DEEP LEARNING for Search

Tommaso Teofili

Foreword by Chris Mattmann

박 경 규

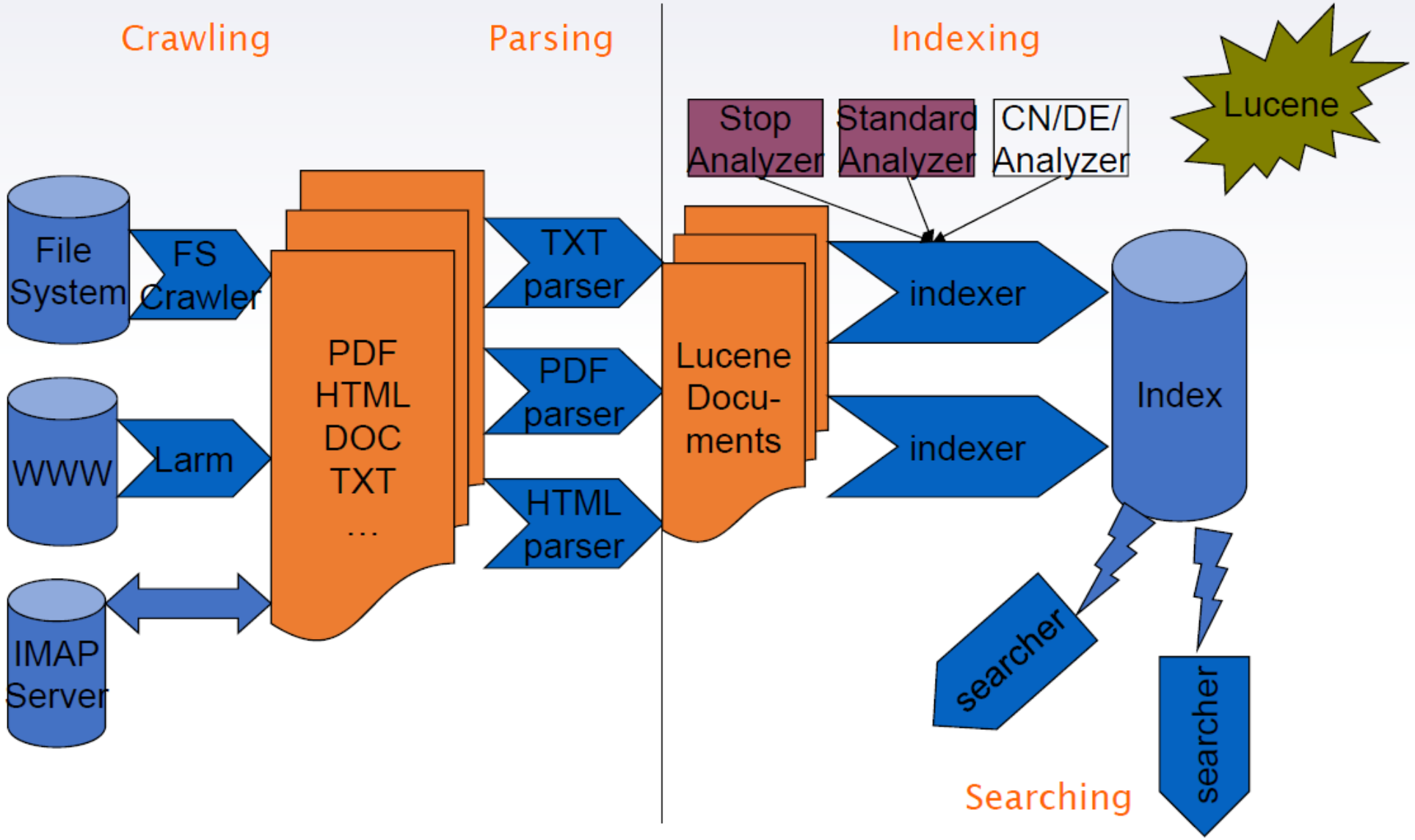
Contents

IR Package 개요

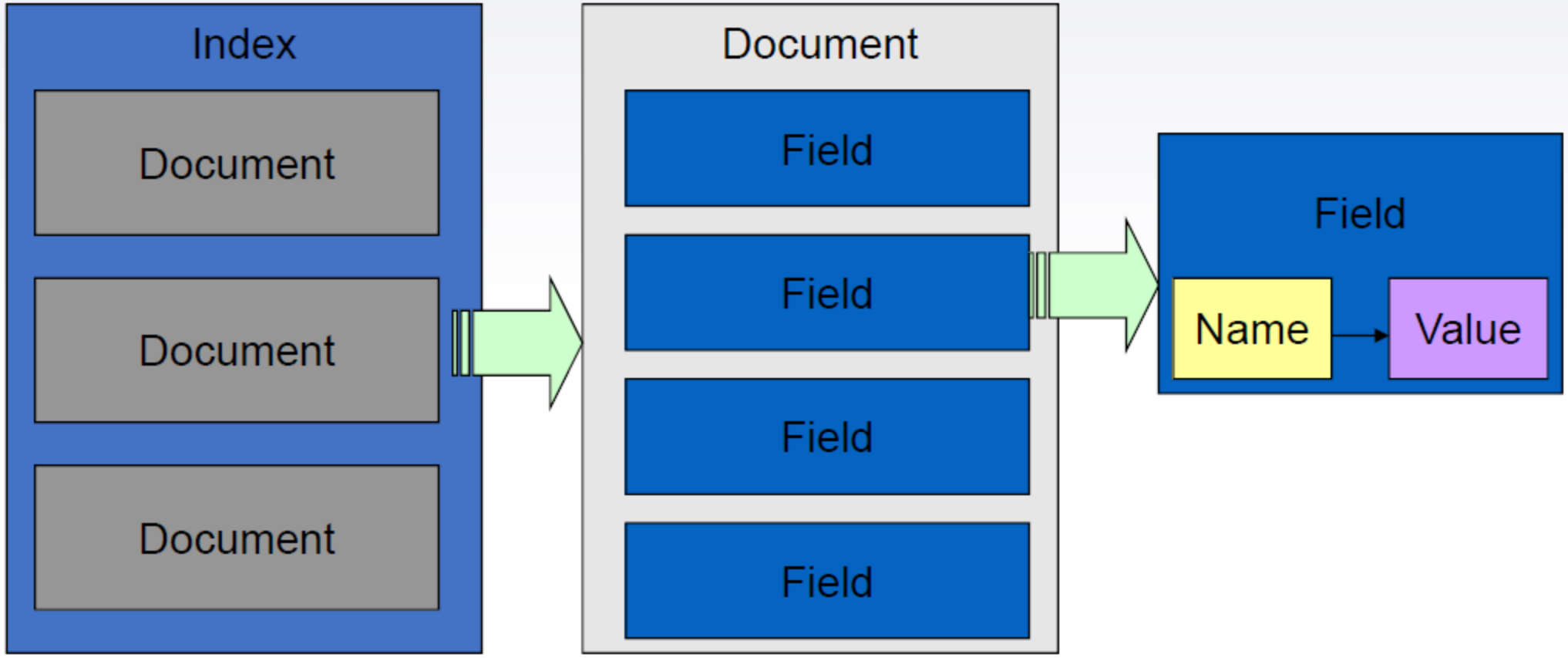
Popular IR packages

- Lucene: low-level full text search library)
- Elasticsearch: based on Lucene, enhanced support for distributed environments
- Solr: based on Lucene, enhanced performance with more functionalities
- Sphinx: SQL database-like full text search engine

Lucene's Open Architecture



Lucene's index (conceptual)

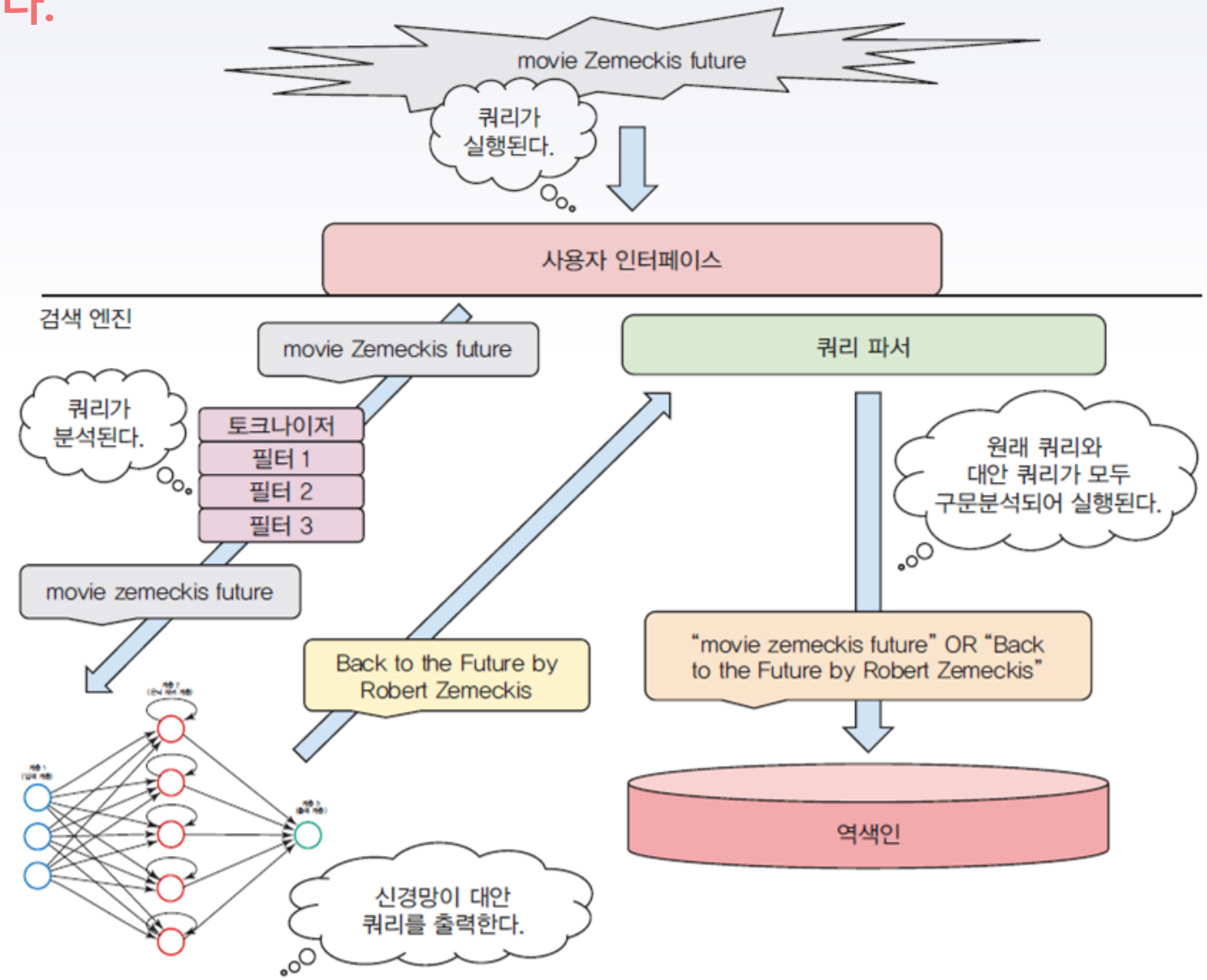


3. From plain retrieval to text generation
(일반검색에서 텍스트생성까지)

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

대안 쿼리(alternative queries) 생성

정보 요구와 사용자가 입력한 쿼리 사이의 간극을 메우기 위한 방법으로 입력된 쿼리를 사용하여 어느 정도 '더 나은' 새 쿼리를 만듭니다.



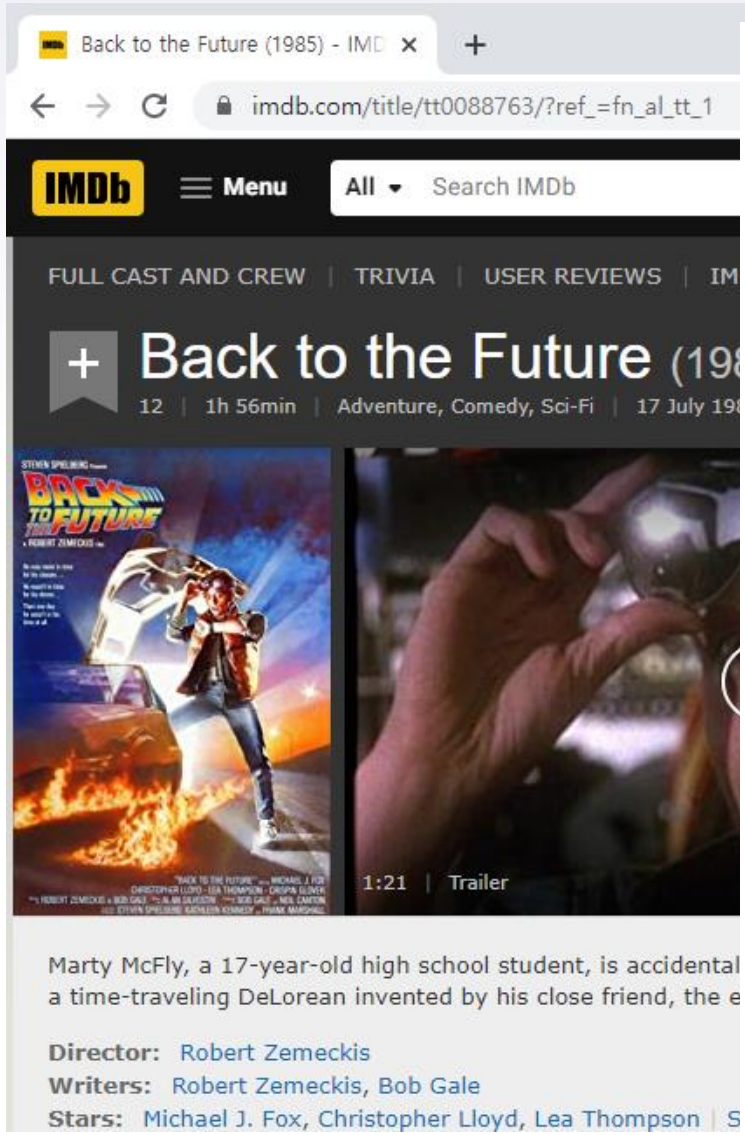
검색엔진 쿼리로그

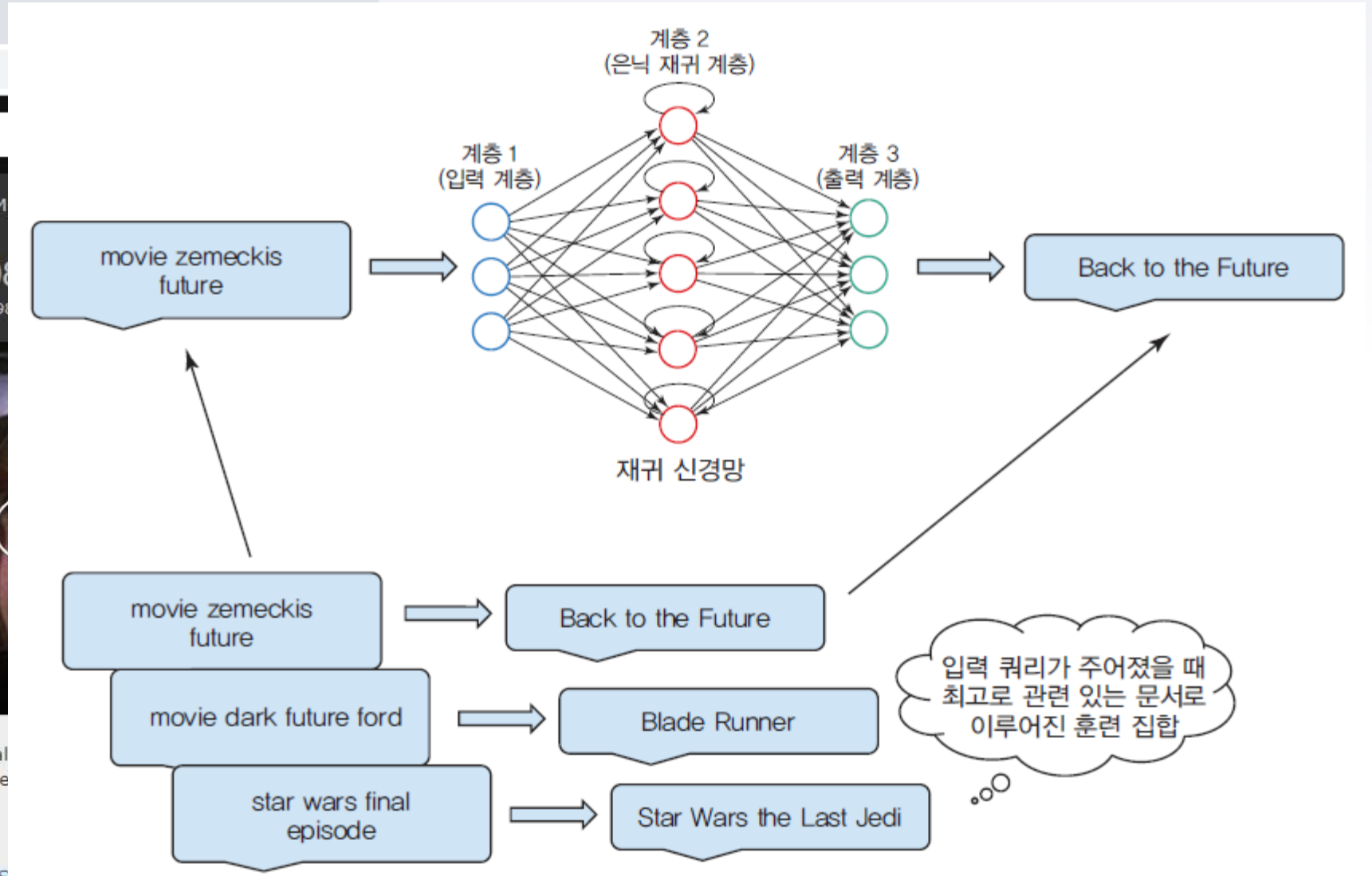
검색엔진의 쿼리 로그를 보면 많은 통찰력을 얻을 수 있다.

```
time: 2017/01/06 09:06:41, query:{"artificial intelligence"}, results:
    {size=10, ids:["doc1","doc5", ...]}
time: 2017/01/06 09:08:12, query:{"books about AI"}, results:
    {size=1, ids:["doc5"]}
time: 2017/01/06 19:21:45, query:{"artificial intelligence hype"}, results:
    {size=3, ids:["doc1","doc8", ...]}
time: 2017/05/04 14:12:31, query:{"covfefe"}, results:
    {size=100, ids:["doc113","doc588", ...]}
time: 2017/10/08 13:26:01, query:{"latest trends"}, results:
    {size=15, ids:["doc113","doc23", ...]}
...
```

The query "covfefe" returned 100 results, and the first two resulting document identifiers are doc113 and doc588.

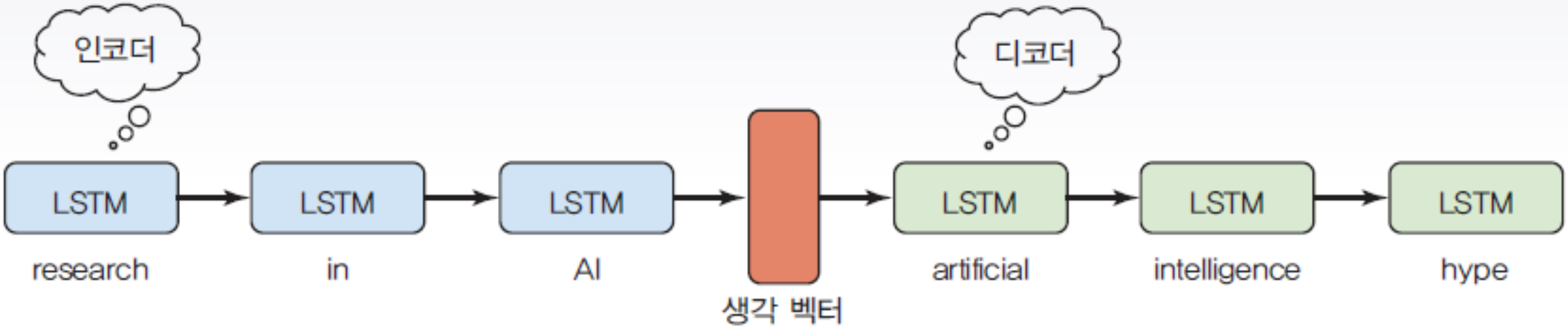
쿼리 학습하기


Back to the Future (1985) - IMD
imdb.com/title/tt0088763/?ref_=fn_al_tt_1
IMDb
Menu
All
Search IMDb
FULL CAST AND CREW | TRIVIA | USER REVIEWS | IM
Back to the Future (19
12 | 1h 56min | Adventure, Comedy, Sci-Fi | 17 July 19
1:21 | Trailer
Marty McFly, a 17-year-old high school student, is accidental
a time-traveling DeLorean invented by his close friend, the e
Director: Robert Zemeckis
Writers: Robert Zemeckis, Bob Gale
Stars: Michael J. Fox, Christopher Lloyd, Lea Thompson | S


계층 2
(은닉 재귀 계층)
계층 1
(입력 계층)
계층 3
(출력 계층)
재귀 신경망
movie zemeckis future
Back to the Future
movie zemeckis future
Back to the Future
movie dark future ford
Blade Runner
star wars final episode
Star Wars the Last Jedi
입력 쿼리가 주어졌을 때
최고로 관련 있는 문서로
이루어진 훈련 집합

시퀀스 학습

Sequence-to-sequence 모델 및 thought vectors는 지도학습 방식으로 텍스트 시퀀스를 생성하기 위한 강력한 도구입니다.



4. More-sensitive query suggestions (그럴듯한 쿼리 제안)

- 제안할 쿼리를 작성하기 위한 일반적 접근 방식
- 문자 수준 신경 언어 모델
- 신경망 내의 조율 파라미터

쿼리 제안 생성

더 강력하고 민감한 쿼리 제안을 생성하여 자동 완성을 위해 널리 사용되는 알고리즘보다 성능이 뛰어나게 할 수 있습니다.

- 쿼리 자동완성 알고리즘의 이점

  결과가 적거나 0인 쿼리 수 감소(재현율에 영향을 미침)

  연관도가 낮은 쿼리수 감소(정밀도에 영향을 미침)

- 효과적인 제안 도출 요점

  제안에 사용할 단어 또는 문장의 정적(수작업) 사전

  이전에 입력한 쿼리 이력(예: 쿼리 로그에서 가져온 쿼리)

  문서의 다양한 부분(제목, 기본 텍스트 내용, 작성자 등)에서 가져온 색인화된 문서

루씬 룩업 API

- queries.txt

```
...
popular quizzes
music downloads
music lyrics
outerspace bedroom
high school musical sound track
listen to high school musical soundtrack
...
```

- 사전 생성

```
Lookup lookup = new JaspellLookup();

Path path = Paths.get("queries.txt");

Dictionary dictionary = new
    PlainTextDictionary(path);

lookup.build(dictionary);
```

Instantiates a Lookup

Locates the input file containing the queries (one per line)

Creates a plain text dictionary that reads from the queries file

Builds the Lookup using the data from the Dictionary

분석된 내용을 활용하는 제안기

Ngram 언어모델 사용

언어모델은 확률분포를 나타내므로 특정 맥락에서 특정 단어나 문자 시퀀스의 가능성을 예측하는데 도움이 될 수 있습니다.

$$P(music\ is\ my\ aeroplane) = P(is|music) * P(my|is) * P(aeroplane|my)$$

```
Lookup lookup = new FreeTextSuggester(new WhitespaceAnalyzer());
```

Let's see it in action, with n set to 2, on the query "music is my aircraft":

```
'm'
--> my
--> music
----
'mu'
--> music
--> museum
----
'mus'
--> music
--> museum
----
'musi'
--> music
--> musical
----
'music'
--> music
--> musical
```

```
'music '
--> music video
--> music for
----
'music i'
--> music in
--> music industry
----
'music is'
--> island
--> music is
----
'music is '
--> is the
--> is a
----
'music is m'
--> is my
```

One of the suggestions for "music is m" matched the desired query ("is my") one character in advance.

내용 기반 제안기

내용 기반(content-based) 제안기를 사용하면 내용은 검색 엔진에서 직접 나옵니다.

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

Dictionary dictionary = new DocumentDictionary(
    reader, "title", "rating");

lookup.build(dictionary);
```

Gets a view (an IndexReader) on the search engine

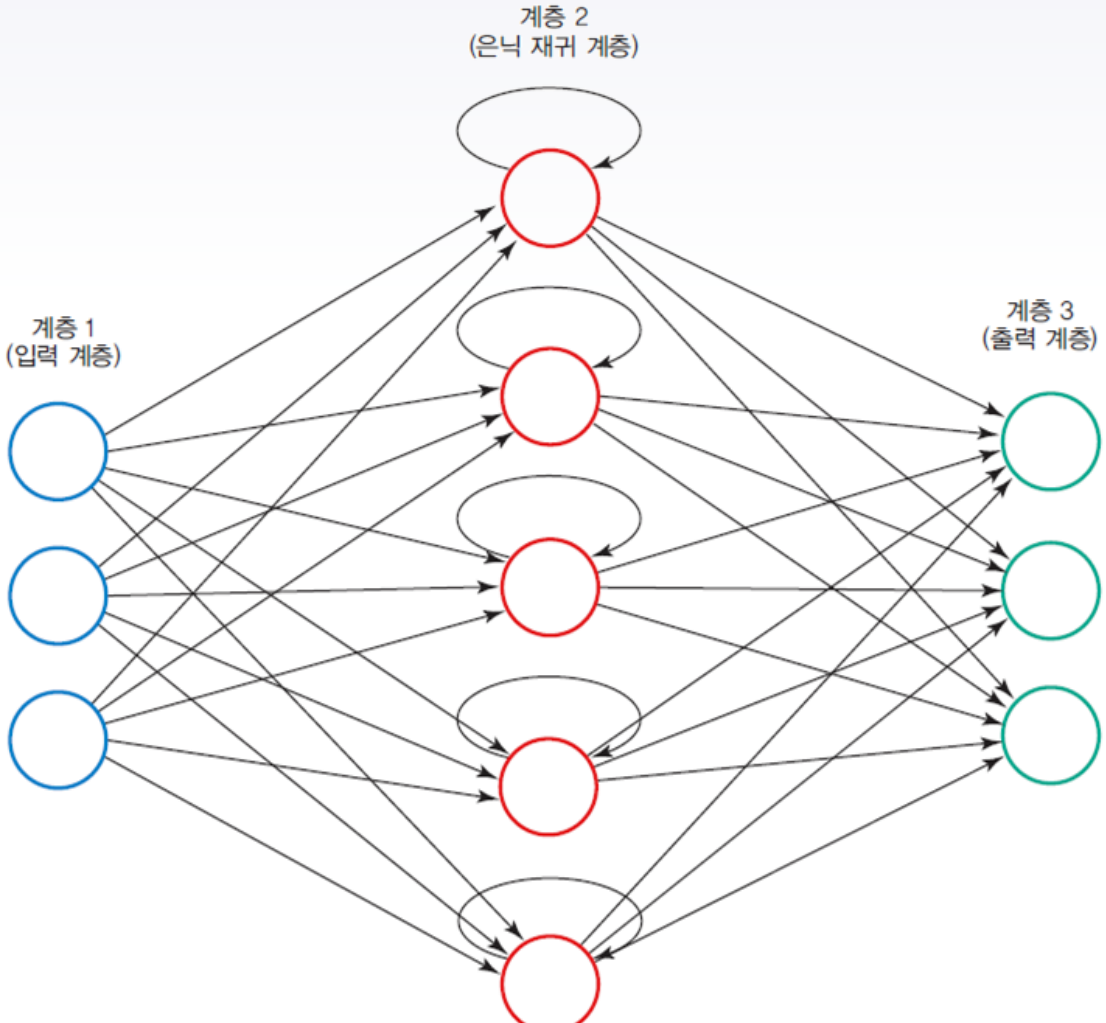
Creates a dictionary based on the contents of the title field, and lets rating decide how much weight a suggestion has

Builds the lookup with the data from the index, just as with a static dictionary

신경 언어 모델(Neural language models)

언어 모델은 텍스트 시퀀스에 대한 정확한 확률을 얻는 법을 학습하며, RNN이 적합합니다.

- 시퀀스 학습을 위한 RNN

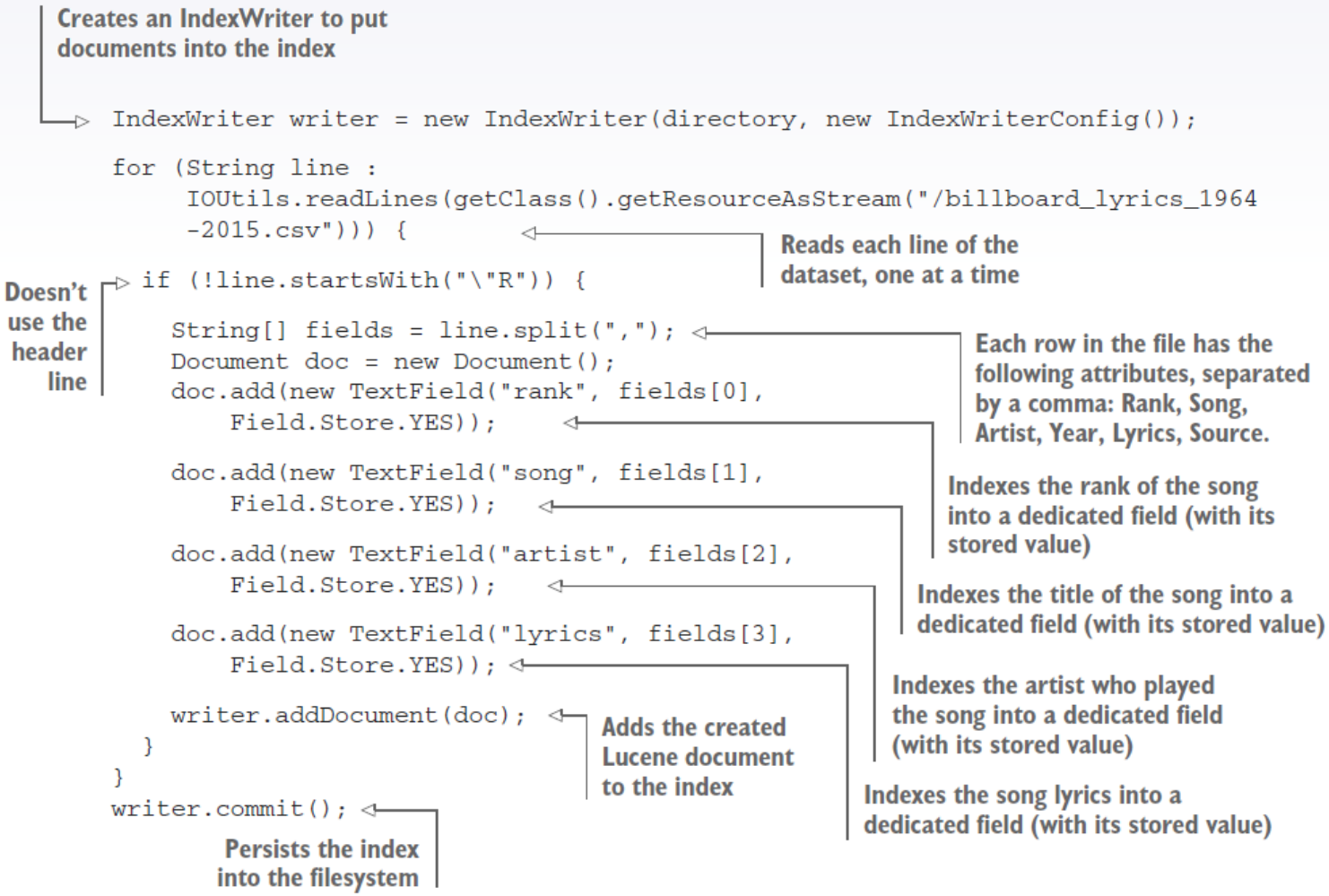



```
LanguageModel lm = ...
for (char c : chars) {
    System.out.println("mus" + c + ":" + lm.getProbs("mus"+c)
}

....

musa:0.01
musb:0.003
musc:0.02
musd:0.005
muse:0.02
musf:0.001
musg:0.0005
mush:...
musi:...
...
```

문자기반(Character-based) 신경언어 모델

신경망을 검색엔진 보조도구로 활용하기 위해서는 신경망에 공급할 데이터가 검색엔진 그 자체에서 나와야 합니다.

- 데이터셋 색인화



문자기반(Character-based) 신경언어 모델

신경망을 검색엔진 보조도구로 활용하기 위해서는 신경망에 공급할 데이터가 검색엔진 그 자체에서 나와야 합니다.

- 문자 LSTM 기반 룩업 구현

```
Dictionary dictionary = new DocumentDictionary(reader, "lyrics", null);
Lookup lookup = new CharLSTMNeuralLookup(...);
lookup.build(dictionary);
```

Creates a DocumentDictionary whose content is fetched from the indexed song lyrics

Creates the lookup based on the charLSTM

Trains the charLSTM-based lookup

- LSTM 훈련 파라미터

```
int lstmLayerSize = 100;
int miniBatchSize = 40;
int exampleLength = 1000;
int tbpttLength = 50;
int numEpochs = 10;
int noOfHiddenLayers = 1;
double learningRate = 0.1;
WeightInit weightInit = WeightInit.XAVIER;
Updater updater = Updater.RMSPROP;
Activation activation = Activation.TANH;

Lookup lookup = new CharLSTMNeuralLookup(lstmLayerSize, miniBatchSize,
    exampleLength, tbpttLength, numEpochs, noOfHiddenLayers,
    learningRate, weightInit, updater, activation);
```

소스코드

https://github.com/dl4s/dl4s

```
public class CharLSTMNeuralLookup extends Lookup {

  private int lstmLayerSize;
  private int miniBatchSize;
  private int exampleLength;
  private int tbpttLength;
  private int numEpochs;
  private int noOfHiddenLayers;
  private WeightInit weightInit;
  private IUpdater updater;
  private Activation activation;

  protected CharacterIterator characterIterator;
  protected MultiLayerNetwork network;

  public CharLSTMNeuralLookup(MultiLayerNetwork net, CharacterIterator iter) {
    network = net;
    characterIterator = iter;
  }

  public CharLSTMNeuralLookup(int lstmLayerSize, int miniBatchSize, int exampleLength, int tbpttLength,
                              int numEpochs, int noOfHiddenLayers, WeightInit weightInit,
                              IUpdater updater, Activation activation) {
    this.lstmLayerSize = lstmLayerSize;
    this.miniBatchSize = miniBatchSize;
    this.exampleLength = exampleLength;
    this.tbpttLength = tbpttLength;
    this.numEpochs = numEpochs;
    this.noOfHiddenLayers = noOfHiddenLayers;
    this.weightInit = weightInit;
    this.updater = updater;
    this.activation = activation;
  }
```

THANKS!

Any questions?

You can find me at:

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