Report

2022.06.23. Wed.

Ringle 개발팀 NLP 인턴 인터뷰 과제

표현 megatwins@kaist.ac.kr

1. How to Run

pip install nltk

pip install pandas

python main.py

Data in data/

Output generated in output/

python 3.9.12

nltk version 3.7

2. Problem

다음과 같이 4가지 문장 종류를 정의한 후에 단순한 것부터 sub-problem 들을 정의하고 순서대로 함수를 작성했다. 그 다음 알고리즘을 검증하는 과정과 running time을 개선할 방법에 대해 고민했다. 4개 문장 구조의 정의는 wikipedia(https://www.wikipedia.org/)의 정의를 따랐으며 cambridge dictionary(https://dictionary.cambridge.org/)의 clause 에 대한 정의를 참고했다.

Definitions

Simple sentence : A sentence with one independent clause and no dependent clauses.

Compound sentence : A sentence with at least two independent clauses and no dependent clauses.

Complex sentence : A sentence with at least one independent clause and one or more dependent clauses.

Compound-complex sentence : A sentence with at least two independent clauses and one or more dependent clauses.

Clause : A group of words, consisting of a subject and a finite form of a verb

Dependent clause : A clause that cannot form a separate sentence but can form a sentence when joined with a main clause.

Independent clause : A clause in a sentence that would form a complete sentence by itself.

Main Problem

임의의 English 문장을 simple, complex, compound, compound complex로 labeling할 수 있는 알고리즘을 만드시오.

1. 문장의 clause 개수를 센다.
2. 문장의 dependent clause 개수를 센다.
3. 문장의 independent clause 개수를 센다.
4. 문장을 simple, complex, compound, compound complex로 labeling한다.

3. Solution

1) **문장의 clause 개수를 센다.** (`extract_nsubjs.py`)

문장에서 clause 와 대응되는 feature를 찾아내려고 했고, clause의 정의를 근거로 주어와 동사 쌍을 feature로 정했다. 이를 위해 (1) 4가지 구조로 labeling 된 예시 문장들 (`data/data.txt`)의 dependency parsing 결과를 그래프로 나타내고 관찰했으며 (2) universal dependency document 를 공부했다. Dependency parsing 은 NLTK book을 공부한 후 문장에서 주어를 추출할 수 있는 가장 간단한 방법이라고 생각해 사용했다.

nsubj relation이 clause의 동사와 주어로 이루어진다는 점을 이용했다. 문장의 dependency parse tree의 *nsubj* relation 개수가 문장의 clause 개수와 같다는 가정 하에 함수를 작성했다. 첫 작성에서 이 가정이 7개 문장 parse 결과를 관찰한 것을 근거로 했었기 때문에 이 가정이 유효한지 확인하는 것을 task로 메모해두었다. 이후에 universal dependencies document를 공부해서 이 부분을 예외가 없도록 수정했다.

Parser로는 Stanford dependency parser를 사용했다.

2) **문장의 dependent clause 개수를 센다.** (`extract_clauses.py`)

3) **문장의 dependent clause 및 independent clause의 개수를 각각 센다.** (`extract_clauses.py`)

dependency parsing 결과에서 메인 clause가 아닌 clause 는 하나의 sub-tree를 구성한다. 새로운 clause에 대응하는 sub-tree의 root 노드가 기존 메인 clause와 연결되는 relation의 종류를 통해 dependent clause인지 independent clause인지를 구분할 수 있다.

Universal dependency document 를 근거로 다음과 같이 각 clause의 root 를 dependent 로 가질 수 있는 relation 의 리스트를 만들었다. dep 의 경우에는

class of clause	dependent clause	independent clause
relation	advcl, ccomp, dep	conj, dep

Sub-problem 1을 해결한 방법대로 각 relation의 dependent 중에 *nsubj* relation을 갖는 dependent만 새로운 clause 의 root 노드로 인식하도록 했다. Dependency graph 의 노드를 traverse 하며 이 과정을 적용했다.

4) **(Main problem) 문장을 simple, complex, compound, compound complex로 labeling한다.** (`sentence_structure_classifier.py`)

문장의 dependent clause 및 independent clause의 개수와 문장 구조의 정의에 따라 문장을 4가지로 labeling 했다. 적은 데이터에 대해서는 불필요할 수 있지만, 입력 데이터의 크기가 많거나 구조가 복잡할 때 코드 수정이 편리하도록 데이터와 라벨을 pandas dataframe 으로 다뤘다.

5) ***dep* relation**

dep relation 은 parser가 정확히 labeling 할 수 없는 relation이다. dependency graph에 따라 *dep* relation은 dependent clause로도, independent clause로도 이어진다.

dep relation으로 시작하는 sub-tree가 independent clause에 대응할 때는 의문문을 구성하는 경우이다. 따라서 *nsubj* relation 의 dependent(주어)보다 앞선 동사 태그(VB~ or MD) 토큰이 있으면 의문문이므로 independent clause로, 그렇지 않으면 dependent clause 로 분류했다.

Reference

Wikipedia. Sentence clause structure.
https://en.wikipedia.org/wiki/Sentence_clause_structure

Cambridge Dictionary. Clause, Independent clause, Dependent clause.

https://dictionary.cambridge.org/dictionary/english/clause

https://dictionary.cambridge.org/dictionary/english/independent-clause

https://dictionary.cambridge.org/dictionary/english/dependent-clause

Universal Dependencies contributors. Universal Dependencies. https://universaldependencies.org/u/overview/complex-syntax.html#ellipsis-in-clause-coordination

Contact

(+82) 10 -3088 -1516

megatwins@kaist.ac.kr