of course, Stanza can't cover all the existing human languages, but you can add a new language and train a Stanza package for it. For more information visit this page which contains a blow-by-blow description of adding a language.

§2. Stanza vs SpaCy

In our previous topics, we have discussed the differences between SpaCy and NLTK. This time we are going to cover Stanza's advantages over SpaCy. Their main features are presented in the table below.

Criteria	SpaCy	Stanza
Number of Supported Languages	15	66
Raw Text Processing	Yes	Yes
Fully Neural System	No	Yes
Pretrained Models	Yes	Yes
State-of-the-art Performance	No	Yes

Stanza can be a useful alternative for other popular NLP toolkits. It can be easily adapted to different texts, and the result of state-of-the-art performance will stay at a high level compared to the existed libraries. If you want to learn more about the differences between Stanza and other NLP libraries, you can read the article Stanza: A Python Natural Language Processing Toolkit for Many Human Languages published by the Association for Computational Linguistics.

§3. NLP in Stanza

Current topic:

Introduction to Stanza

Topic depends on:

✕ Overview of SpaCy

Table of contents:

Stanza supports a lot of **processors** of the text processing pipeline. *Processors* in Stanza are the procedures that can be used for text processing: tokenization, lemmatization, etc. To work with them, you first need to install them using `stanza.Pipeline()`:

```
nlp = stanza.Pipeline(lang="en", processors="tokenize, pos, lemma, depparse, ner, sentiment")
```

We store the instance of the `Pipeline` class in the `nlp` variable, which we are going to use later on. Note that it accepts the language you are going to work with, as well as the names of the desired processors. If you do not need some of the processors for your experiments, you may omit them.

To start working with a text, first, we need to create the `doc` variable by giving our text as an argument to the `nlp` pipeline:

```
doc = nlp('There are a lot of vegetables in my kitchen garden! My granny will gather all of them!')
```

And that is basically it! Our text is now processed with all the specified processors, all we need is to get access to this knowledge. For example, the text is split into sentences and tokens that can be accessed by iterating over them like in the following example:

```
for sentence in doc.sentences:
    print(sentence.text)
# There are a lot of vegetables in my kitchen garden!
# My granny will gather all of them!

second_sentence = doc.sentences[1]
for word in second_sentence.words:
    print(word.text)
# My

# granny

# will

# gather

# all

# of

# them

# !
```

Let's have a closer look at the `doc.sentences` . It begins like this:


```
[[
  {
    "id": 1,
    "text": "There",
    "lemma": "there",
    "upos": "PRON",
    "xpos": "EX",
    "head": 2,
    "deprel": "expl",
    "misc": "start_char=0|end_char=5",
    "ner": "O"
  },
  {
    "id": 2,
    "text": "are",
    "lemma": "be",
    "upos": "VERB",
    "xpos": "VBP",
    "feats": "Mood=Ind|Tense=Pres|VerbForm=Fin",
    "head": 0,
    "deprel": "root",
    "misc": "start_char=6|end_char=9",
    "ner": "O"
```

It produces a list of lists. Each sublist represents one sentence and contains dictionaries with information about each token: its id in a sentence, its morphological features, its headword, etc. — all information that our pipeline covers. We will explain most of them in more detail in the following sections. For now, remember the general structure of the `doc.sentences`.

§4. POS-tagging and Lemmatization

We can use Stanza to obtain lemmas and POS-tags for each word in the sentence. Use the `word.lemma` method for obtaining lemmas, and the `word.pos` method for POS-tags, and the `word.feats` for obtaining morphological features of each word form:

```
doc = nlp('Call the police! My cat is missed!')
for sentence in doc.sentences:
    for word in sentence.words:
        print(word.text, '-->', word.lemma, ':', word.feats)
# Call --> call : Mood=Imp|VerbForm=Fin
# the --> the : Definite=Def|PronType=Art
# police --> police : Number=Plur
# ! --> ! : None
# My --> my : Number=Sing|Person=1|Poss=Yes|PronType=Prs

# cat --> cat : Number=Sing

# is --> be : Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin

# missed --> miss : Tense=Past|VerbForm=Part|Voice=Pass

# ! --> ! : None
```

There are a lot of different tags that specify the morphological features of words. For instance, the `PronType=Art` refers to the article and the `Poss=Yes` means that the pronoun is possessive. See the Universal Features section of the official documentation to find more information about the tags.

§5. Syntactic Parsing

We can parse the syntactic relations between words in the sentence in Stanza. Each `word` in the sentence has the `head` and the **dependency relation** between the words are shown with `deprel`. The **dependency relation** can refer to the syntactic properties of a sentence, semantic properties, or their combination.

```
doc = nlp("The cat is here!")
for sent in doc.sentences:
    for word in sent.words:
        print(word.text, '<--', sent.words[word.head-
1].text if word.head > 0 else "root", ':', word.deprel)
    # The <-- cat : det
    # cat <-- here : nsubj
    # is <-- here : cop
    # here <-- root : root
    # ! <-- here : punct
```

The first element in the sequence is a dependant word, the second element is a head, and the last one is a type of relation. Mind the `sent.words[word.head-1].text`. If you recall the picture with dictionaries in one of the previous sections, you can notice that an id of each word starts from 1, meanwhile, the indexing of elements in Python lists starts with 0. So, we should deduce 1 from each index to get the right headword for each dependant one.

As for dependencies, the `det` refers to the relations between the definite article (dependant word) and the noun (head). You can find more information about the dependency types in the Universal Dependency Relations section.

In the example above, if the word `root` is in the columns with the `head` values, it means that the analyzed word is treated as the root of the sentence.

§6. Named Entity Recognition

In Stanza, we cam also perform named entity recognition:

```
doc = nlp('London is the capital of Great Britain.')
for ent in doc.ents:
    print(ent.text, ':', ent.type)
# London : GPE
# Great Britain : GPE
```

As you can see, Stanza correctly identifies *London* and *Great Britain* as geopolitical entities (`GPE`). We use the `ent.text` for printing a named entity itself and the `ent.type` to specify the named entity class. If you want to find out about other entity types, read the Available NER Models section.

§7. Sentiment Analysis

The sentiment analysis is used to classify opinions as negative, neutral, or positive. The result of the sentiment analysis in Stanza is represented by `0`, `1`, or `2` respectively:

```
doc = nlp('Yesterday I saw the film. It was awful.')
for sentence in doc.sentences:
    print(sentence.sentiment)
# 1
# 0
```

In the example above, we can see that Stanza classified the first sentence as a neutral one, while the second one is thought to contain negative information.

§8. Conclusion

In this topic, we have explained how to work with Stanza. So far, we have learned:

- how to install Stanza and download language models;
- the common and distinct features of Stanza and SpaCy;
- how to implement the basic procedures of NLP.

If you want to learn more about Stanza, read the documentation on the official site.

Report a typo

10 users liked this theory. **0** didn't like it. **What about you?**

Start practicing

Comments (5) Hints (0) Useful links (0) Show discussion