# Обзор парсеров NLTK, UDPipe, Stanford NLP, Spacy

Дарья Матяш
Ника Смилга
Антон Бузанов
Михаил Папоротский

# Задачи

- Для использования моделей, опирающихся на синтаксис - определение подлежащего, границ клауз и тд, нужен синтаксический парсинг текста.
- Парсер должен отвечать следующим характеристикам:
- работает как библиотека на питоне
- Имеет понятный тагсет частеречной разметки
- Выдает conllu файлы

# Тестовая выборка

- Предложены 3 текста из нашего корпуса, в которых допущены разного рода ошибки, например, misspelling, которое может повлиять на неправильное определение парсером части речи и синтаксической функции в предложении.

# Spacy

+ быстрый

+ достаточно просто сохранить файл в csv

+ достаточно просто сохранить файл в .conllu

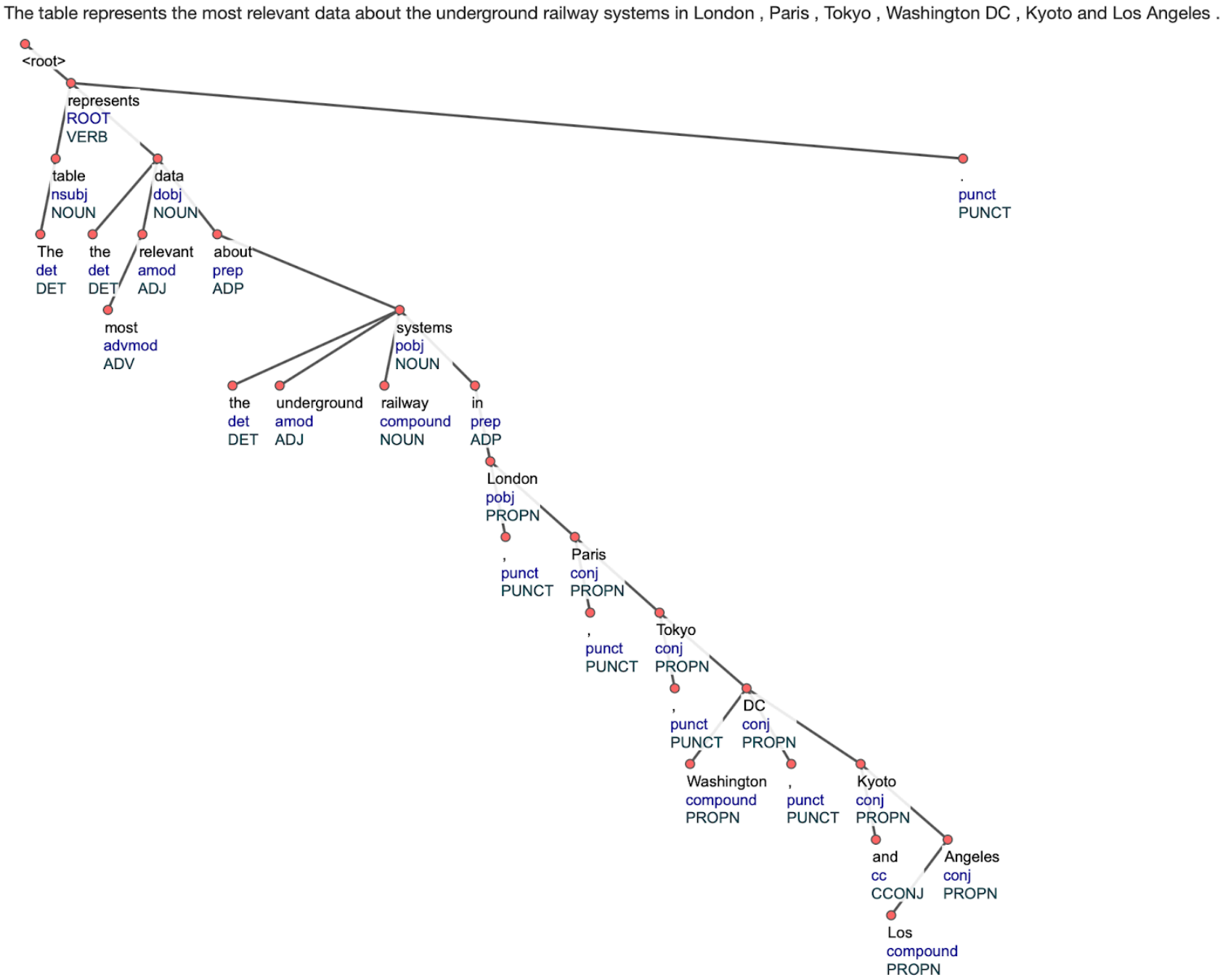
+ хорошо понимает однородные члены

+ делит предложения на клаузы и считает их отдельно, например, предложение *As we can see, the cities' railway systems can be compared by 3 criterias: date opened, kilometres of route and a number of passengers per year.* он делит на 2 предложения.

- нехорошо определяет вершину: например, из пары [вспомогательный глагол - смысловой глагол] выбирается смысловой -> возникает проблема: в предложении *As we can see , the cities '* railway *systems can be compared by 3 criterias - can, be, by* и *systems* считаются зависимыми от *compared*, в похожих случаях, наверное, будет тяжело определить согласование подлежащего со сказуемым

НО! Это особенность и других парсеров (см. далее обзор и выод)

# Spacy, пример



# UDPipe (Ника Смилга)

+ достаточно быстрый

+ в принципе, с ним можно работать в Питоне, тк есть написанные кем-то прекрасным работающие bindings (см. мой код, я их сперла)

- изначально написан под **C++**

- хорошо **понимает только границы реальных предложений** (просто все от точки до точки размечается, нет деления на отдельные клаузы внутри, итоговое число размеченный предложений = исходному числу предложений)

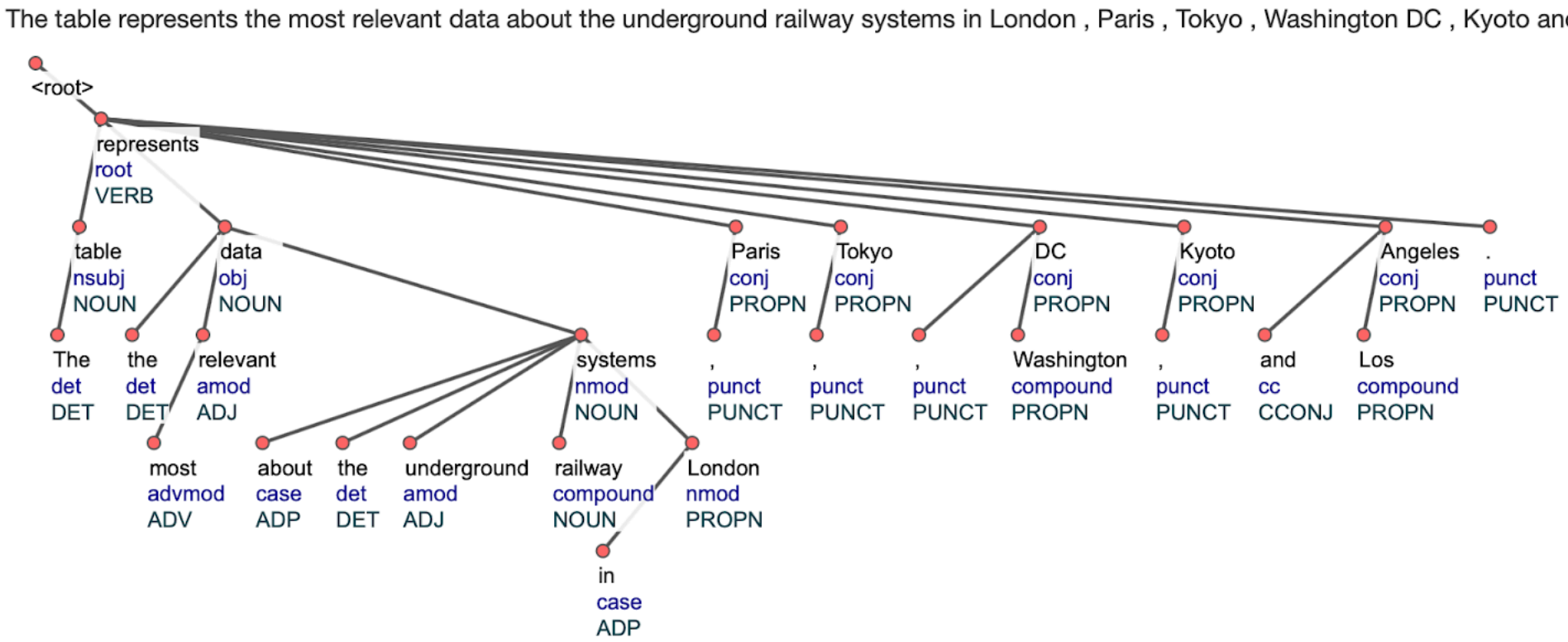
- **плохо** работает с **однородными** членами предложения

- нечто непонятное с **именными группами** (у spacy адекватнее)

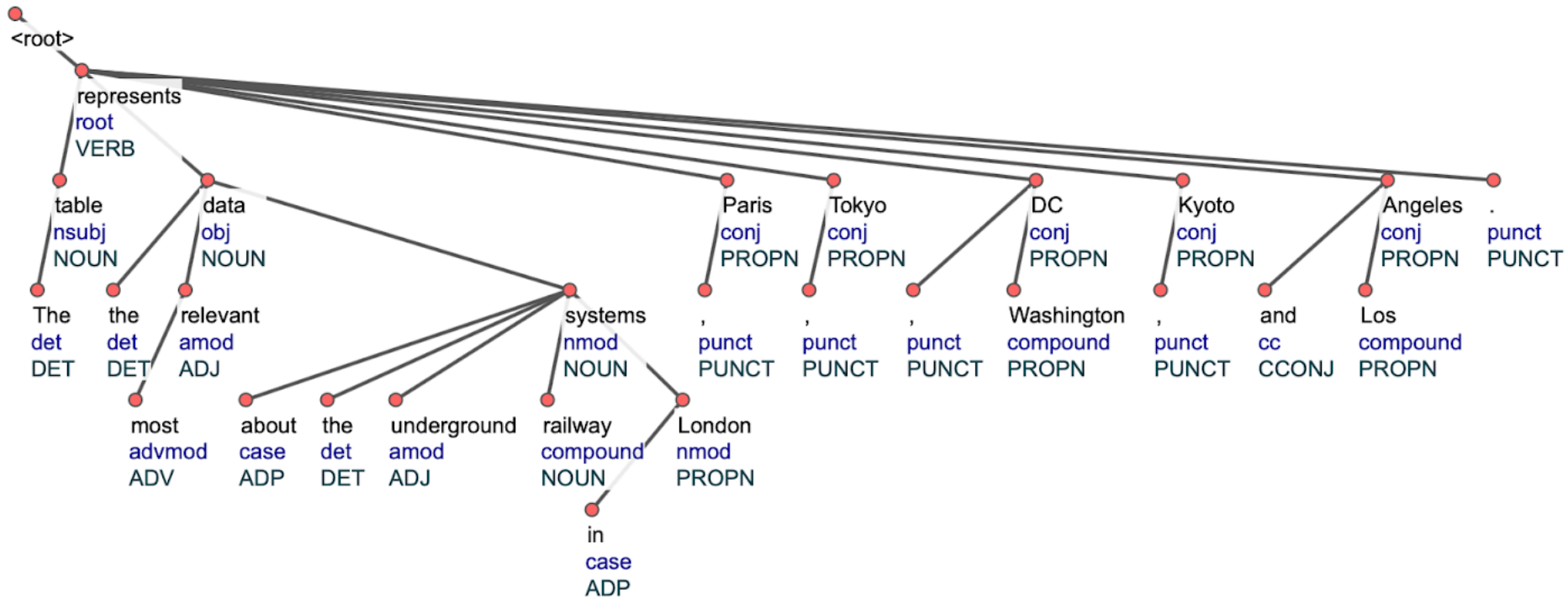
- в именных предложениях со вспомогательными глаголами be вершинами считает существительные/прилагательные, а spacy -- вспомогательный глагол (подумать, что лучше, мне пока кажется, что глагол, тк полезно для моделей с согласованием)

- пытается различать **have-main verb и have-auxilary,** получается плохо, это портит вершины, тк auxilary он в вершину не ставит

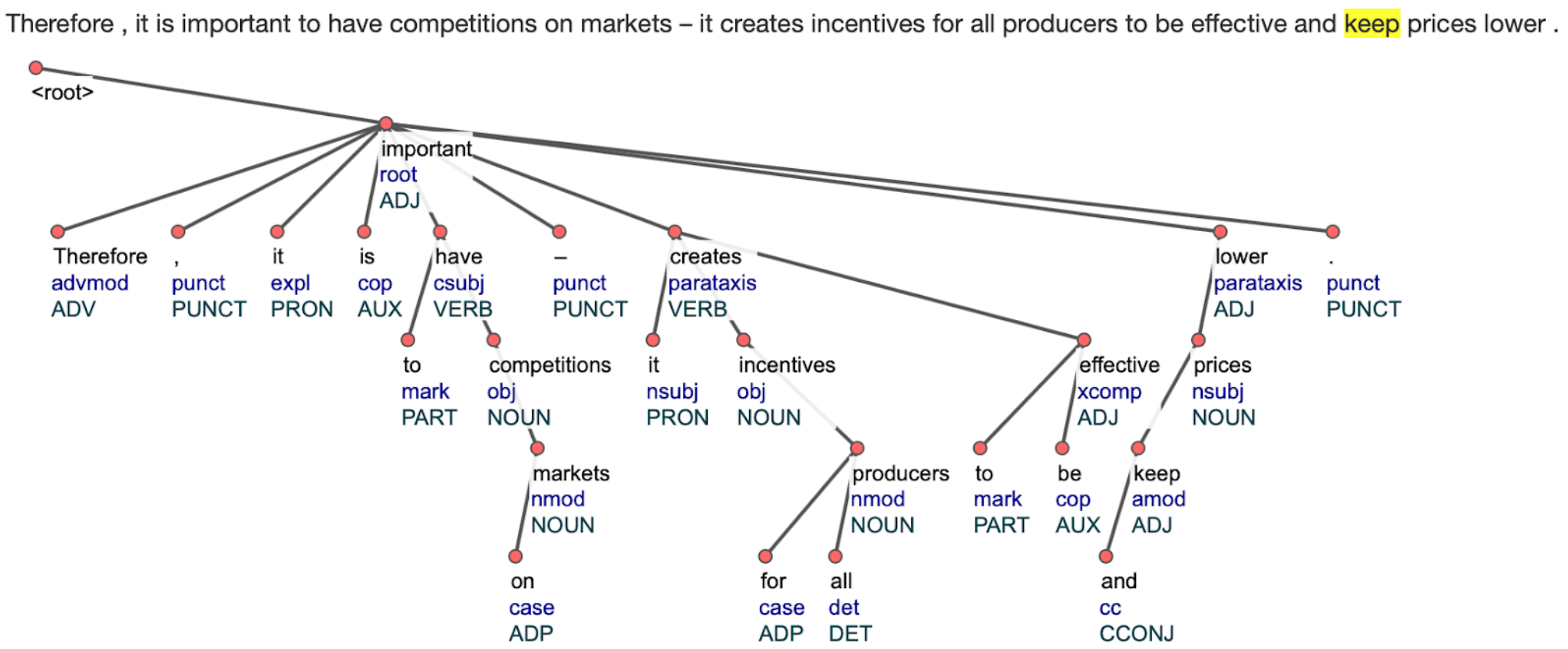
# UDPipe, пример плохого деления на однородные члены



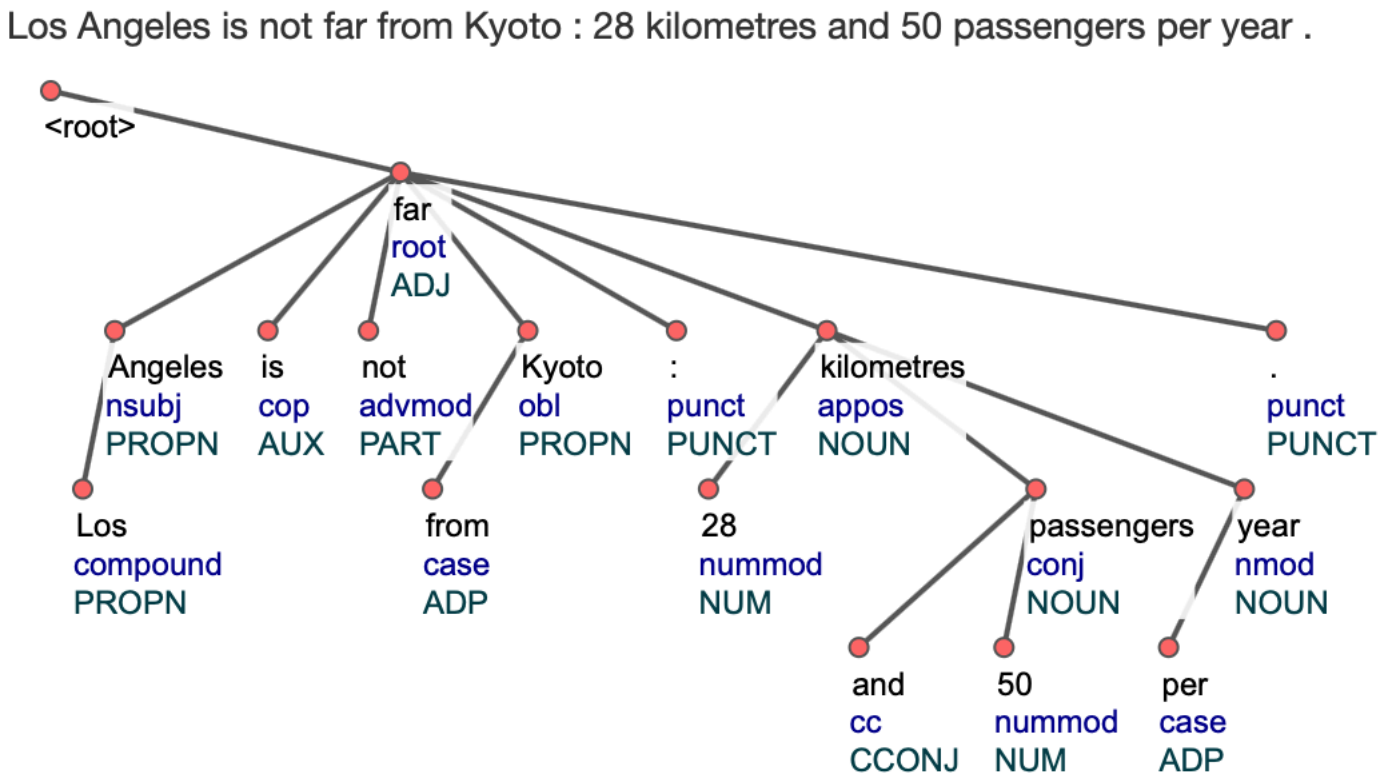
The table represents the most relevant data about the underground railway systems in London , Paris , Tokyo , Washington DC , Kyoto and Los Angeles .



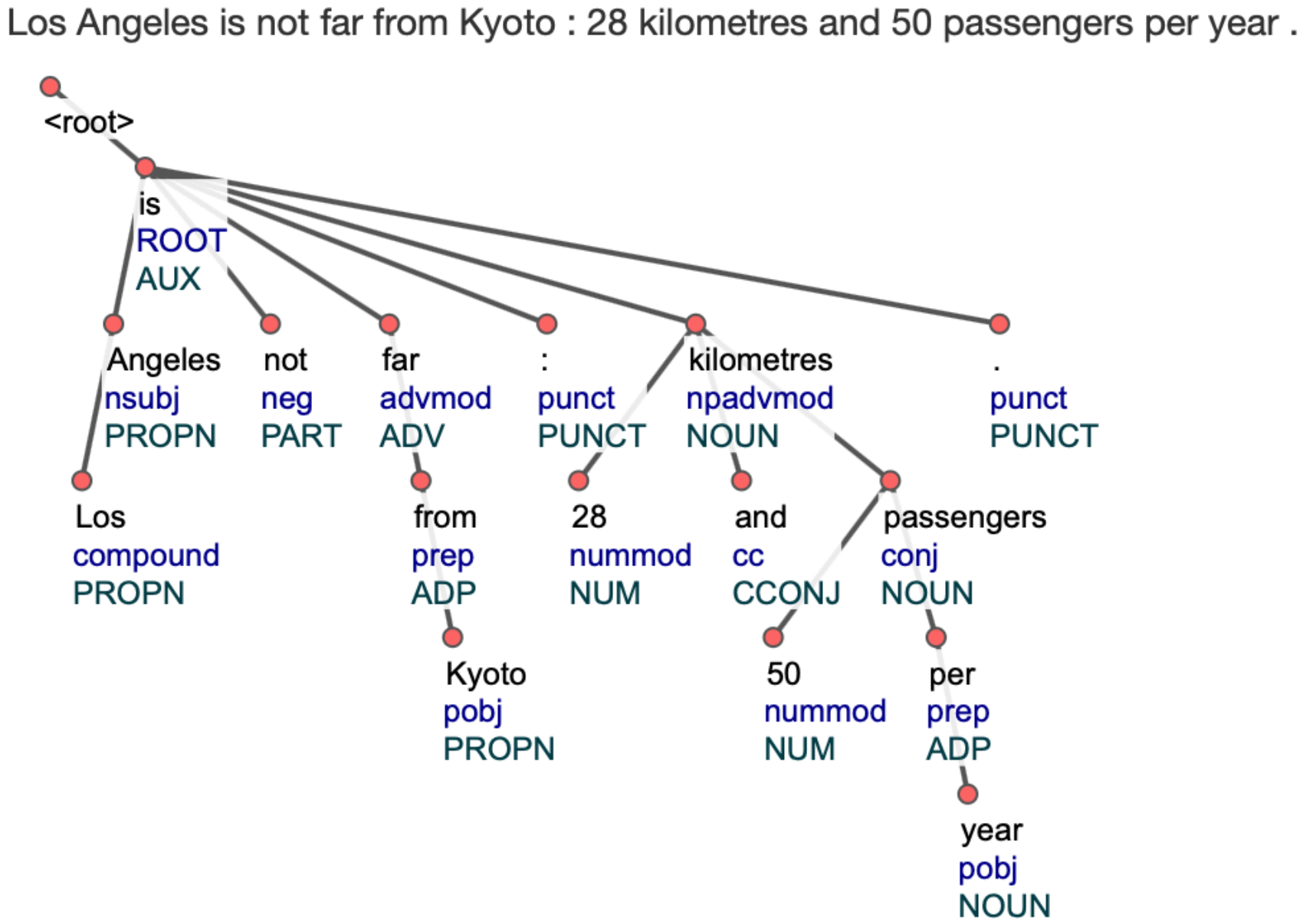
# Udpipe, пример, обратим внимание на правую часть изображения



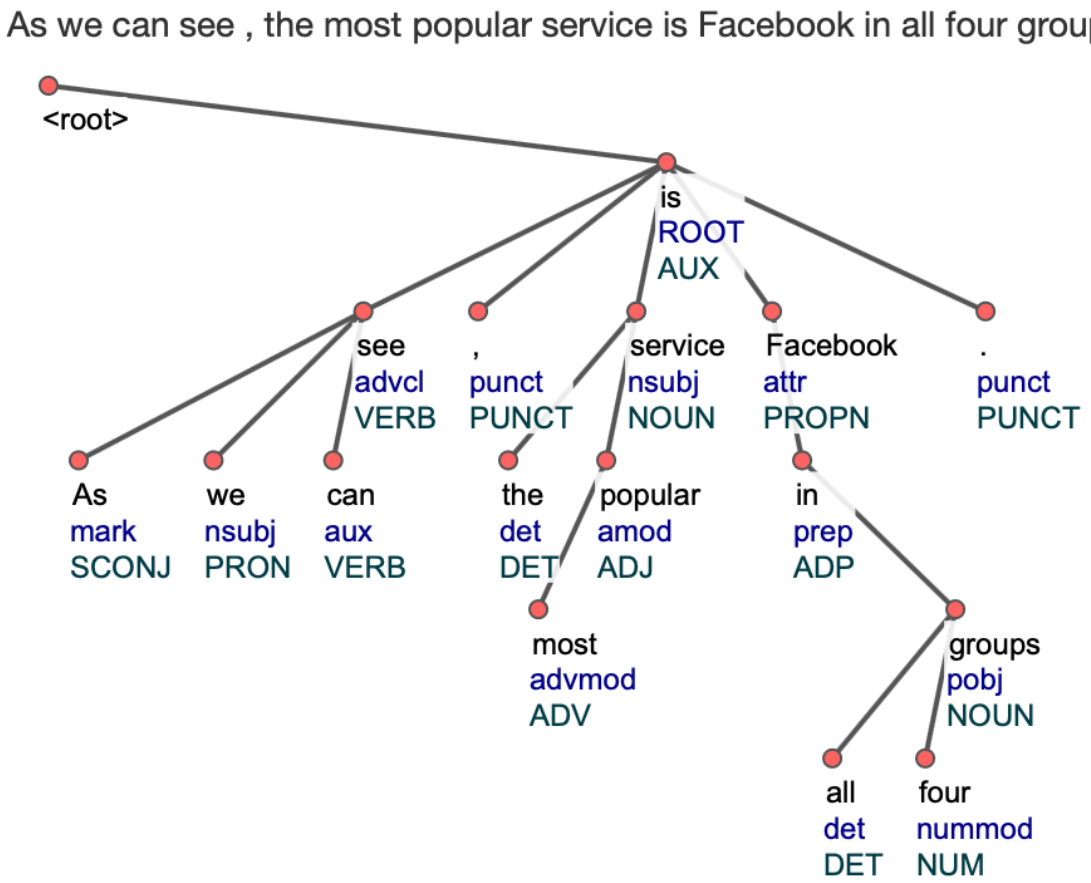
# Пример с вершинами в предложениях со вспомогательным в UDPipe:



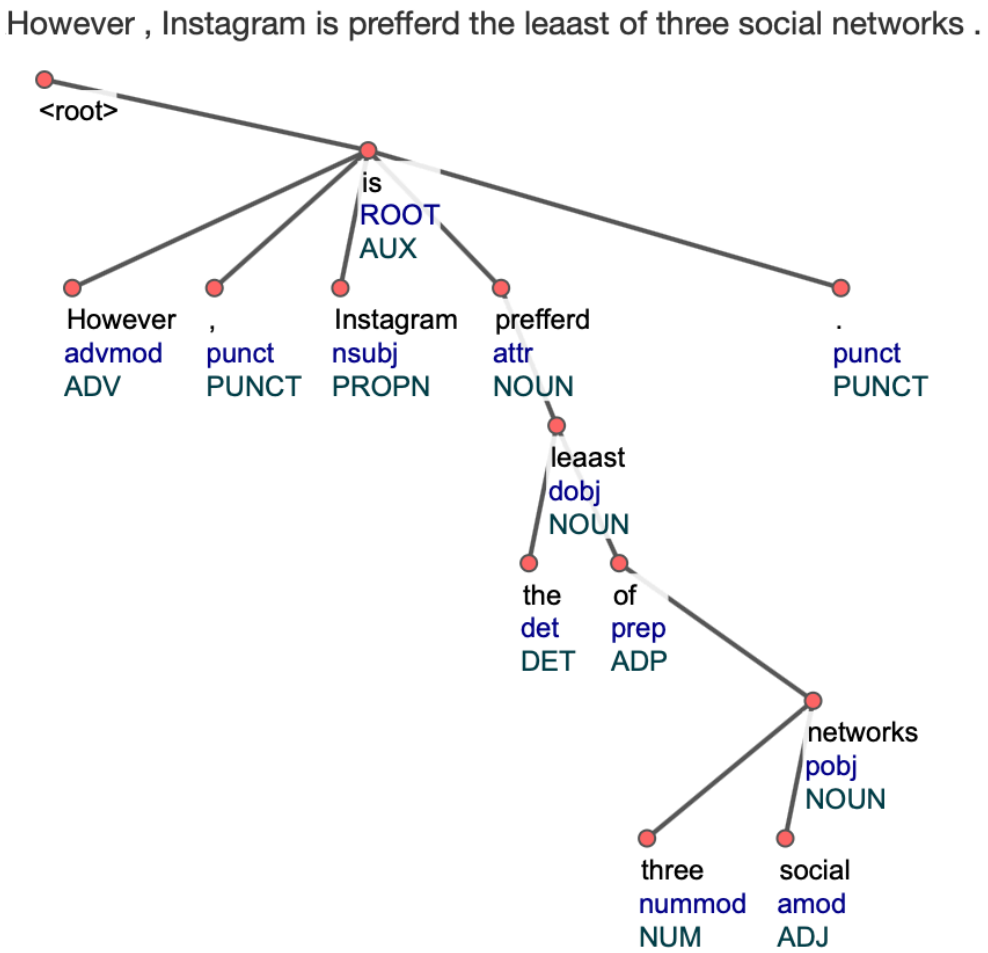
В SpaCy: проблема заключается в том, что так происходит не всегда



# Например, (Spacy)не справляется в сочетании вспомогательный глагол _ смысловой глагол



# А с тем, что он определяет как ИГ, все хорошо с нахождением вершины клаузы



# UDPipe, вывод:

- Как и у Spacy, у него проблемы с определением,  как глаголы зависят от друг друга
- Не может делить предложения на нормальные клаузы
- Вообще через библиотеку stanfordnlp нельзя делать необходимую синтаксическую разметку. У Stanford NLP есть клиент CoreNLP, который работает через Java и делает все необходимые операции. Питоновскую библиотеку можно использовать как интерфейс для работы с CoreNLP, но на компьютере все равно должна быть установлена Java, через которую, собственно, все операции и будут производиться.
- Помимо всего, CoreNLP работает как сервер, поэтому операции получаются следующим образом: код в питоне дает команды для Java -> Java запускает CoreNLP на сервер -> CoreNLP делает операции на сервере -> отдает результат Java -> Java отдает результат питону. Поэтому этот парсер работает медленнее, чем другие.
- О том, как работать с этим всем, **не очень много написано**. Есть материалы об использовании CoreNLP на Java, но не понятно, как это транслируется, когда CoreNLP используется через Python. Когда я (Миша) разбирался, как получить из CoreNLP деревья, я добился этого, грубо говоря, методом тыка.

- Conllu файлы получаются без строк #sent_id и #text (как и у Spacy, но это достаточно просто решается)

# Stanford NLP (Михаил Папоротский)

**Использование, техническая сторона:**

- Вообще через библиотеку stanfordnlp нельзя делать необходимую синтаксическую разметку. У Stanford NLP есть клиент CoreNLP, который работает через Java и делает все необходимые операции. Питоновскую библиотеку можно использовать как интерфейс для работы с CoreNLP, но на компьютере все равно **должна быть установлена Java**, через которую, собственно, все операции и будут производиться.
- Помимо всего, CoreNLP работает как сервер, поэтому операции получаются следующим образом: код в питоне дает команды для Java -> Java запускает CoreNLP на сервер -> CoreNLP делает операции на сервере -> отдает результат Java -> Java отдает результат питону. Поэтому этот **парсер работает медленнее, чем другие.**
- О том, как работать с этим всем, **не очень много написано**. Есть материалы об использовании CoreNLP на Java, но не понятно, как это транслируется, когда CoreNLP используется через Python. Когда я (Миша) разбирался, как получить из CoreNLP деревья, я добился этого, грубо говоря, методом тыка.
- +/- Conllu файлы получаются без строк #sent_id и #text **(т.е. дополнительно требуется время (хоть и незначительное) для «подгонку» к формата .conllu)**

# Stanford NLP (Михаил Папоротский)

**Результаты разметки, плюсы и минусы:**

+ Неплохо справляется с однородными членами
- Как и у Spacy, у него проблемы с определением,  как глаголы зависят от друг друга

- Не может делить предложения на нормальные клаузы

# Сравнение одного предложения

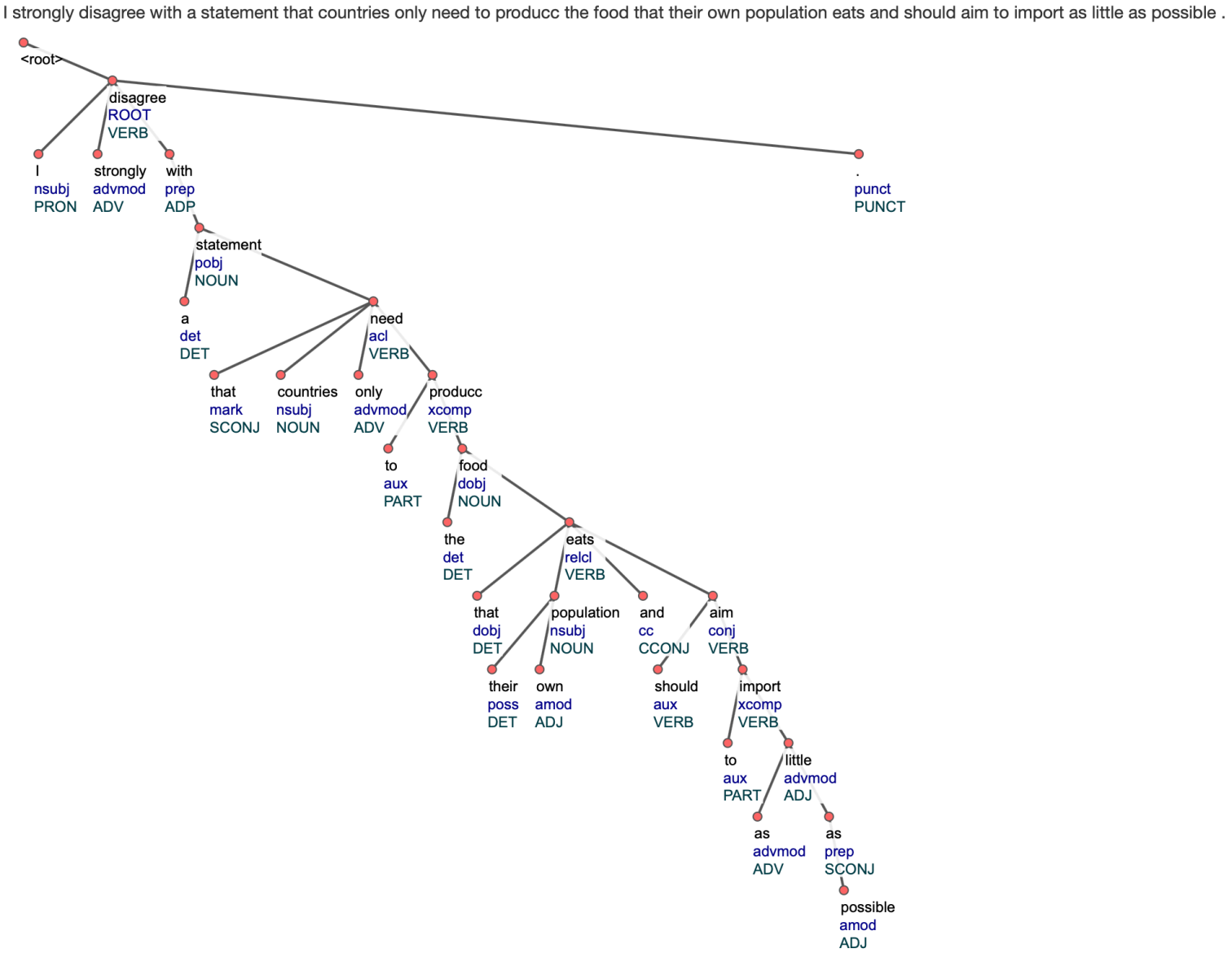
- **Предложение:**
  *I strongly disagree with a statement that countries only need to* ***producc*** *the food that their own population eats and should aim to import as little as possible .*

- **Важно обратить внимание на то, как парсеры:**
  - делят предложение на клаузы
  - находят вершины этих клауз
  - определяют отношения между клаузами
  - определяют отношения внутри клауз
  - справляются с опечатками/ошибками в тексте

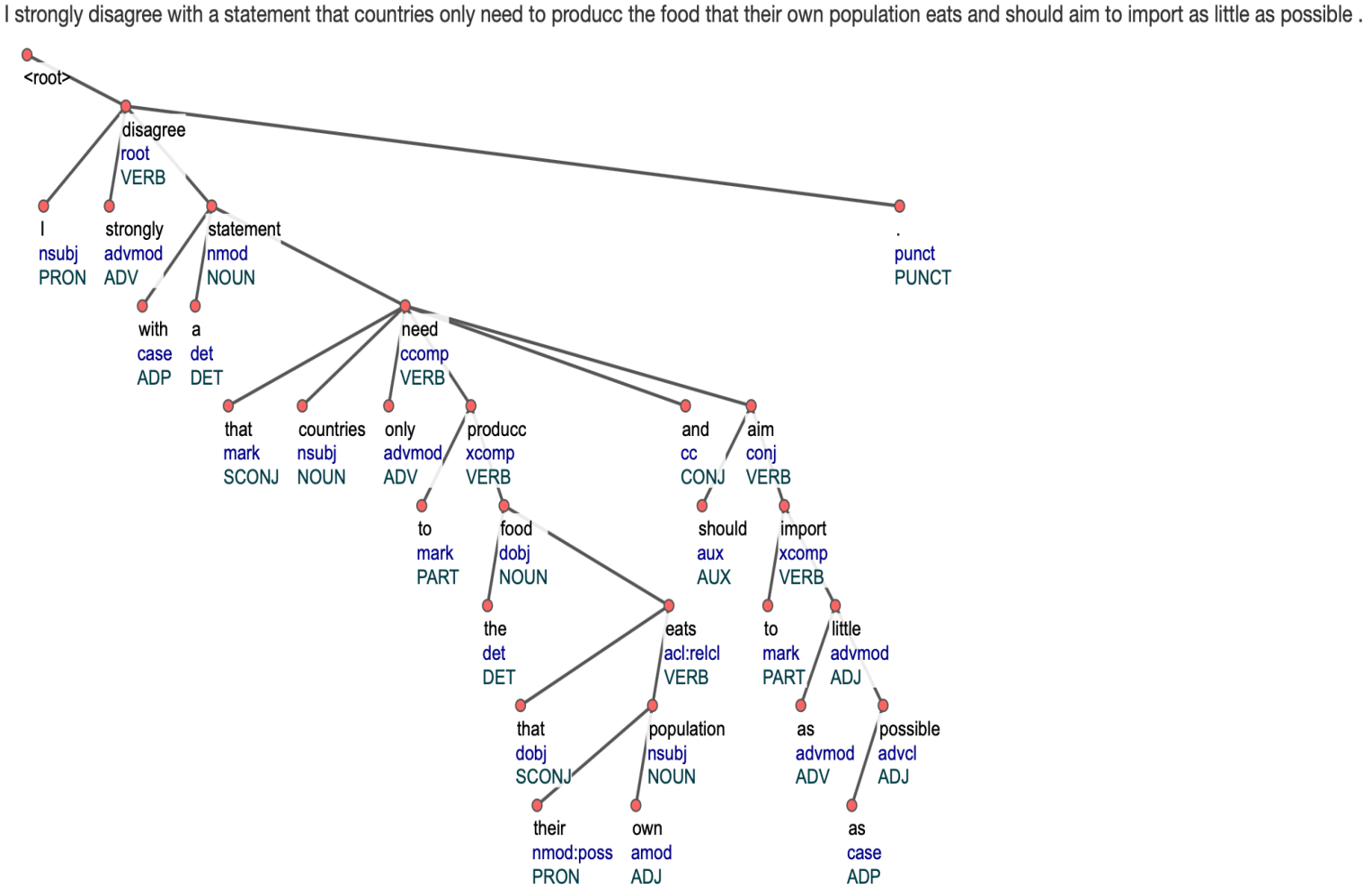
# Spacy, lab_2.conllu

+ Правильно определяются сами клаузы
- зависимости клауз между собой
- а также их вершины не всегда выделены корректно.

+ Несмотря на это, ошибка в слове *product* не помешала понять парсеру синтаксическую роль и часть речи этого слова.



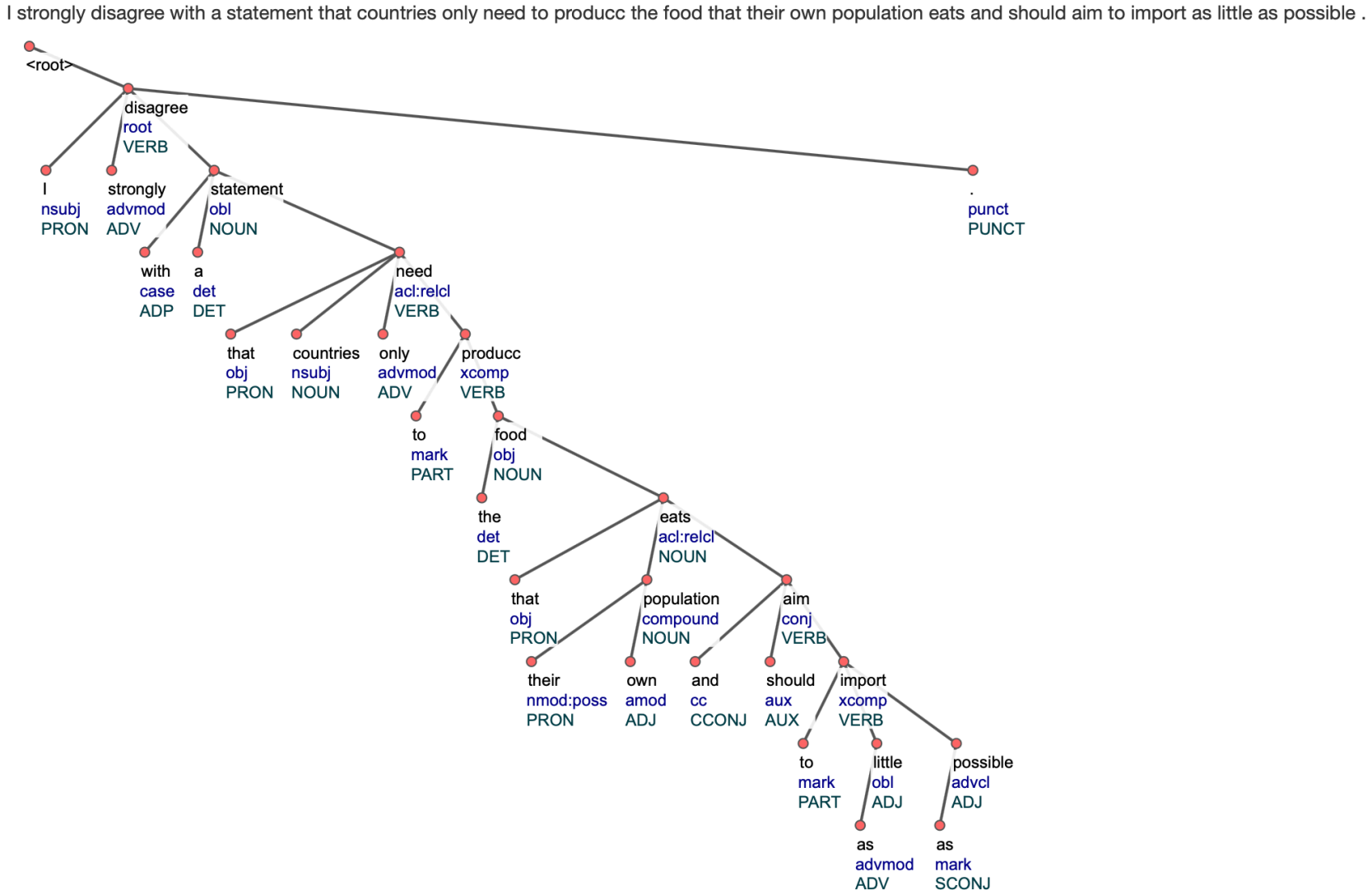
# Stadford NLP, text1_NLP.conllu.txt




+ Клаузы поделены правильно
+ зависимости клауз найдены верно (в отличие от Spacy)
+ хорошо определяет однородные члены в данном предложении
- но неправильные связи выделяет между ними
- Вершины найдены не всегда корректно

# UDPipe (Ника Смилга)

+ понимает, что такое *producc* на самом деле
+ правильно определяет здесь вершины клаузы
- неверно определяет однородные члены



# NTLK (Антон Бузанов)

+ можно выбирать таггер

+ можно вручную контролировать создание групп (прописывать какие-то паттерны)

- нужно писать руками _все_ вариации

- просто комбинаторика, чтобы перебрать все слова в предложении

- грамматика составляющих, а не зависимостей

- не понятно, в каком порядке работают регулярные выражения → результат непредсказуем

**мини-вывод:**

- Работать практически невозможно, так как готовых баз выражений нет в интернете (на гитхабе). Все другие таггеры смотрятся более выгодным решением.

# Выводы

- Поскольку ни один парсер не может корректно выделить вершину клаузы, если сказуемое выражено аналитически, поэтому логично будет смотреть на согласование подлежащего и сказуемого проверкой на наличие вспомогательного глагола: при его отсутствии сравниваем подлежащее непосредственно с вершиной, при наличии – подлежащее с «зависимым» от вершины вспомогательным глаголом.
- Смотреть на «однородность» в предложениях лучше в Spacy
- Проблему «сохранения в .conllu-формат» можно решить автоматической разметкой на Python за незначительное время