

Hadoop 기반 문서 검색

박치완

Software Maestro 3rd Mentee
chiwanpark91@gmail.com

September 17, 2012

Section 1

검색 시스템 소개

목표

- ❶ 방대한 양의 데이터를 수집하고, HDFS에 저장하는 작업을 통해 HDFS에 대해서 익힌다.
- ❷ 오픈소스 검색엔진 Lucene에서 사용하고 있는 TF-IDF(Term Frequency-Inverse Document Frequency) 알고리즘을 분산 환경에 맞게 설계하여, MapReduce로 구현해본다.

시스템 구조

전체 시스템은 크게 3가지 구성요소로 이루어져 있다.

- ① 수집 - 웹에서 문서를 수집해 단순한 가공만 거친 후, 분산 파일 시스템(HDFS)에 업로드한다.
- ② 색인 - 수집 된 문서를 Hadoop을 통해 Full-Text 색인 과정(TF-IDF)을 거친다.
- ③ 검색 - 사용자의 질의어가 들어오면, 이를 미리 색인된 데이터와 비교하여 연관성이 높은 순서대로 보여준다.

Section 2

수집

수집기(Crawler) 요구사항

- ① 웹에서 문서를 수집해 HDFS에 업로드 할 수 있어야 한다.
- ② 수집한 문서를 기초적인 가공(제목과 본문 분리 등)을 할 수 있어야 한다.
- ③ 특정 URL 규칙을 만족하는 문서만 수집할 수 있어야 한다.
- ④ 문서 수집은 robots.txt 등 수집기가 지켜야 할 사항들을 준수한다.
- ⑤ 수집 대상은 기본적으로 IT 관련 블로그 포스트를 우선적으로 하나, Hadoop을 이용하는 만큼 많은 데이터를 확보할 수 있도록 추후 확장한다.
- ⑥ 수집 과정 중 중단이 일어나더라도 이어서 수집할 수 있어야 한다.

수집기 구조

수집기는 크게 두 부분, Manager와 Worker로 구성된다.

Manager

- 수집 과정을 사용자에게 보여주는 프로세스다.
- 수집 중단, 재개, 새로운 규칙 추가 등을 할 수 있다.

Worker

- 실제 수집을 진행하는 프로세스다.
- Raw Data를 가공하여 HDFS에 올리는 역할도 수행한다.
- 매 수집 과정마다 Manager 프로세스에게 보고하여야 한다.
- 수집 중단, 재개 등 Manager의 요청을 처리 할 수 있어야 한다.

Section 3

색인

TF-IDF 소개

- 특정 단어와 문서 사이의 연관성을 구하는 알고리즘이다.
- 문서에서 등장하는 단어 빈도 TF(Term Frequency)와 전체 문서 집합에서 단어 빈도의 역수 IDF(Inverse Document Frequency)를 기본으로 계산한다.
- 단순한 TF-IDF 보다는 변형을 가한 TF-IDF가 정확도가 높다.
- 어떤 문서에 특정 단어가 자주 출현한다면, 해당 단어는 그 문서와 연관성이 높다고 말할 수 있다.
 - ▶ 건강과 관련된 문서는 건강이라는 단어를 다수 포함할 수 밖에 없다.
- 하지만, 무조건적으로 출현 빈도에 의존하면 전체적인 정확도가 떨어진다.
 - ▶ 어느 문서에나 빈번하게 등장하는 단어는 연관성 측정에서 제외해야 한다.

TF-IDF Algorithm

- 내용 소개에 앞서, 앞으로 사용되는 공통되는 표현을 먼저 소개한다.

표기	의미
t	임의의 단어 (일반적으로 문서 내부에서 단어를 추출)
D	임의의 문서 집합
$n_{t,d}$	단어 t 가 문서 d 에 나타나는 횟수
$ D $	해당 문서 집합에 포함된 문서의 수

TF-IDF Algorithm

- Term Frequency는 문서에서 빈도가 높으면 높을 수록 큰 값을 가져야 하므로 아래와 같이 써볼 수 있다.

$$tf_{t,d} = n_{t,d}$$

- Inverse Document Frequency는 문서 집합에서 단어의 빈도가 낮을 수록 커져야 하므로 아래와 같이 쓸 수 있다.

$$idf_{t,d} = \frac{1}{|\{d : t \in d \in D\}| + 1}$$

- 위의 계산을 통해 TF와 IDF를 구했다면, 우리는 특정 단어 t 와 특정 문서 집합 D , 그리고 집합에 속한 문서 d 에 대해서 TF-IDF 가중치를 다음 식으로 구할 수 있다.

$$tfidf_{t,d,D} = tf_{t,d} \cdot idf_{t,d}(t \in d \in D)$$

Enhanced TF-IDF

앞서 알아본 TF-IDF 알고리즘은 몇 가지 부족한 점이 있다.

- ① 길이가 긴 문서는 빈도 수가 클 확률이 높고, 길이가 짧은 문서는 빈도 수가 작을 확률이 높다. 자연히 위의 경우에는 길이가 짧은 문서가 TF값이 높아 위에 나올 확률이 높아진다.
- ② 단어 1000개로 이루어진 문서 안에서 1번 나온 단어 A에 비해 2번 나온 단어 B는 연관도가 두 배라고 할 수 있을까?

Enhanced TF-IDF

- 이와 같은 문제들을 해결하기 위해 TF-IDF 알고리즘에 로그 함수를 도입하였다.

$$tf_{t,d} = \begin{cases} 1 + \ln(n_{t,d}) & \text{if } n_{t,d} > 0 \\ 0 & \text{if } n_{t,d} = 0 \end{cases}$$

$$idf_{t,d} = \ln\left(\frac{|D|}{|\{d : t \in d \in D\}| + 1}\right)$$

Example

- 임의의 단어 t 를 'health'로 지정하고 아래 예제를 계산해 보자.

$$idf_{t,d} = \ln\left(\frac{4}{2}\right) = 0.6931$$

문서	문서 내용	$\sum n_{i,d}$	$n_{t,d}$	$tf_{t,d}$	$tfidf$
d_1	Health is a necessary condition for happiness.	7	1	0.134	0.093
d_2	It is the business of the police to protect the community.	11	0	0	0

Example

- 이어서

문서	문서 내용	$\sum n_{i,d}$	$n_{t,d}$	$tf_{t,d}$	$tfidf$
d_3	The city health business department runs several free clinics for health professionals throughout the year.	15	2	0.13	0.087
d_4	That plane crash was a terrible business.	7	0	0	0

- 따라서, 사용자가 'health'를 질의어로 선택하였을 경우 TF-IDF 계산값이 높은 순서(d_1, d_3)대로 보여주게 될 것이다.

Section 4

검색

Vector Space Model

- 문서와 단어 사이의 관계를 표현하기 위해 벡터를 사용한다.
- 문서 또는 질의어가 Vector가 되고, Vector의 각 차원(Dimension)이 각 단어별 가중치를 갖는 값으로 표현된다.
- 일반식을 통해 특정 문서 d 를 VSM으로 표현하면 다음과 같다.

$$\mathbf{V}_d = [w_{1,d}, w_{2,d}, \dots, w_{N,d}]^T$$

- 이 때, 각 단어와 문서 사이의 연관성 가중치 $w_{t,d}$ 는 아래의 식으로 구할 수 있다.

$$w_{t,d} = tfidf_{t,d,D} = tf_{t,d} \cdot idf_{t,d}$$

Cosine Similarity

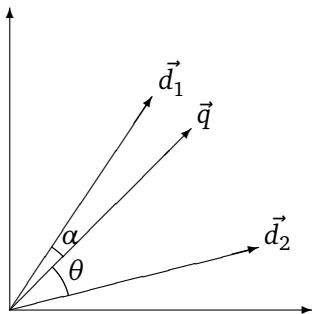


Figure : 문서와 질의어를
벡터로 표현

- \vec{q} 는 사용자가 입는 각각의 문서 벡터이다. 이 벡터간의 사이각에 대한 정보를 \cos 과 벡터 내적의 관계에서 구할 수 있다.

$$\cos \alpha = \frac{\vec{d}_1 \cdot \vec{q}}{|\vec{d}_1| |\vec{q}|}$$

- 두 벡터가 유사하고 연관성이 있으면 있을수록 두 벡터의 사이각은 작아지게 되고, 우리는 연관성을 나타내는 척도로 Cosine Similarity를 사용할 수 있다.

문서 검색 알고리즘

위의 내용들을 종합하여, 문서 검색 알고리즘을 기술하면 아래와 같다.

- ❶ 입력된 질의어를 문서 색인 과정과 동일한 과정을 거쳐 벡터로 표현한다.
- ❷ 미리 색인된 데이터베이스에서 질의어를 포함한 문서 목록을 불러온다.
- ❸ 각각의 문서에 대해 질의어 벡터와의 Cosine Similarity를 계산한다.
- ❹ 계산된 Similarity에 따라 정렬하여 상위 문서들을 출력한다.

Section 5




구현

Subsection 1

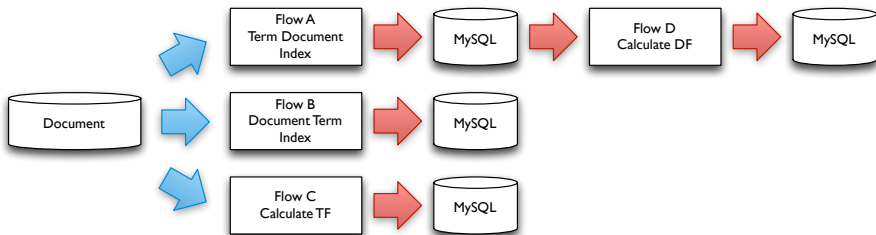
TF-IDF(색인)의 구현

Flow Diagram 규칙

MapReduce Flow를 설명하기 전에, Flow Diagram에서 사용하는 기호들을 소개한다.

-  - HDFS가 아닌 다른 데이터 소스에서의 데이터 입출력을 의미한다.
-  - HDFS에서의 TextFile 입출력을 의미한다.
-  - 시스템 내부에서의 데이터 입출력을 의미한다.

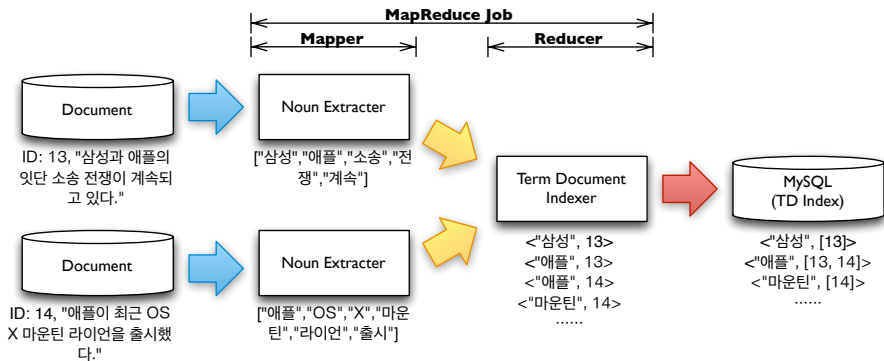
TF-IDF 색인 과정 Data Flow Diagram



크게 두 가지 작업으로 분류할 수 있다.

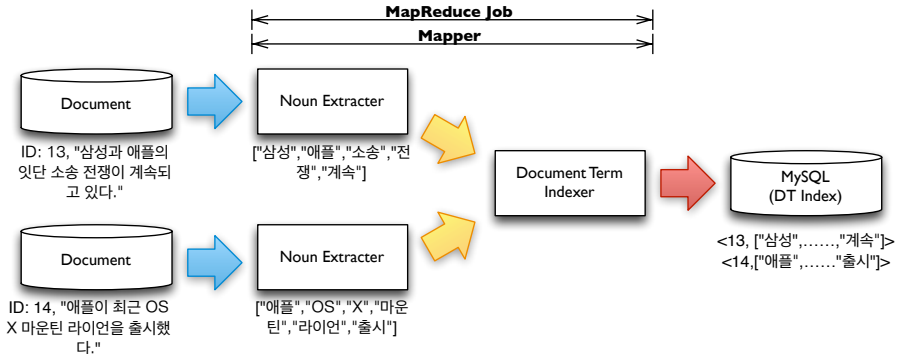
- 가중치 계산의 속도를 높이기 위해 TD, DT 색인과정을 거치는 작업
- 실제 가중치 계산에 필요한 TF, DF를 계산하는 작업

Flow A. Term-Document Index



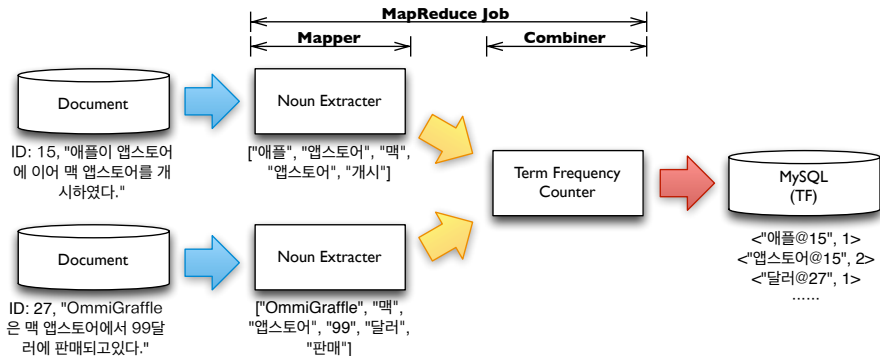
- 특정 단어가 포함된 문서들의 인덱스를 생성하는 작업

Flow B. Document-Term Index



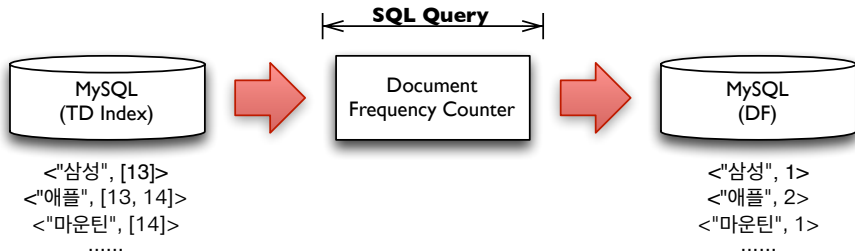
- 특정 문서에 포함된 단어들의 인덱스를 생성하는 작업

Flow C. Term Frequency



- 특정 문서에 포함된 특정 단어에 대해 빈도 수를 계산하는 작업
- 추후 다양한 활용을 위해 일단 WordCount만 수행한다.

Flow D. Document Frequency

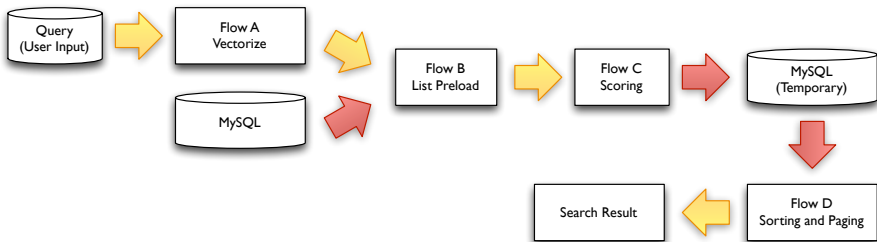


- IDF를 계산하기 위해 선행되어야 하는 DF 계산하는 작업
- 추후 다양한 활용을 위해 일단 DocumentCount만 수행한다.

Subsection 2

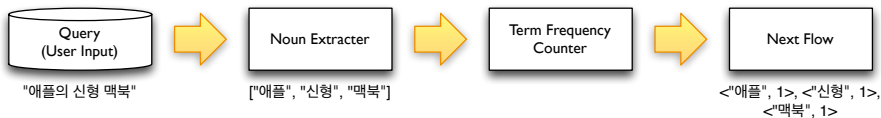
검색의 구현

검색 과정 Data Flow Diagram



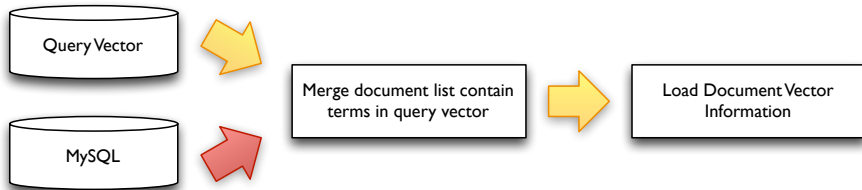
- 사용자로부터 입력된 질의어(Query)로 검색을 수행하는 과정

Flow A. Vectorize



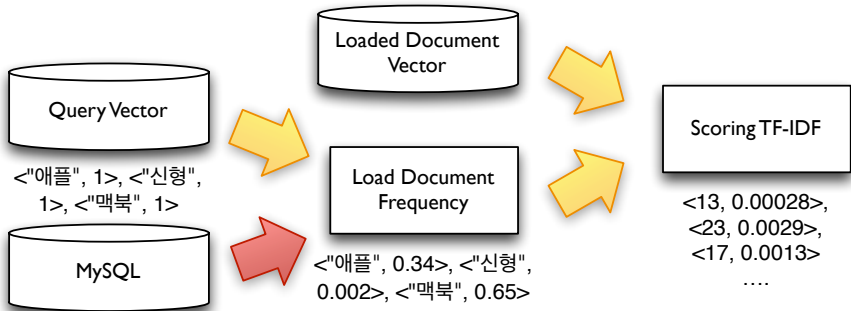
- 사용자가 입력한 질의어를 VSM에 표현할 벡터로 변환하는 과정
- 여러가지 활용을 위해 오로지 Term Frequency 벡터로만 변환한다.

Flow B. List Preload



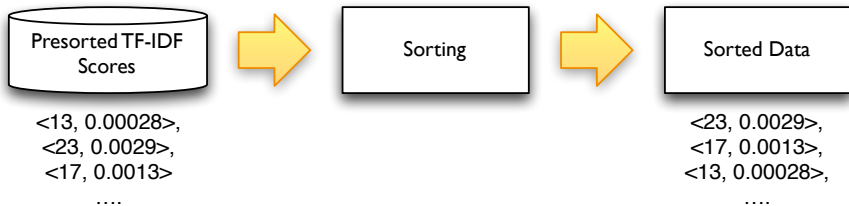
- 질의어 벡터에 속한 단어들을 포함하고 있는 문서 리스트를 불러와 합친다.
- 전체 목록을 합칠 경우, 고려해야하는 문서양이 많아지므로 해당 단어의 TF가 높은 순으로 정렬하여 300개 미만으로 가져오도록 한다.

Flow C. Scoring



- 질의어 벡터와 미리 불러온 비교 문서 목록의 연관성을 앞서 사용했던 Cosine-Similarity 방법을 통해 계산한다.

Flow D. Sorting and Paging



- 계산된 결과를 정렬하여 출력한다.

Section 6

테스트

테스트 환경 소개

- SKT T cloud biz 서버 4대
- 서버 1대의 사양: 1 Vcore, 2GB RAM, 40GB HDD, CentOS 5.5 64bit
- Sun Java 1.6.0_35
- Apache Hadoop 1.0.3
- 서버 IP
 - ▶ Hadoop1: 1.234.45.90 (Namenode, Secondary Namenode)
 - ▶ Hadoop2: 1.234.45.94 (Datanode)
 - ▶ Hadoop3: 1.234.62.102 (Datanode)
 - ▶ Hadoop4: 1.234.62.101 (Datanode)

색인 과정 테스트

- 색인 과정은 Hadoop1 (1.234.45.90) 서버에 ssh로 접속해 이루어진다.
- 색인, 검색 과정에 사용할 데이터는 HDFS에서 /chiwanpark/memento-input에 올려져 있어야 한다.
 - ▶ > `hadoop jar memento-engine-0.1-SNAPSHOT.jar com.chiwanpark.memento.mapreduce.WorkRunner`
- 입력 파일 갯수에 따라 시간이 소요된다.

색인 과정 테스트

```
chiwanpark — root@tcb384:~ — ssh — 115x43
root@tcb384:~
root@tcb384:~
bash
root@tcb384:~ #> hadoop jar memento-engine-0.1-SNAPSHOT.jar com.chiwanpark.memento.mapreduce.WorkRunner
12/09/17 09:43:30 INFO input.FileInputFormat: Total input paths to process : 1862
12/09/17 09:43:30 INFO util.NativeCodeLoader: Loaded the native-hadoop library
12/09/17 09:43:30 WARN snappy.LoadSnappy: Snappy native library not loaded
12/09/17 09:43:33 INFO mapred.JobClient: Running job: job_201209161711_0017
12/09/17 09:43:34 INFO mapred.JobClient: map 0% reduce 0%
12/09/17 09:44:38 INFO mapred.JobClient: map 1% reduce 0%
12/09/17 09:45:26 INFO mapred.JobClient: map 2% reduce 0%
12/09/17 09:46:08 INFO mapred.JobClient: map 3% reduce 0%
12/09/17 09:46:55 INFO mapred.JobClient: map 4% reduce 0%
12/09/17 09:47:34 INFO mapred.JobClient: map 5% reduce 0%
12/09/17 09:47:55 INFO mapred.JobClient: map 5% reduce 1%
12/09/17 09:48:10 INFO mapred.JobClient: map 6% reduce 1%
12/09/17 09:48:22 INFO mapred.JobClient: map 6% reduce 2%
12/09/17 09:48:53 INFO mapred.JobClient: map 7% reduce 2%
12/09/17 09:49:31 INFO mapred.JobClient: map 8% reduce 2%
12/09/17 09:50:13 INFO mapred.JobClient: map 9% reduce 2%
12/09/17 09:50:25 INFO mapred.JobClient: map 9% reduce 3%
12/09/17 09:50:51 INFO mapred.JobClient: map 10% reduce 3%
12/09/17 09:51:33 INFO mapred.JobClient: map 11% reduce 3%
12/09/17 09:52:12 INFO mapred.JobClient: map 12% reduce 3%
12/09/17 09:52:25 INFO mapred.JobClient: map 12% reduce 4%
12/09/17 09:52:55 INFO mapred.JobClient: map 13% reduce 4%
12/09/17 09:53:34 INFO mapred.JobClient: map 14% reduce 4%
12/09/17 09:54:24 INFO mapred.JobClient: map 15% reduce 4%
12/09/17 09:54:32 INFO mapred.JobClient: map 15% reduce 5%
12/09/17 09:54:55 INFO mapred.JobClient: map 16% reduce 5%
12/09/17 09:55:34 INFO mapred.JobClient: map 17% reduce 5%
12/09/17 09:56:10 INFO mapred.JobClient: map 18% reduce 5%
12/09/17 09:56:21 INFO mapred.JobClient: map 18% reduce 6%
12/09/17 09:56:46 INFO mapred.JobClient: map 19% reduce 6%
12/09/17 09:57:21 INFO mapred.JobClient: map 20% reduce 6%
12/09/17 09:57:58 INFO mapred.JobClient: map 21% reduce 6%
12/09/17 09:58:10 INFO mapred.JobClient: map 21% reduce 7%
12/09/17 09:58:34 INFO mapred.JobClient: map 22% reduce 7%
12/09/17 09:59:10 INFO mapred.JobClient: map 23% reduce 7%
12/09/17 09:59:51 INFO mapred.JobClient: map 24% reduce 7%
12/09/17 10:00:08 INFO mapred.JobClient: map 24% reduce 8%
12/09/17 10:00:20 INFO mapred.JobClient: map 25% reduce 8%
```

검색 과정 테스트

- 검색 과정 역시 Hadoop1 서버에 ssh로 접속해 테스트한다.
 - ▶ `> java -classpath
memento-engine-0.1-SNAPSHOT.jar:/opt/hadoop/conf
com.chiwanpark.memento.searcher.cli.SearchRunner -query "스마트폰"`
- 명령을 수행하면 그 결과로 문서 id와 TF-IDF Score를 보여준다.
- 문서 ID를 통해 HDFS에서 해당 문서를 열람할 수 있다.
 - ▶ `> hadoop fs -cat /chiwanpark/memento-input
/e02f5b1df830e8fcf89df333dc2dd642a9f0569ee6aea26cc1e3ec3a22e4
b988bfadb397c1ba7bd593feb5bd99276b9ce15a84741b5fe583d1dc2cb9
110ae70c.txt`

검색 과정 테스트

```
chiwanpark — root@tc3b384:~ — ssh — 115x43

root@tc3b384:~
root@tc3b384:~
bash

root@tc3b384:~ # java -classpath memento-engine-0.1-SNAPSHOT.jar:/opt/hadoop/conf com.chiwanpark.memento.searcher.C
li.SearchRunner --query "스 마 트 폰"
query: 스 마 트 폰
query vector
termList count: 2
폰 document count: 10
스 마 트 document count: 10
20
document id: document vector: 3fcf1cc91feacfd4af63a46bf3373027db91a3b73bf02a21ffd448970d401e52ea3ae188431522f196c32
eb348c1ab73d26c805b1ceef5aaae9a93a185b8e61b point: 0.00534169746439613
document id: document vector: e02f5b1df830e8fcf89df333dc2dd642a9f0569ee6aea26cc1e3ec3a22e4b988bfadb397c1ba7bd593feb
5bd99276b9ce15a84741b5fe583d1dc2cb9110ae70c point: 0.0032723699839665443
document id: document vector: 2ac0e1628134d8983dbd7fa1b6bc1b9a789de4f15487b9dea7adc7276a893a8145aa1494af4ee58c08e8f
e7ec062c33f280a1128509bb0d54874700ed9d4732 point: 0.0018438941196472767
document id: document vector: 2af88ef1811a428d595a2a0eddce650482827236b8057492373fc5bcb832780b5a7ccfffee6a1f53ef22b
d96e8689a0c1d8acbbd66e6f19fb60769d792ed7b9 point: 0.0013371597485748962
document id: document vector: e1096d62ff2c23ab6edda106dd5ad287420a882e74c34aafb813a77202dbd0e38bd72e389897622a5b9af
efe348017b3adac2765c5c1f42a5e6474814c18b2a6 point: 0.001265400818351811
document id: document vector: 2ad45cd2a0b75dd06284ca2d54326d1ebba772bc18d50326cfd780e74b80a55e841f6fef5dcbbf9e9b6
67c3808d21e7beeefca718b31127f4b9221c68486d92 point: 0.0012590399722664592
document id: document vector: 2ade24c4b087bd62624748efad29dfac874cead75865ffe90e6068b838fabbcb651be425844fe846c542
e1efa456ea86df7f365f0009133fd5f5bad462a5b81 point: 0.0011809109119233385
document id: document vector: 2a06acfa2a5a9e460f78b4d1a05f04d16f1a61baf83ebe68b169b0e3d0da0169ea6c8bc6e1b31184b99618
cea96657b57f8ca21b0e549d9ed66af732dc296cae8 point: 0.0011194069506153543
document id: document vector: 3f02c80bce128060d76cbcd3c429bd41bc22d82cd9718003e89853186e830e914776d5a0dbb1db24e4229
f3f962ca9b7a5a8b6fdd3f11147af7b6cc6df1f7054 point: 0.0010716498706265583
document id: document vector: 2a9ed7d07d536b180bb5dd14392a55381398ce8be19e7c74714e1a5e0768b01e11149827269c0c695fb57
a26c731b60c8834cfef19ba72f6db9b241b10f8b04 point: 9.34177646326643E-4
document id: document vector: 3f203c994850a501db2ddadb34f671e43a3da42d2799826317bef16851dd1d68426a90b3e97dde2b1926
7f42e3035146b837f1ab97e6b4b2f3dcd835504404 point: 7.26626333445435E-4
document id: document vector: 3f2110ef05c3d85ab426e226d9b77e731d16ae51fcd2364117d4bcd7af4e177664b6adb98144cc936cd88
919b8d1658b300b7abb2bbd09383ece7030c344dfc4 point: 7.03519318646433E-4
document id: document vector: 2ac34d5804f91eab62515627808200d36303ef3fbb0a2a43b222f62495d913bc383f04b99c4d2ce870b00
958c5c506ef1785b2e412719560a205c99a3c16bcd20 point: 6.461733798088524E-4
document id: document vector: 3f2ab3bc3e05789c61e2b940783b35952c24f8325c8ccf76346d24695595221453b7bbdb7c6efb0a19022
d77e0f5c756762fc68a94ab907efea5d0b4454c0dec point: 5.209115392593751E-4
document id: document vector: e0af2d1442919e9b1de877f93ac7ca31c6c7a2b606037028d9e1adc259cd5c52f379dfba33e308bac1ee3
bb5723d18e0ff2a0813ff1b30020138e0ccdb20feaf point: 4.665278908758919E-4
document id: document vector: 3f05defa1ce7d731fd2f3478a066587a7acc631ef32bf83c757a412435bf1835341f402f6b9cdae34f6
```

검색 과정 테스트

```
chiwanpark — root@tcb384:~ — ssh — 93x28
root@tcb384:~
root@tcb384:~
bash
root@tcb384:~ #> hadoop fs -cat /chiwanpark/memento-input/3fcf1cc91feacf4daf63a46bf3373027db91a3b73bf02a21ffd448970d401e52ea3ae188431522f196c32eb348c1ab73d26c805b1ceef5aaae9a93a185b8e61b.txt
[블로터 TV] 얼굴이 박 찬 방송 @갤럭시 노트, 갤럭시 8.9 이번 주 '얼찬방송'은 삼성전자 갤럭시 시리즈로 돌아왔습니다. 삼성전자가 지난 월요일 미디어데이를 열고 3종의 새 모바일 기기를 발표했죠. '갤럭시 노트'와 '갤럭시 탭 8.9', '갤럭시 넥서스'입니다. 삼성전자쪽 설명에 따르면, 갤럭시 노트는 출시 전부터 해외에서 큰 호평을 받았다고 합니다. 5.3인치 대형 화면을 채택했으며 'S펜'이라는 이름이 붙은 디지털라이저도 갤럭시 노트의 특징입니다. 하루가 다르게 간편한 조작법을 원하는 사용자에게 어떤 평가를 받게 될 지 벌써 궁금해합니다. 갤럭시 탭 8.9는 기존 '갤럭시 탭 10.1'과 차이점이 없습니다. 화면 크기가 작아진 만큼 무게가 200g 정도 줄었죠. 들고 다니기 편해졌다는 장점이 생겼지만, LTE 대응 기기라는 점에서 역시 시장의 반응이 궁금합니다. 갤럭시 넥서스는 구글과 삼성전자의 두 번째 합작 레퍼런스 폰이죠. 구글의 따끈따끈한 안드로이드 4.0root@tcb38root@tcb38
root@tcb384:~ #>
```


Section 7

토의

Subsection 1

성능 측정과 품질 검증

성과와 품질 검증 방법

- 성능 측정은 전체 MapReduce의 수행 시간을 구하고, 해당 시간 동안 처리한 파일의 수를 구해 성능 측정의 기준으로 삼는다.
- 품질 검증은 이번 TF-IDF 시스템 구현이 Lucene의 시스템과 유사한 부분이 많이 Lucene에 해당 도큐먼트 집합을 넣었을 때의 Score와 구현한 시스템이 계산한 TF-IDF Score를 비교하는 방법을 생각해 볼 수 있다.

성능 측정 결과

- Test1

- ▶ Job1 - 102개 문서/3분 58초 (참고 자료 열기)
- ▶ Job2 - 102개 문서/3분 43초 (참고 자료 열기)
- ▶ 초당 0.22개 문서 처리

- Test2

- ▶ Job1 - 99개 문서/3분 54초 (참고 자료 열기)
- ▶ Job2 - 99개 문서/4분 4초 (참고 자료 열기)
- ▶ 초당 0.21개 문서 처리

성능 측정 결과

● Test3

- ▶ Job1 230개 문서/8분 44초 (참고 자료 열기)
- ▶ Job2 230개 문서/8분 16초 (참고 자료 열기)
- ▶ 초당 0.22개 문서 처리

● Test4

- ▶ Job1 1862개 문서/1시간 3분 55초 (참고 자료 열기)
- ▶ Job2 1862개 문서/1시간 4분 27초 (참고 자료 열기)
- ▶ 초당 0.24개 문서 처리

Subsection 2

개선 사항

품질 개선 사항

- 이번 프로젝트에서 사용한 한나눔 형태소 분석기의 분석 품질이 좋지 않아, 오히려 공백을 기준으로 단어를 분리하고 그 결과에서 조사를 직접 제거한 후, 미리 준비한 단어 사전과 매칭하여 키워드를 추출하는 방법이 더 좋은 품질을 가져올 수 있다고 생각한다.
- 버즈니 형태소 분석기의 경우 분석 품질은 우수하나 많은 양의 자료를 처리할 수 없어 사용하지 않았다.

성능 개선 사항

- 데이터 저장 구조가 현재는 하나의 문서를 하나의 파일로 구현하였는데, 로그 분석 결과 Hadoop에서 File Split마다 Mapper 클래스를 초기화하기에 이 때 초기화 시간으로 많은 시간을 소요하였다. 따라서 이를 개선하여, 하나의 문서를 Single line으로 표현하고 수십개의 문서를 묶어서 Split 단위를 늘려 초기화 횟수를 감소시킴으로써 성능 향상을 꾀할 수 있다.
- 테스트 시스템에서는 Cloud System 4대를 사용하였는데, 이는 VM으로 이루어져 I/O 성능이 별로 좋지 않다. VM이 아닌 실제 시스템에서 돌리면 보다 나은 성능을 보여줄 것으로 기대한다.