DEEP LEARNING for Search

Tommaso Teofili

Foreword by Chris Mattmann

박 경 규

Contents

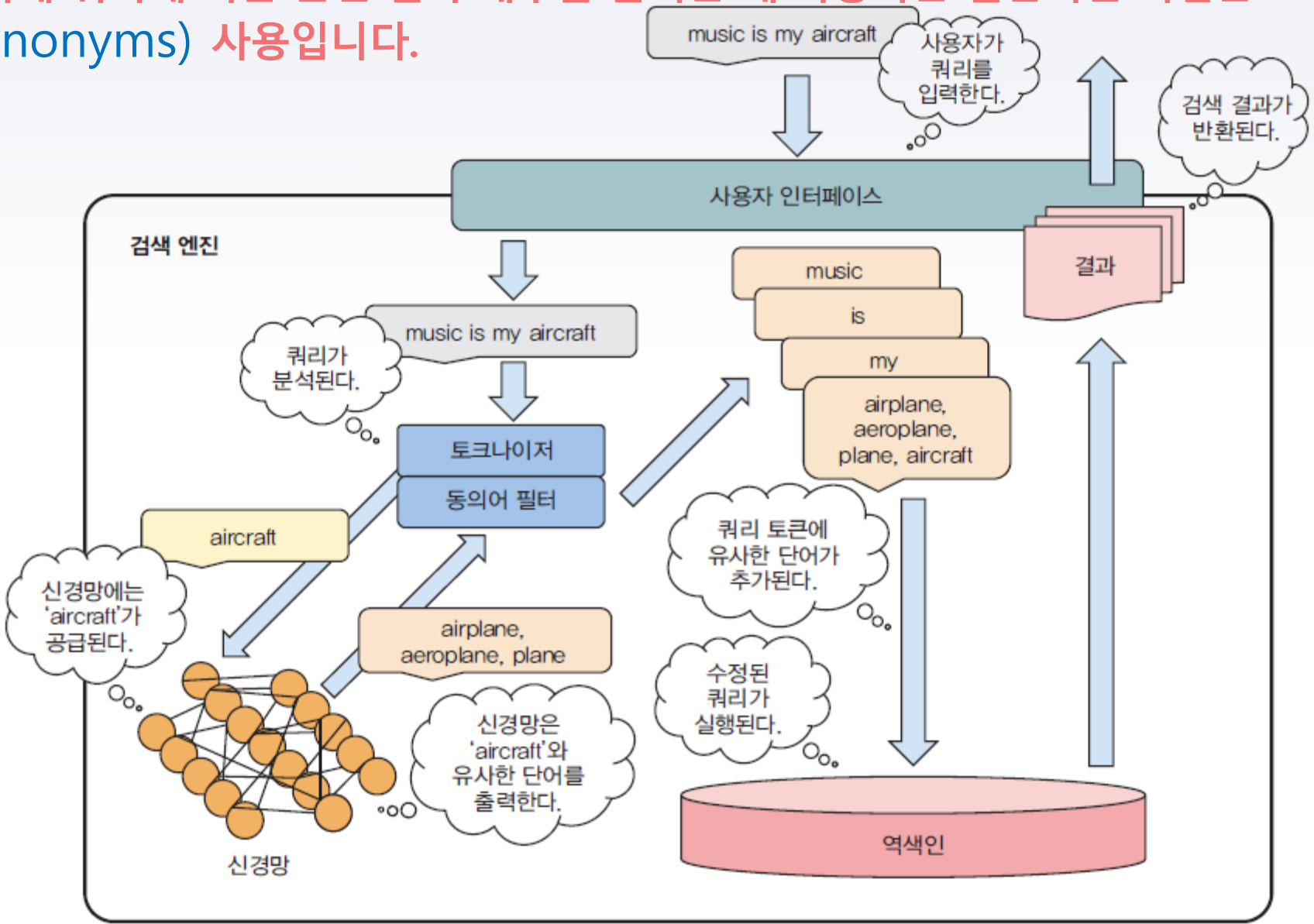
1. Neural search
2. Generating synonyms
3. From plain retrieval to text generation
4. More-sensitive query suggestions
5. Ranking search results with word embeddings
6. Document embeddings for rankings and recommendations
7. Searching across languages
8. Image contents and search
9. A peek at performance

2. Generating synonyms
(동의의 생성)

- 검색시 동의어가 사용되는 이유 및 방법
- 아파치 루씬 소개
- Word2vec을 사용해 동의어 생성

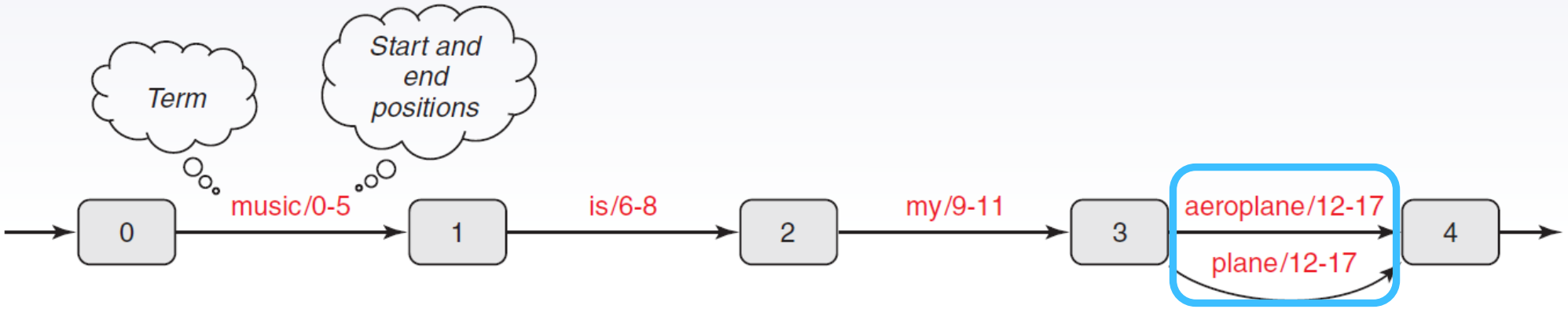
동의어 확장

정보 검색 시에 쿼리에 따른 관련 결과 개수를 늘리는 데 사용하는 일반적인 기법은
동의어(synonyms) **사용입니다.**



동의어 확장 그래프

검색엔진이 용어의 흐름을(stream of terms) 수신할 때 해당 용어와 같은 자리에 동의어를 추가함으로써 검색 결과를 늘릴 수 있다는 생각이 동의어 확장이라는 개념에 기본적으로 깔려 있습니다.




```
aeroplane -> plane, airplane, aircraft
boat -> ship, vessel
car -> automobile
```

Term	Document(position)
aeroplane	1(12,17)
aircraft	1(12,17)
airplane	1(12,17)
is	1(6,8)
music	1(0,5)
my	1(9,11)
plane	1(12,17)

PyLucene 설치

아파치 루씬은 자바로 작성된 오픈소스 검색 라이브러이며,
PyLucene은 Lucene에 액세스 하기 위한 Python 확장입니다.

- PyLucene : https://lucene.apache.org/pylucene/
  - PyLucene is a Python extension for accessing Java Lucene TM.
  - Its goal is to allow you to use Lucene's text indexing and searching capabilities from Python

- 아파치 ANT 설치 : http://ant.apache.org

- PyLucene 사용을 위한 jcc 빌드 : https://bit.ly/37VewUu

- PyLucene 설치 : https://lxsay.com/archives/365

- lupyne 설치 : pip install lupyne (https://pypi.org/project/lupyne/)

PyLucene, Lupyne 사용법

https://github.com/kgpark88/nlp/blob/main/pylucene_intro.ipynb

Lupyne is:

- high-level Pythonic search engine library, built on PyLucene
- RESTful JSON search server, built on CherryPy
- simple Python client for interacting with the server

In [ ]:
```
from lupyne import engine   # don't forget to call lucene.initVM

indexer = engine.Indexer()                                  # create an in-memory index (no filename supplied)
indexer.set('name', stored=True)                          # create stored 'name' field
indexer.set('text', engine.Field.Text)                      # create indexed 'text' field (the default)
indexer.add(name='sample', text='hello world')  # add a document to the index
indexer.commit()                                            # commit changes; document is now searchable
```

In [ ]:
```
hits = indexer.search('text:hello')              # run search and return sequence of documents
```

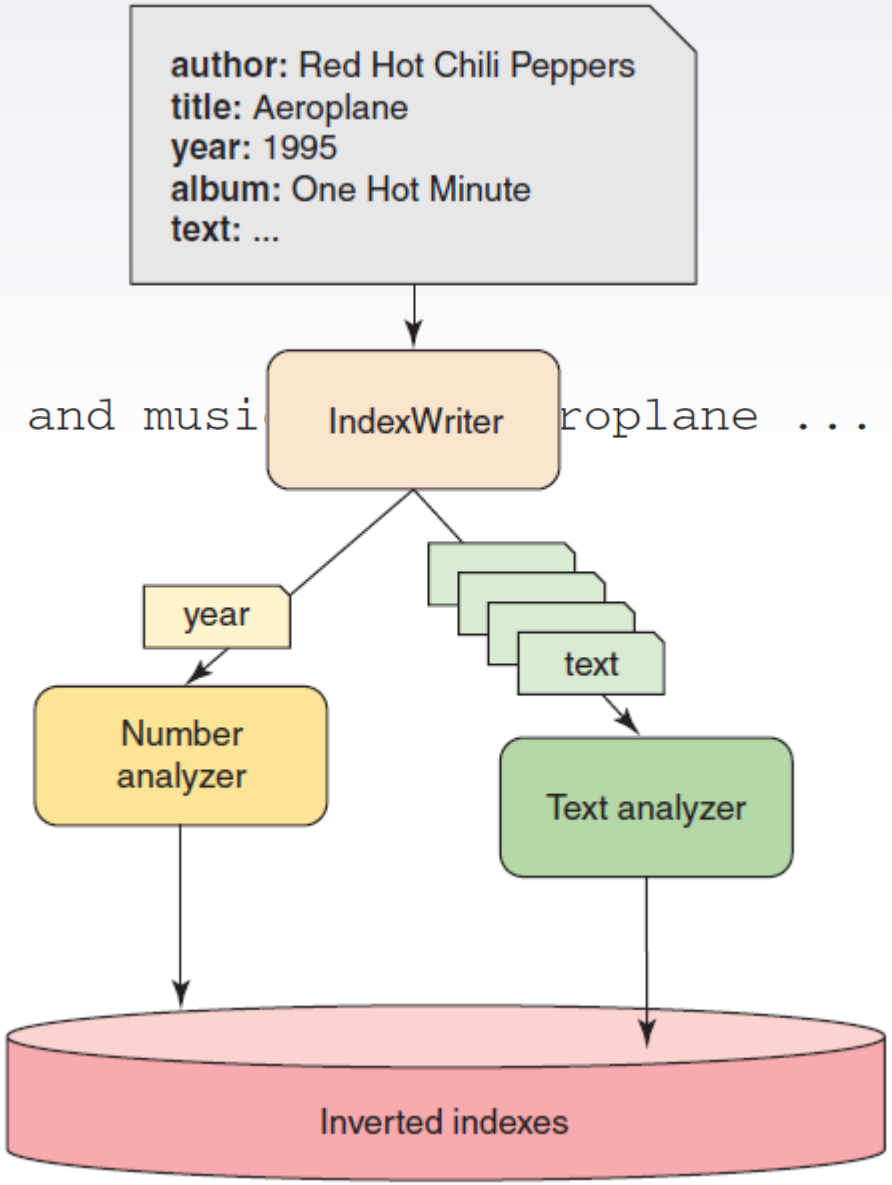
In [ ]:
```
hits
```

Out[ ]:
```
<lupyne.engine.documents.Hits at 0x21d682c2d48>
```

동의어 확장을 통해 루씬 색인 설정

- 음악 정보

```
author: Red Hot Chili Peppers
title: Aeroplane
year: 1995
album: One Hot Minute
text: I like pleasure spiked with pain and musi                roplane ...
```



① 필드별 분석기 구축

Listing 2.1 Building per-field analyzers

Sets up a map where the keys are the names of fields and the values are the Analyzers to be used for the fields

```
Map<String, Analyzer> perFieldAnalyzers = new HashMap<>();

CharArraySet stopWords = new CharArraySet(Arrays
    .asList("a", "an", "the"), true);

perFieldAnalyzers.put("pages", new StopAnalyzer(
   stopWords));

perFieldAnalyzers.put("title", new WhitespaceAnalyzer());

Analyzer analyzer = new PerFieldAnalyzerWrapper(
    new EnglishAnalyzer(), perFieldAnalyzers);
```

Creates a stopword list of the tokens to remove from the books' contents while indexing

Uses a StopAnalyzer with the given stopwords for the pages field

Uses a WhitespaceAnalyzer for the title field

Creates a per-field Analyzer, which also requires a default analyzer (EnglishAnalyzer, in this case) for any other field that may be added to a Document

② 루씬 색인에 문서 추가

Listing 2.2 Adding documents to the Lucene index

Creates a configuration for indexing

Creates an IndexWriter to write Documents into a Directory, based on an IndexWriterConfig

```
IndexWriterConfig config = new IndexWriterConfig(analyzer);
IndexWriter writer = new IndexWriter(directory,
    config);

Document dl4s = new Document();
dl4s.add(new TextField("title", "DL for search",
    Field.Store.YES));
dl4s.add(new TextField("page", "Living in the information age ...",
    Field.Store.YES));

Document rs = new Document();
rs.add(new TextField("title", "Relevant search", Field.Store.YES));
rs.add(new TextField("page", "Getting a search engine to behave ...",
    Field.Store.YES));

writer.addDocument(dl4s);
writer.addDocument(rs);
```

Creates Document instances

Adds Fields, each of which has a name, a value, and an option to store the value with the terms

Adds Documents to the search engine

③ 동의어 확장 구성

Listing 2.3 Configuring synonym expansion

```
SynonymMap.Builder builder = new SynonymMap.Builder();
builder.add(new CharsRef("aeroplane"), new CharsRef("plane"), true);
final SynonymMap map = builder.build();

Analyzer indexTimeAnalyzer = new Analyzer() {
  @Override
  protected TokenStreamComponents createComponents(
        String fieldName) {
    Tokenizer tokenizer = new WhitespaceTokenizer();
    SynonymGraphFilter synFilter = new
        SynonymGraphFilter(tokenizer, map, true);
    return new TokenStreamComponents(tokenizer, synFilter);
  }
};

Analyzer searchTimeAnalyzer = new WhitespaceAnalyzer();
```

Programmatically defines synonyms

Creates a custom Analyzer, for indexing

Creates a synonym filter that receives terms from the whitespace tokenizer and expands synonyms according to a map word, ignoring case

Whitespace analyzer for search time

④ 색인화 및 검색을 위한 별도의 분석 사슬

Listing 2.4 Separate analysis chains for indexing and search

```
Directory directory = FSDirectory.open(Paths.get(
    "/path/to/index"));

Map<String, Analyzer> perFieldAnalyzers =
    new HashMap<>();

perFieldAnalyzers.put("year",
    new KeywordAnalyzer());

Analyzer analyzer = new PerFieldAnalyzerWrapper(
    indexTimeAnalyzer, perFieldAnalyzers);

IndexWriterConfig config = new IndexWriterConfig(
    analyzer);

IndexWriter writer = new IndexWriter(
    directory, config);
```

Opens a Directory for indexing

Creates a map whose keys are the names of the fields and the values in the corresponding analysis chain to be used

Sets up a different analyzer (keyword; doesn't touch the value) for the year

Creates a wrapping analyzer that can work with per-field analyzers

Builds all the above in a configuration object

Creates an IndexWriter to be used for indexing

⑤ 문서 색인화

Listing 2.5 Indexing documents

Creates a document for the song "Aeroplane"

Adds all the fields from the song lyrics

```
Document aeroplaneDoc = new Document();
aeroplaneDoc.add(new Field("title", "Aeroplane", type));
aeroplaneDoc.add(new Field("author", "Red Hot Chili Peppers", type));
aeroplaneDoc.add(new Field("year", "1995", type));
aeroplaneDoc.add(new Field("album", "One Hot Minute", type));
aeroplaneDoc.add(new Field("text",
    "I like pleasure spiked with pain and music is my aeroplane ...", type));

writer.addDocument(aeroplaneDoc);
writer.commit();
```

Adds the document

Persists the updated inverted index to the filesystem, making the changes durable (and searchable)

⑥ ‘plane’ 단어 검색

Listing 2.6 Searching for the word “plane”

```
IndexReader reader = DirectoryReader.open(directory);

IndexSearcher searcher = new IndexSearcher(reader);

QueryParser parser = new QueryParser("text",
    searchTimeAnalyzer);

Query query = parser.parse("plane");

TopDocs hits = searcher.search(query, 10);
```

Opens a view on the index

Instantiates a searcher

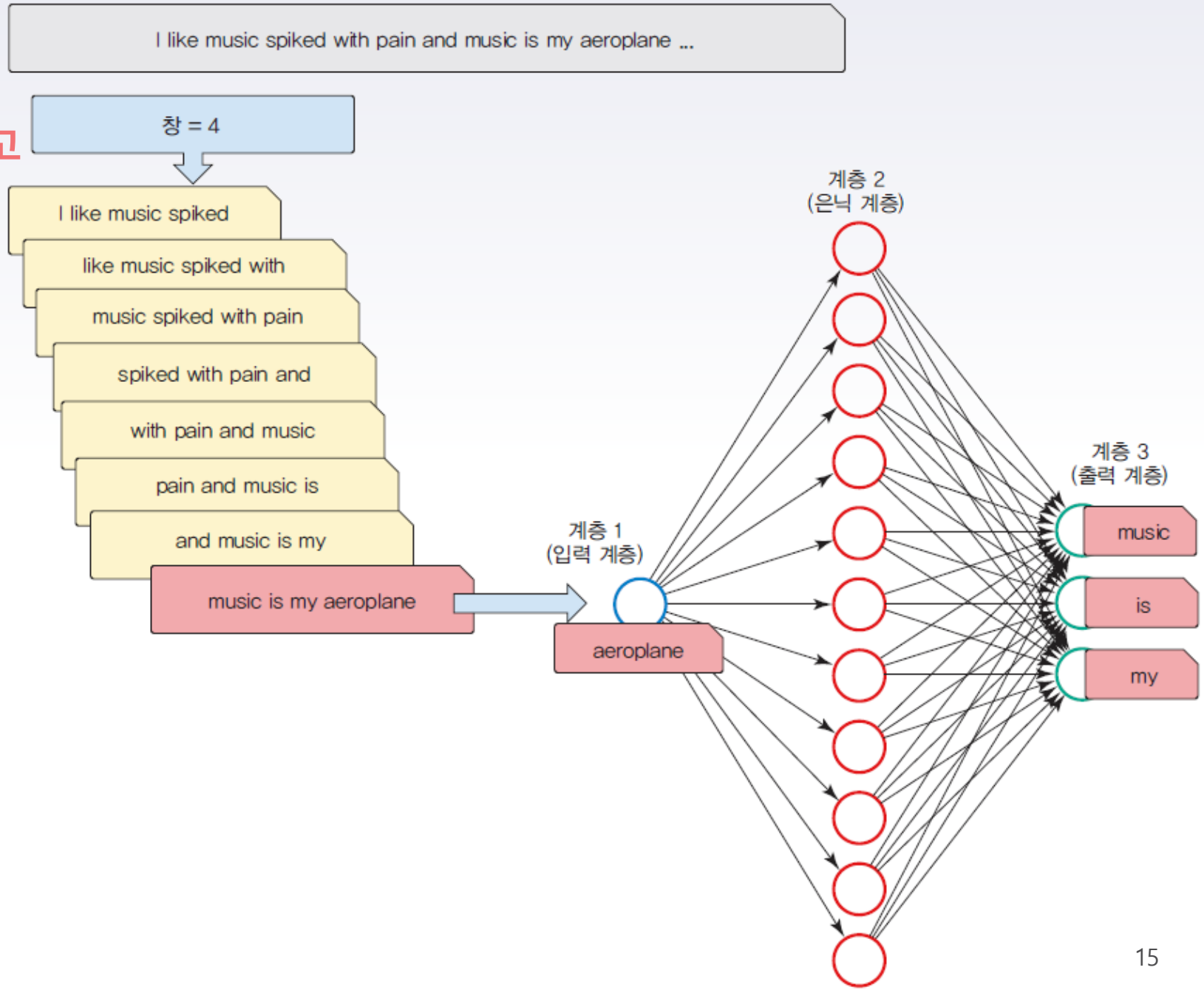
Creates a query parser that uses the search-time analyzer with the user-entered query to produce search terms

Transforms a user-entered query (as a String) into a proper Lucene query object using the QueryParser

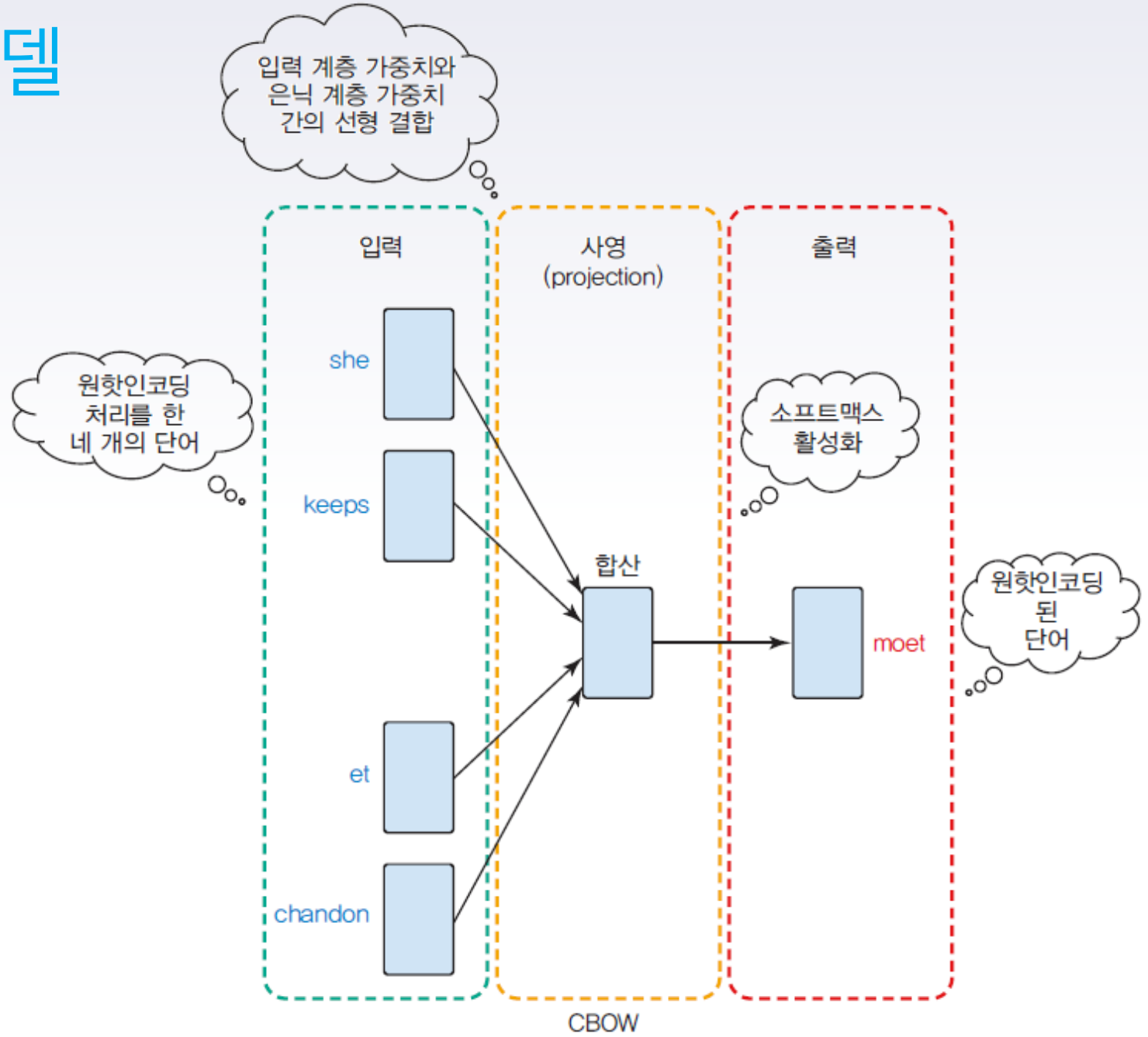
Searches, and obtains the first 10 results

word2vec 사용

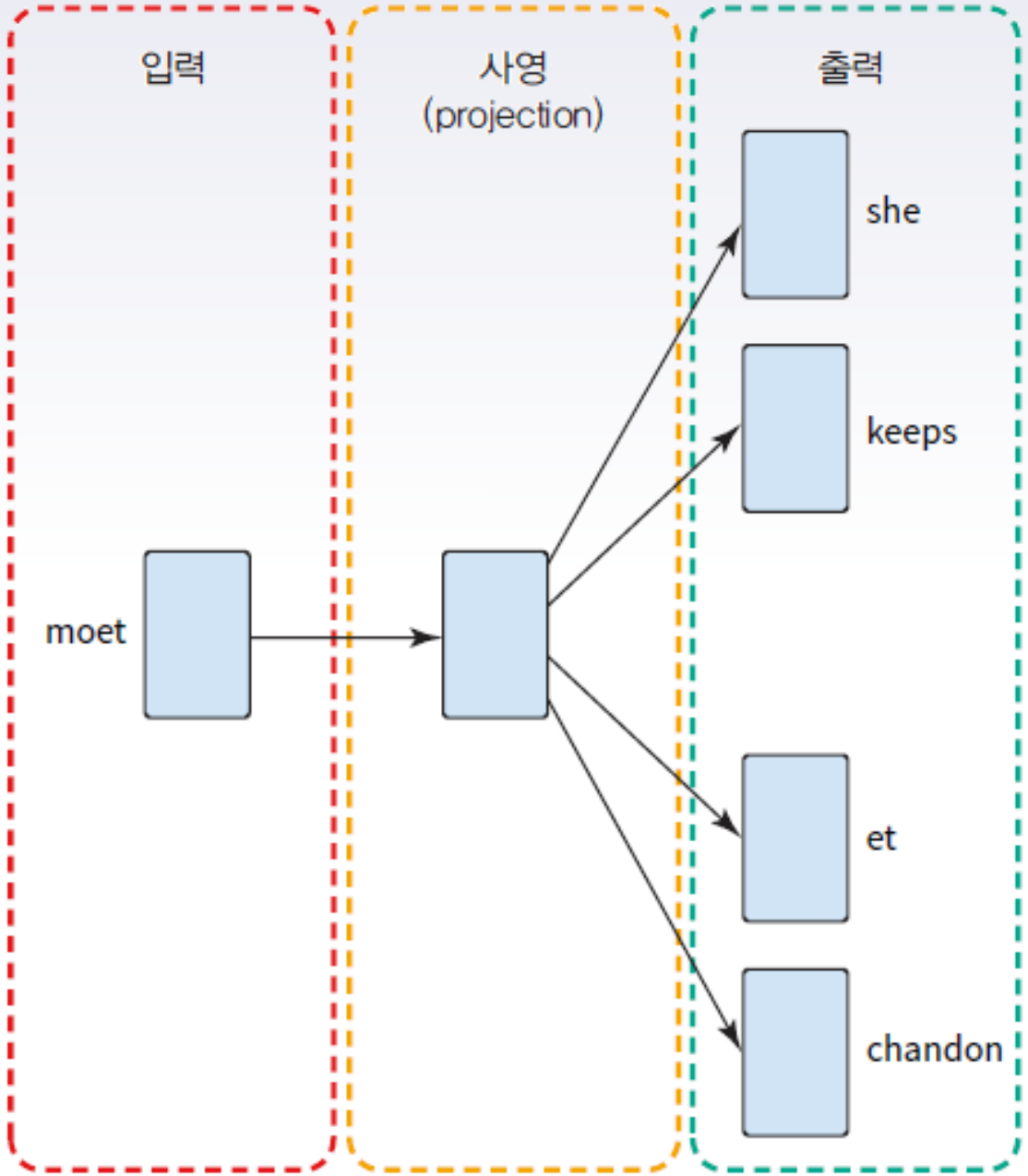
Word2vec 모델을 설정하고 색인화를 할 가사의 텍스트를 공급하고 각 단어에 대한 출력벡터를 구한 다음 동의어를 찾을 수 있습니다.



CBOW 모델

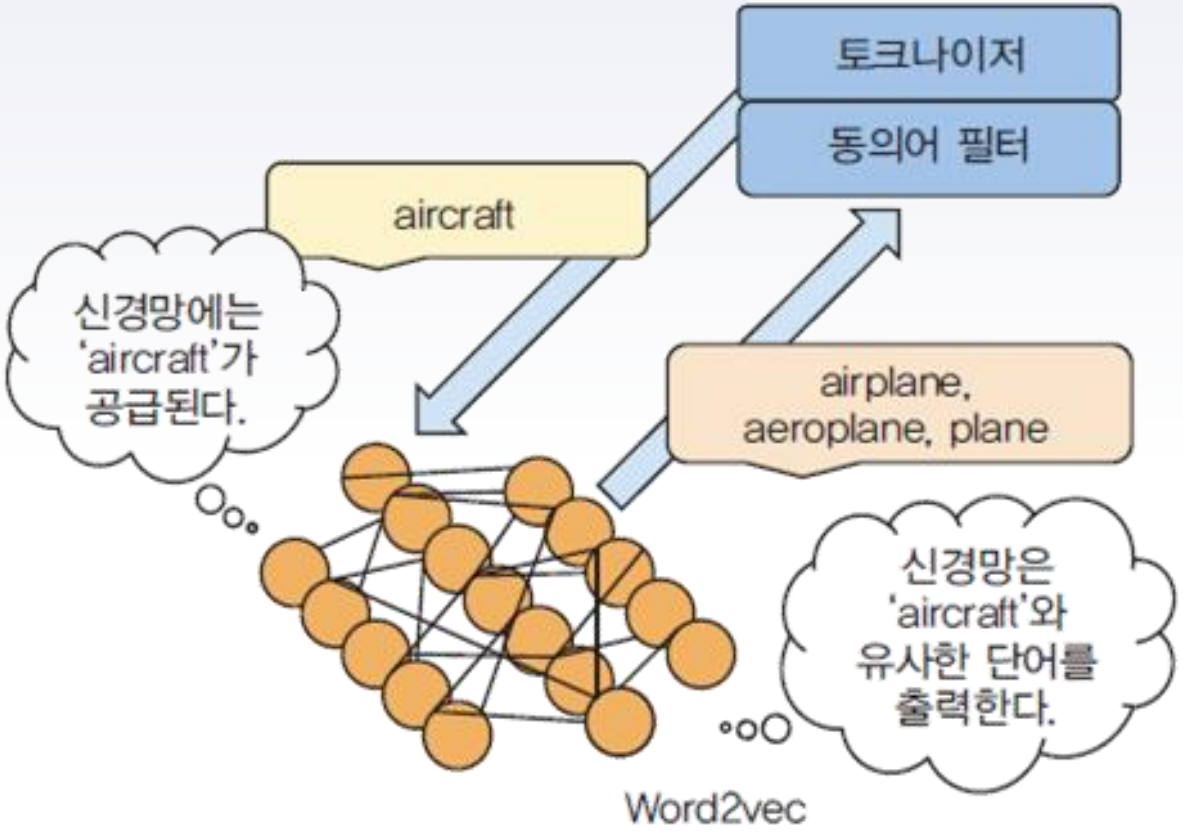

입력 계층 가중치와
은닉 계층 가중치
간의 선형 결합
입력
사영
(projection)
출력
원핫인코딩
처리를 한
네 개의 단어
she
keeps
et
chandon
합산
소프트맥스
활성화
moet
원핫인코딩
된
단어
CBOW

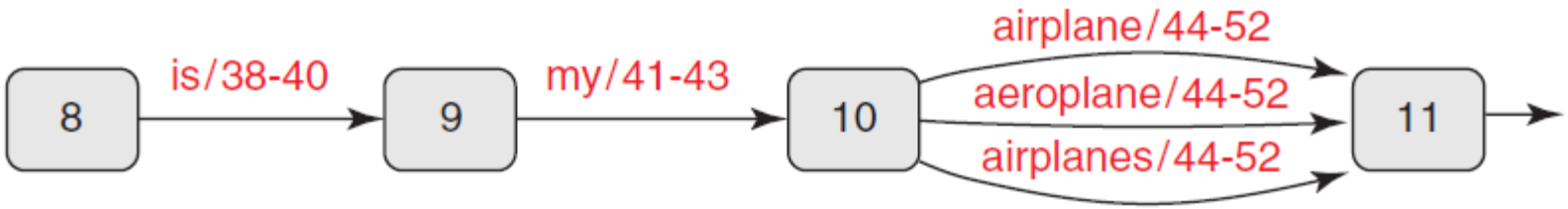
Skip Gram 모델



빌보드 word2vec 벡터

Word2vec 기반 동의의 확장





Summary

- 동의어(synonyms)를 확장하면 재현율(recall)이 개선되므로, 검색 엔진 사용자 만족도를 높일 수 있습니다.

- Word2vec은 유사한 의미의 단어를 찾거나 유사한 문맥에 나타나는 단어를 찾기 위해 사용할 수 있는 단어들에 대한 벡터 표현을 학습하는 순반향 신경망 기반 알고리즘이며 동의어 확장에 사용하는 것이 합리적입니다.

- Word2vec을 사용할 때는 반의어(antonyms)가 동의어로 여겨지지 않도록 주의하여야 합니다.

3. From plain retrieval to text generation

- 쿼리 확장
- 검색 로그를 사용해 훈련 데이터 작성
- RNN을 사용한 대안 쿼리 생성

THANKS!

Any questions?

You can find me at:

- kgpark88@gmail.com
- https://github.com/kgpark88