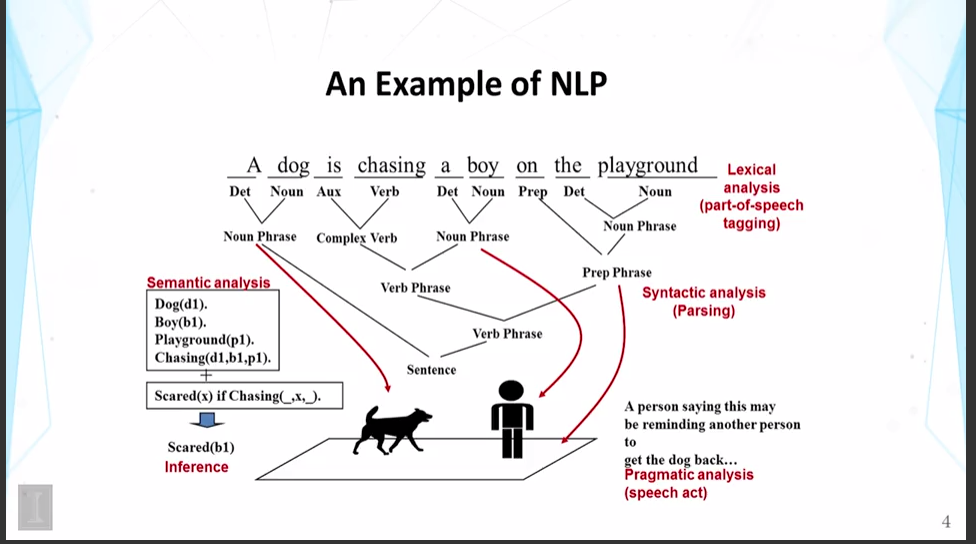
# NLP의 예



Lexical analysis(=품사태깅)

* 첫 번째 단계임

syntactic analysis

* 문장 구조를 파악
* 이 단계에서는 구조만 파악할 뿐, 의미 확인은 불가

semantic analysis

* 각 구조를 표현하는 symbol을 사용해서 의미 추론을 함
* ex. chasing(dog, boy, playground) = scared
  + dog, boy, playground를 chasing이라는 관계로 묶으면 scared가 나옴

pragmatic analysis

* 더 나아가, 왜 이 말을 했는지에 이해할 수 있음
* ex> 위 문장을 말한 이유가 다른 사람에게 개을 가져가라는 것을 상기시켜 주기 위해서 일 수 있음
* 이러한 경험적 추론을 하는 것을 의미함

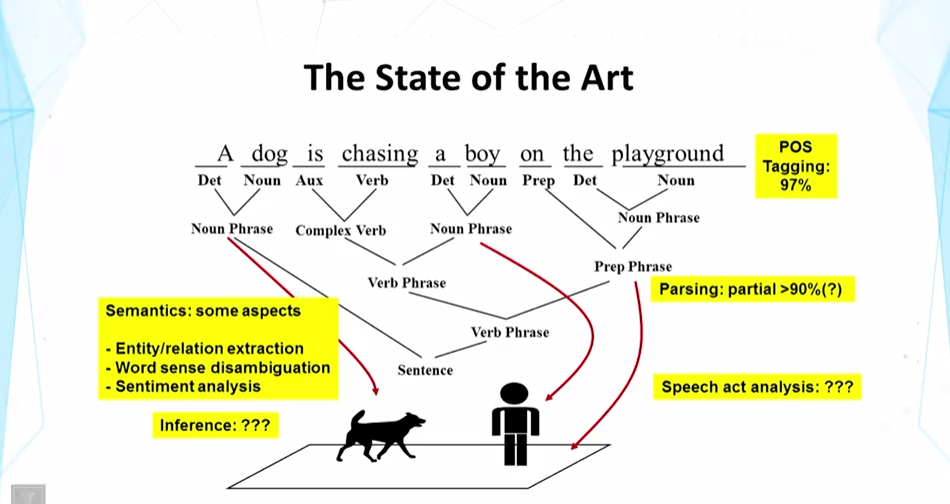
# NLP가 어려운 이유

* 자연어는 인간이 의사소통하기 위해서 만들어진 것임
* 자연어는 청자/독자가 가지고 있는 상식적 내용을 생략함
* 의미상 불분명한 것은 청자 독자가 알아서 해석함

# NLP를 위해 해결해야할 문제들

* 단어 수준의 모호성을 어떻게 할 것인가
  + design은 명사이면서 동사이다.(모호한 품사)
  + root는 여러 의미를 가지고 있다.(모호한 의미)
* 문장구조적 모호성
  + natural language processing에서 natural이 수식하는 구문이 language인지, language processing인지 모호함(자연어처리 / 언어처리는 자연스럽다라고 해석이 가능한..)
  + a man saw a boy with a telescope(telescope를 가진 주체는 누구인가? man인가 boy인가?)
    - 이런 경우를 prepositional phrase attachment ambiguity라고 함
* 대명사 문제
  + John persuaded Bill to buy a TV for himself(himself는 John인가 Bill인가?)
* 추측
  + He has quit smoking(그는 이전에 흡연을 했다는 것을 추론해야한다)

# 현재 NLP 방식



* 품사 태깅 정확성은 97%가량
* 구문 분석 정확성은 90% 이상(dataset에 따라 다를 수 있음, 뉴스로 만들어진 dataset이라면 더 높을 수도 있지만, 소셜미디어 기반일 경우 낮을 수도 있다.)
* semantic 분석에서는, Entity/관계 추출, 단어 의미 모호성 해소, 감정 분석 등을 함. 정확성은 인공지능 측면에서 다루는 문제

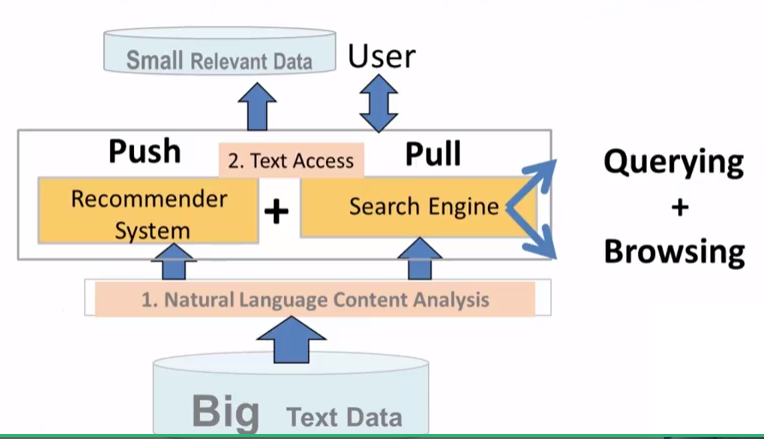
# NLP에서 할 수 없는 것

* 100% 정확한 품사 태깅
* Shallow understanding vs deep understanding
  + shallow : 품사태깅, 구문분석, 감정 인식(문장의 완전한 이해가 아니므로 deep이 아님)
  + deep : 문장에 대한 완전한 이해, 일반적으로 쉽게 발전하지 않고, 잘 동작하지 않음. 일반적으로 기계학습 기반에서 그나마 잘 동작함

# TR을 위한 NLP

* Shallow NLP, 가볍고 효율적으로 동작해야 함.
* 대부분의 검색작업에서 단어셋만으로도 충분한 경향이 있다(대신 단어의 순서가 무시되므로, 단어의 순서가 중요한 경우는 문제가 될 수 있음)

# 관련 있는 텍스트 데이터에 접근하는 방법



### Push vs Pull

Pull : 검색 엔진, User가 주체

Push : 추천 시스템, 시스템이 주체

### Querying vs Browsing

Quering : 사용자가 직접 키워드를 날림, 시스템은 해당 키워드와 관련 있는 문서들을 리턴함. 사용자가 자신이 필요한 것에 대한 키워드를 알 때 잘 동작함

Browsing : 분류된 문서에서 사용자가 직접 필요한 정보를 찾아감. 사용자가 자신이 필요한 것에 대한 키워드를 알 지 못할 때 유용.

* 놀이동산을 간다고 가정하면, Quering은 택시타고 놀이동산으로 가주세요 하는것이고, Browsing은 저 쯤 가주세요 하고 다시 저 근처 가주세요 하면서 점차 찾아가는 과정에 비유할 수 있음

TR vs Database Retrieval

|  |  |
| --- | --- |
| TR | DR |
| Unstructured/free text | structured data |
| Ambiguous Imformation/Query | well defined Imformation/Query |
| incomplete Query | complete specification |
| return relevant document | matched records |
| -can’t mathematically prove one method is better than another  -rely on empirical evaluation involving users |  |

# Document Selection vs ranking

### Selection

* 연관 있는지 아닌지 딱 두가지로만 판단
* 두 가지 상태밖에 없기 때문에, 기준점을 어떻게 잡느냐에 따라, 아래와 같은 문제점이 있음
  + Over-constrained -> 관련 문서 없음
  + Under-constrained -> 관련 없는 문서 선택
* 분류가 정확하더라도, 관련 있는 문서들끼리는 또 동등한 값이기 때문에 어떤 것이 더 정확한지에 대해 결정하기 어려움

### Ranking

* 관련 함수에 의해 연관도 높은 순으로 정렬
* 두 가지 가정이 있음
  + document 유용성에 대해서는 서로 영향을 미치지 않음
  + 사용자는 ranking 결과를 순차적으로 확인함

# Ranking 함수는 어떻게 디자인 할 것인가?

유사성 기반 모델

* VSM

확률 기반 모델

확률적 추론 모델

공리적 모델

TF(Term Frequency) : 문서에서 단어가 몇 번 출현했는가?

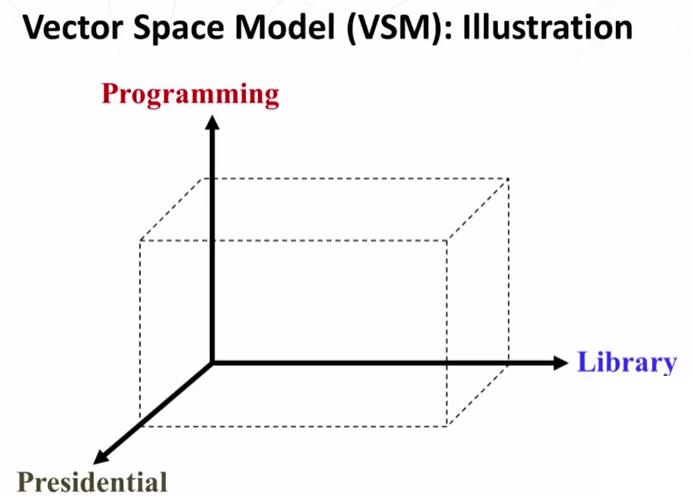
Document length : 문서 길이

DF(Document Frequency) : 단어가 출현한 문서는 몇 개인가?

### 어떤 모델이 제일 좋을까? 아래 4개는 일반적으로 비슷하게 잘 동작함

* Pivoted length normalization
* BM25(가장 흔히 사용되는 방법)
* Query likelihood
* PL2

# VSM



* 문서를 벡터화 함
* VSM은 query/document를 어떻게 vector화 할지에 대한 방법을 제공해주지 않음
* 단지 문서의 특정 단어를 뽑아 벡터로 만드는 방법일 뿐임.
* 이 벡터를 가지고 얼마나 유사한가에 대한 판단 방법은 다른 문제임

Bit Vector

* 출현 단어는 1로, 출현하지 않으면 0으로 표기
* ex> (Programming, Library, Presidentail)의 벡터의 경우 document = “My Library is cool library”인 경우는 (0, 1, 0)이 됨

TF Vector

* 출현한 횟수만큼 가중치를 부여
* ex>(Programming, Library, Presidentail)의 벡터의 경우 document = “My Library is cool library”인 경우는 (0, 2, 0)이 됨