Tetrapod Documentation

목차

1. 이 어플리케이션의 목적
2. 이 어플리케이션의 구동 원리
3. 구성 요소
4. Utils.js 설명
5. Tetrapod.js 설명
6. 이 어플리케이션의 목적

이 어플리케이션은 저장된 비속어 데이터를 이용해서 문장 속의 비속어 데이터를 찾아내는 어플리케이션입니다. JavsScript 기반으로 작성되어 있습니다.

1. 이 어플리케이션의 구동 원리

기본적으로 문장에서 비속어를 작성할 때는 단순히 비속어 표기법 그대로 사용하지 않습니다. 따라서 이 어플리케이션에서는 지정된 비속어 단어를 낱자 단위로 분해한 뒤에 낱자들이 문장에 적절히 근접하게 있다면 비속어가 있는 것으로 판단하고 비속어를 잡아냅니다. 단순히 비속어가 순서대로 배열된 경우뿐만 아니라 순서가 어긋난 상태도 찾을 수 있습니다.

II.1 설치 방법

1. 우선 Node.js와 git을 설치해야 합니다.
2. 그 다음에 <https://github.com/utolee90/tetrapod>에 접속하신 뒤 자신이 다운로드 받고 싶은 디렉토리에 콘솔창을 열고   
   > git clone <https://github.com/utolee90/tetrapod.git> 으로 입력하시면 tetrapod 디렉토리에 이 어플리케이션이 설치됩니다.
3. Git이 설치되어 있지 않은 경우 소스를 직접 다운로드 받을 수도 있습니다.
4. 설치한 뒤에 npm install –save-dev라고 입력하시면 관련 확장 패키지들이 설치됩니다. 이제 이 어플리케이션을 사용할 준비가 되었습니다.

II.2 사용방법

1. 관련 패키지들을 설치한 뒤에 ./test/find.js에서 원하시는 기능을 실행시킬 수 있습니다.
2. 호출할 때에는 import Tetrapod from ‘../tetrapod’; 유틸리티 사용할 때에는 import Utils from ‘../components/Utils’;를 입력하셔야 합니다.
3. 테스트를 실행하려면 tetrapod가 있는 디렉토리를 연 뒤에 콘솔창에 npm run test라고 입력하시면 find.js에 있는 요소들이 실행됩니다.

II.3. 구동 원리

1. Tetrapod 소스의 nativeFind 함수가 비속어 목록에서 비속어를 찾는데 핵심적인 함수입니다. 이 함수의 구동 방식에 대해 설명합니다.
2. 구성 요소

이 어플리케이션은 다음과 같은 파일들로 구성되어 있습니다.

1. tetrapod.js -> 메인 어플리케이션으로 비속어 필터링에 필요한 함수들을 저장하고 있습니다.
2. Components 디렉토리 – tetrapod에서 사용하는 유틸리티들을 저장한 디렉토리입니다.
3. Bias 디렉토리(삭제 예정) – 반복적인 리스트를 이용해서 비속어 패턴을 작성할 때 사용합니다. Tetrapod 자체적으로도 재귀적 방식으로 비속어 리스트를 지원하고, 많은 유사패턴을 만드는 것보다 유사패턴은 함수를 이용해서 변환하는 방식으로 바꿀 예정이라 사용하지 않을 예정입니다.

III.1. Components 유틸리티

* JavaScript 확장기능 Hangul에 의존합니다. 아래 두 함수를 사용합니다.
  + Hangul.disassemble(msg, grouped=false) -> 한글을 자모단위로 리스트로 분해합니다. Grouped=true로 지정되면 글자 단위로 리스트를 분해한 후 다시 각 글자마다 자모단위로 분해하는 2중 리스트로 표현됩니다.

(예: 가상 => (grouped가 거짓이면 [‘ㄱ’,’ㅏ’,’ㅅ’,’ㅏ’,’ㅇ’], 참이면 [[‘ㄱ’,’ㅏ’],[‘ㅅ’,’ㅏ’,’ㅇ’]]

* + Hangul.assemble(list) -> 자모단위로 분해된 리스트를 재조합해서 한글로 된 문자열을 조합합니다. (예: [‘ㄱ’, ㅏ’, ’ㄴ’, ‘ㅎ’, ‘ㅗ’, ‘ㅅ’, ‘ㅏ’] => 간호사)
* HangulObject.js -> Utils.js와 tetrapod.js에서 사용될 오브젝트를 모았습니다.
* ObjectOperation.js -> JavaScript는 Python에 비해 오브젝트의 포함관계를 판단하는 것이 어렵습니다. 여기서는 자주 사용하는 오브젝트 비교 함수들을 제작해서 모았습니다.
  + objectEqual(a,b) => 두 오브젝트 a,b 가 같은지 확인하는 함수
  + objectInclude(inc, exc, order=false) => inc 오브젝트가 exc 오브젝트 안에 들어가는지 확인하는 함수. Order 옵션이 true이면 inc 리스트의 순서도 exc가 보존하는지 여부도 확인합니다.
  + removeMultiple(list) => 리스트 상의 중복 원소를 제거합니다. 순서를 보존합니다.
  + joinMap(lix, func) => 오브젝트/리스트와 콜백을 합성할 때 사용하는 함수입니다.
  + addList (…List) =>리스트의 원소를 더합니다. 각 리스트는 (원소1,원소2,…) 형태로 원소를 풀어써서 입력하거나 …기호를 이용해서 리스트를 풀어써서 입력합니다. 원소가 숫자면 값을, 문자면 문자열을 연결하고, 리스트면 concat함수를 이용해 뒤에 원소가 추가된 형태로 값을 출력합니다.
  + productList(…list) => 입력값은 리스트 원소들이며, 리스트의 카테시안 곱을 유도할 때 사용됩니다. 예를 들면 productList([1,2,3],[4,5,6],[7,8])로 입력할 경우 첫 번째 리스트에서 원소 1개, 두 번째 리스트에서 원소 1개, 세 번째 리스트에서 원소 1개를 뽑은 결과인 [[1,4,7], [1,4,8], [1,5,7], …] 형식으로 결과가 출력됩니다.
  + objectIn(elem, object) => elem 오브젝트가 object 오브젝트의 원소인지 판단하는 함수입니다.
  + listUnion(…List) => 리스트들을 입력값으로 한 다음에 합집합을 구할 때 사용하는 함수입니다. 입력할 때에는 (리스트1, 리스트2, …) 형태로 합집합을 구할 리스트들을 풀어쓰거나 단일 리스트일 경우 …기호를 추가해서 원소를 풀어쓰는 형식으로 처리하면 됩니다.
  + listIntersection(…list) => 리스트들을 입력값으로 한 뒤에 교집합을 구할 때 사용하는 함수입니다. 입력할 때에는 (리스트1, 리스트2, …) 형태로 합집합을 구할 리스트들을 풀어쓰거나 단일 리스트일 경우 …기호를 추가해서 원소를 풀어쓰는 형식으로 처리하면 됩니다.
  + listDisjoint(a,b) = > 두 리스트가 서로소인지 판단합니다.
  + listDifference(a,b) =>두 리스트의 차집합을 구할 때 사용합니다. 원소 a에 포함되어 있으면서 b에는 포함되지 않은 원소들을 확인합니다.
  + filterList(list, type) => list 안에 type에 해당하는 원소를 걸러냅니다. Type는 특정 타입을 가리키는 문자열 또는 그런 문자열을 포함하는 리스트일 수 있습니다.
* Utils.js – tetrapod.js에서 다용도로 사용하는 유틸리티 함수들을 정리한 오브젝트입니다. 편의상 HangulObjects.js의 오브젝트들과 ObjectOperation.js의 모든 함수들을 Utils 오브젝트의 하위 함수 형태로 사용할 수 있게 디자인됐습니다.
  + replaceAll(message, search, replace) => 메시지의 search로 된 정규표현식 패턴을 찾아서 replace로 바꿔주는 함수입니다.
  + getPositionAll(message, search isString=true) =>

이 어플리케이션은 hmmhmmhmm의 원래 소스를 바탕으로 utolee90이 제작했습니다.