D 系クエリと AA の LDA について

木村　健

平成 30 年 10 月 25 日

1　このドキュメントについて

このドキュメントでは D 系（ドコモ）クエリの LDA(木村は未達) と AA(All About) 記事の LDA（今絶賛この文章を書きながら回している）について、作業した内容の共有と、その時生成されたデータや使ったプログラムについて説明を加える。

2　D 系クエリ LDA

まず、D 系クエリの膨大なデータの蓄積がある。これは Apache Hive のデータ構造になっていて、スキーマを構成するフィールドにクエリ文字列と時間、個人識別の id が含まれている。解析するためにはまずこれらのデータを単一の CSV に落とすことが考えられるが、現状 2 ヶ月ほどのデータについてこれを試みたところ、おそらく 600GB くらいのデータになることが試算と実験でわかった。

このため CSV をインメモリで処理できる限界量まで CSV を膨らませて解析するか、そもそも CSV ファイルで解析することを諦めて、Hive 上で最終的に LDA 解析に必要なデータをデータベースのデータとして収集するか、どちらかを選択する必要があり、木村は後者の手法（データベース上で Hive で操作）を選び作業を途中まで進めた。

まず Hive で形態素解析をする Apache Hive UDF(User Defined Function) として mecab を使ったものを作る必要性を感じ、

https://github.com/kazuhira-r/kuromoji-with-mecab-neologd-buildscript

から派生した新しいレポジトリを作成し、kuromoji を経由せず mecab を直接 Java API で制御する Hive UDF を作った。

https://github.com/kimrin/spark-hive-udf-mecab

本レポジトリの一連のスクリプトは、

mecab をインストールし、neologd をインストールし、次に Hive UDF を含んだ Java プログラムをコンパイルして二つの Jar にまとめる。

使用側では、Hive を Hue のインタフェース上などから、ADDJAR して TEMPORARY FUNCTION を作り、これを SELECT 文のフィールドに対して関数のように使う。今のところ表層を分かち書きする surface 関数だけが提供されている。

```
ADD JAR MeCab.jar;
ADD JAR spark-hive-udf_2.10-0.1.0.jar;
CREATE TEMPORARY FUNCTION surface AS 'com.ardentex.spark.hiveudf.MecabSurface';
```

```
SELECT keyword, surface(keyword) FROM full_query WHERE ver='2018-06-01' AND sub_ver='00-00-00';
```

ただ、調査の結果性能としては D 系クエリ 5 日分くらいで mecab のインスタンス破壊が観測され、回避策を入れたが、結局のところ mecab の対クエリ耐性からこれ以上を MapReduce や Tez で実行すると途中で落ちてしまう現象に遭遇した。

結果的にこれらの危ない日付を回避してデータベースのテーブルを構成すれば良いのだが、そこまで行かなかったのと、LDA のために必要な TF（term frequencies) を Hive SQL でどのように順に集計していけばいいのかまだ学習途中で途中までしか進められなかった。

少し spark-hive-udf-mecab に力を入れすぎた感がある。のちの調査で Hive Mall に含まれている kuromoji では Hive Mall を導入するだけである程度の性能の Kuromoji 解析機が Hive UDF として使えることがわかり、事実上 spark-hive-udf-mecab は遺跡となった。

結果、LDA に必要な擬似 SVM データを得られないまま今日に至っている・・・

棚谷さんが 3 日分のクエリーについて結果を出しておられるそうです・・・

3 AA の LDA について

AA からほぼ HTML ママの CSV ファイルを頂いた（zip で)。これを Python3 の beautiful soup で解析して、手頃な扱いのできる JSON ファイルに直したところ、27GB くらいの大きさになった。(この作業は木村が実施した)。

この JSON は次のような構成になっている。

```
{"記事番号": {
                    "記事番号枝番": {
                                                "articles": [
                                                      {"H3": "見出し",
                                                       "text": ["テキスト 1", "テキスト
2", ..., "テキスト_n"]
                                                      }, {...}, {...}, ...
                                                ]
                     }
}
```

基本的に H3 タグが見出しなのだが、一部 font タグで代用されているものがあった。それらについては単純に抜き出すのではなく、b タグで囲ってあるものだけを H3 として抽出した。

また text については連続しているものが途中で物理的に切断されているものがある。このため文の区切りとしては不適切な切り方も多く存在する。

3.1 LDA やってみた

上記の JSON データを使って LDA の元データ（SVM ファイル）を作ってみた。ベースとして ja.text8 のプログラムを使った。ドキュメントとしては H3 の全てを分かち書きしたものと、text を連結した一つのテキストとして分かち書きしたものをデータとして使った。

結果、かなり大きな svm ファイルができた。

documents = 393007, lexicon = 835732, nwords = 38600153.

今回は時間の関係上 N=700 とやや小さい値で検証した。topics も 100 とやや小さめである。それでも m4.4xlarge のマシン（RAM 64GB) で 17 時間ほど掛かった。

時間があれば、topics=2000, N=1000 くらいのタスクをやってみたい。

あと付記として、theta を save するとき巨大な np.zeros を実行するようで、theta を save する途中でメモリエラーで落ちてしまう現象が出た。単純に theta のセーブをコメントアウトして（現在のところ使わない）、対処した。

結果の Excel ファイルについては別箇添付する。N=700 ではあるが、AA コーパスの品質の良さが幸いしてか、妥当な分類解となっている。

N=700 時の perplexity=9466 である。

参考文献

[1] Kevin P. Murphy: Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series) The MIT Press, 2012.

[2] Blei, David M. and Ng, Andrew Y. and Jordan, Michael I.: Latent Dirichlet Allocation, J. Mach. Learn. Res. 3/1, volume 3, 2003.