Topic Analysis in AceMap

Hao Lu

With group members Zhengtian Xu, Yiyi Shen, and Ziming Cheng

5130309109

June 24, 2016

Contents

1 Motivation

- Exploit the idea of Data Mining: find what's under the explicit statistics, rather than just visualize plain data.
- Design an algorithm for calculating the trend of a topic and deciding the dependencies between topics.
- Deploy the algorithm in AceMap: most theoretical algorithms are not practical, thus we need a tradeoff between being pragmatic and precise.

2 Principle

The popular algorithms for topic analysis often use LDA models, which is used on a set of untagged papers. After iterative clustering, the papers will be clustered into different topics. As a result, we can get two kinds of distributions, that is, distribution of papers over topics, and distribution of topics over papers. As you might have expected, this is a rather time-consuming process.

3 Model

3.1 Database

The database from Microsoft is a powerful one, with topics already labeled among keywords.

PaperID	KeywordName	FieldOfStudyI...
000000BE	vaporization	0BABB10F
0000024B	population dynam	0206C527
00000569	planetary nebula	05AF48EC
000010BF	boundary layer	06A6EEEF
00001561	falsification	06ED6B1B
000017D0	three dimensiona	09633D62
00001B5B	educational techn	00FAE71A

However, we must acknowledge that the data is in a poor manner. The topics are in a tree-like hierarchy. Sometimes one subtopic may belong to multiple topics in different layers, with confidence or probability. This makes our work difficult. So we design a recursive procedure for classifying a topic's related, parent topics.

ChildFieldOfSt...	ChildFieldOfSt...	ParentFieldOf...	ParentFieldOf...	Confidence
0001BE57	L3	09892D7B	L1	0.5
0001DDB2	L3	0B0FEB68	L0	1
000226EC	L3	052C8328	L0	1
0002487D	L3	092C7EF1	L1	0.5467854356075009
0002C926	L3	065808E3	L2	0.5
00032F94	L3	064E5072	L1	1
0004676F	L3	07982D63	L0	0.6038416171751301
0005EC96	L3	007E3B49	L2	1

3.2 Distribution of papers over topics

For each paper, we get the distribution over topics in the following way:

1. Gather the papers that this paper has referred to.

2. For each reference, get the topics that the keywords map to.

3. For each topics, follow the hierarchy and get the topics in the layer we want.

In this way we'll get a probabilistic distribution.

```
def collectPhi(fieldID):
    f = codecs.open(fieldID + ".phi",
        encoding="utf-8", mode="w")

    # Need to fill in the user and passwd.
    connection = MySQLdb.connect(...)


    cursor = connection.cursor()

    # Find all related topics
    cursor.execute(
        "SELECT ChildFieldOfStudyID, Confidence
        FROM FieldOfStudyHierarchy
        WHERE ParentFieldOfStudyID = '{}'".format(fieldID))
    fields = cursor.fetchall()
    print len(fields)
    #for i in fields:
    #    print i

    # Find all papers in a specific topic
    for field in fields + ((fieldID, 1),):
        print field
        cursor.execute(
            "SELECT PaperID
            From PaperKeywords
            Where FieldOfStudyIDMappedToKeyword =
            '{}'".format(field[0]))
        result = cursor.fetchall()

        # Find information of each paper
        for paper in result:
            cursor.execute
                ("SELECT PaperID,
                PaperPublishYear, PaperRank
                FROM Papers
                WHERE PaperID = '{}'
                LIMIT 0, 10".format(paper[0]))
            arr = cursor.fetchone()
            cursor.execute(
                "SELECT *
                FROM PaperKeywords
                WHERE PaperID = '{}'".format(paper[0]))
            keywordNum = len(cursor.fetchall())
            f.write(arr[0])
            f.write(u"\t")
```

```
            f.write(str(arr[1]))
            f.write(u"\t")
            f.write(str(arr[2] / 10000.0))
            f.write(u"\t")
            f.write(str(keywordNum))
            f.write(u"\t")
            f.write(str(field[1]))
            f.write(u"\n")

    connection.close()
    f.close()
```

Code Listing 1: `collectPhi()`

3.3 Distribution of a topic over papers

For each topic in the layer we want, we generate its distribution in the following way:

1. Find all child topics.
2. For each child topic and the topic itself, find papers belonging to the topic.

The program for this is not completed by me, so I'll not post it here.

3.4 Topic Strength and Dependencies

With the two kinds of distributions, we can perform many valuable calculations. For example, we can collect all papers related to a topic, thus forming its strength or evolution pattern. Also, we can get the dependency between two topics, thus generating a dependency graph.

```
def calculateStrength(paperYear,
    paperRank, paperKeywordNum, paperConfidence):
    strength = [0] * 120
    num = [0] * 120
    connection = MySQLdb.connect(...)
    cursor = connection.cursor()
    cnt = 0
    try:
        for paper in paperYear.keys():
            year = paperYear[paper]
            rank = paperRank[paper]
            keywordNum = paperKeywordNum[paper]
            Confidence = paperConfidence[paper]
            paperStrength = rank / keywordNum * Confidence
            strength[year - baseYear] += paperStrength
            num[year - baseYear] += 1
            cursor.execute(
                "SELECT␣PaperID
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣FROM␣PaperReferences
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣WHERE␣PaperReferenceID␣=␣'{}'".format(paper))
            citations = cursor.fetchall()

            for citation in citations:
```

```
                if citation[0] in paperYear:
                    citationStrength =
                        paperStrength * alpha / len(citations)
                    strength[paperYear[citation[0]] - baseYear]
                        += citationStrength
            cnt += 1
            #if cnt % 100 == 0:
            print cnt
    except Exception:
        pass
    finally:
        connection.close()
        return strength, num
```

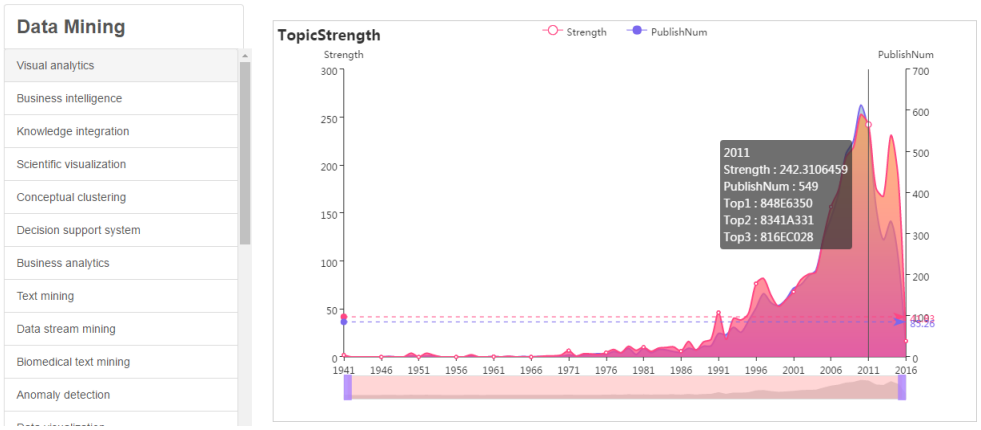
Code Listing 2: `calculateStrength()`

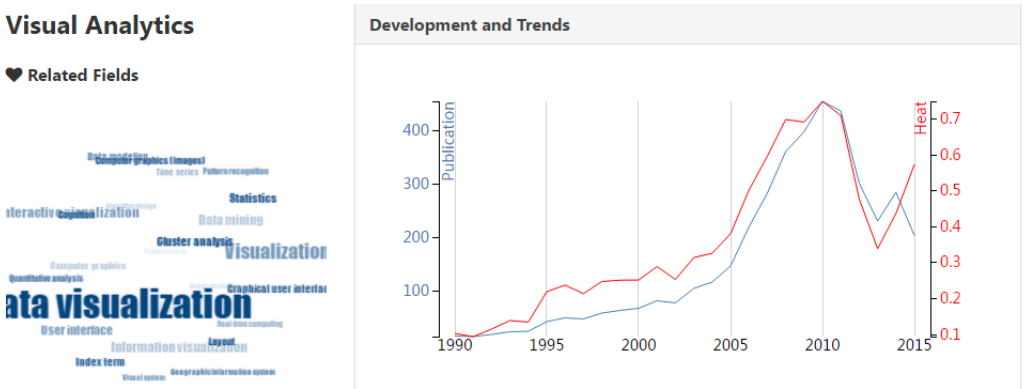
4 Demo

The visualization is mainly finished by the two girls in our group. I and Zhengtian Xue also contributes in some aspects.

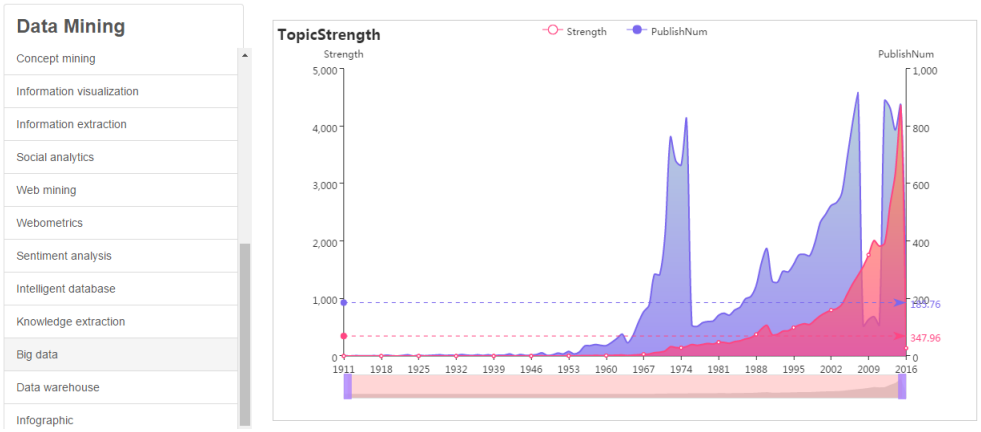
4.1 Topic Strength

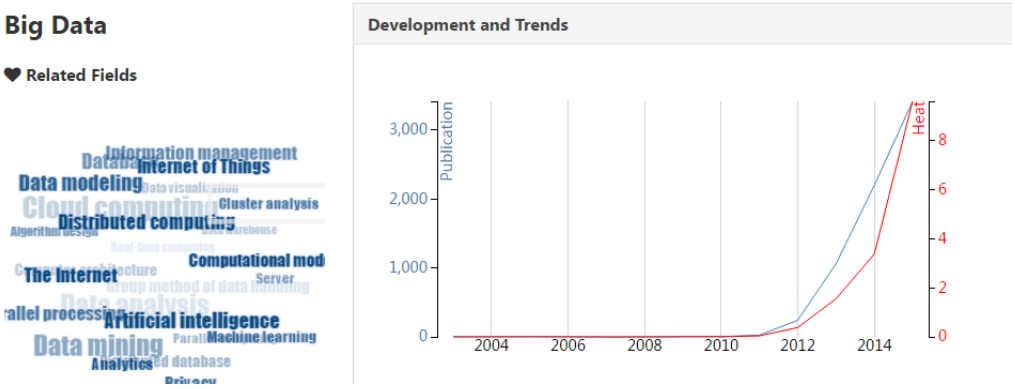
Here I select two topics under Data Mining and show the results of ours compared to the results now on AceMap.





You'll see we've get a smoother and more reasonable strength. For example, you can see that in recent years, although the number of publication falls (which may be because of an incomplete database), the strength doesn't fall that much. This corresponds to our expectation.

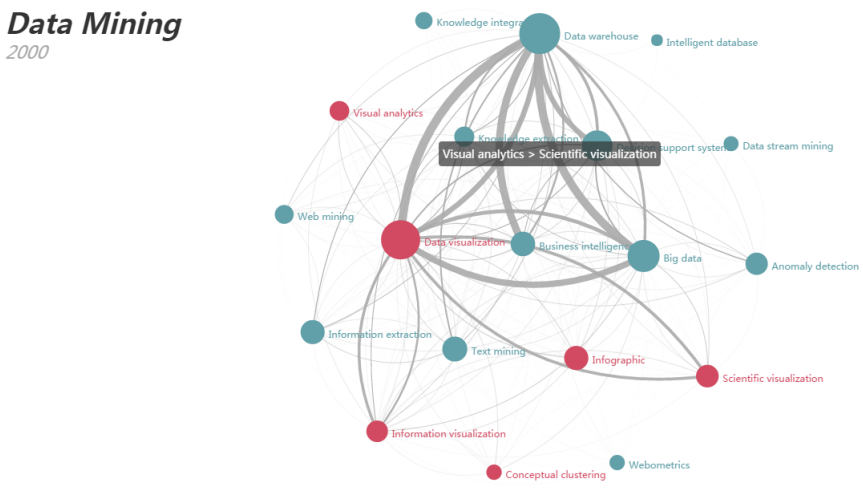


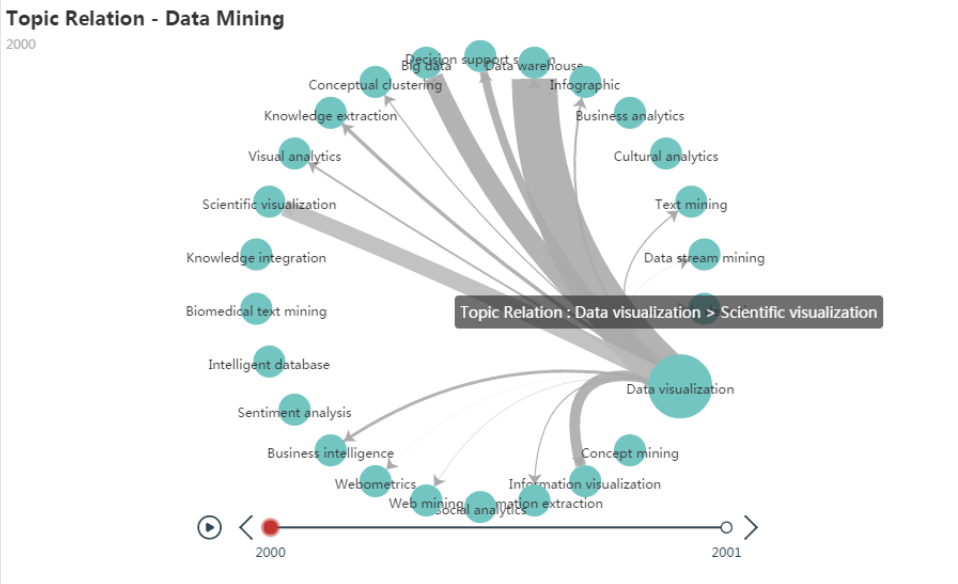


Here we'll get another view. You'll see that maybe due to the false classification of the database, the number of papers changes very abnormally. However, strength here is a good indicator, since it's very smooth and grows to our expectation.

4.2 Topic Relation

We can also show the relation among topics by their dependencies.





5 Future work

- The program is still not fast enough. The very bottleneck is the cost of accessing the database.
- Many internal procedures are not integrated.
- Further analysis of the data we collect is available such as timeline and evolution of topics.
- We'll try to cover all CS topics in the future.