SOCIAL SEARCH ENGINE AND INTELLECTUAL DATABASE OF PEOPLE

Oleksandr Kuzomin, Mariia Tkachenko

Abstract: *Social networks are the core of modern Web. Nowadays almost every person have pages in several social networks. Social networks are very structured, so it is possible to make advanced queries for data in social networks. Special algorithms can make Web search more powerful using social information. We developed this social search engine as a part of our Tempus project for Business-Alumni. It searches for alumni in Web by list of alumni from universities.*

Keywords: *information retrieval, content analysis, document indexing, search engine, crawler, social networks, social search, alumni*

ACM Classification Keywords*: D.2.11 Software Architectures, H.3.1 Content Analysis and Indexing, H.3.2 Information Storage, H.3.3 Information Search and Retrieval*

Introduction

We created social search engine. It collects people public information across the Web and stores it in our database. It uses different social networks as as source of information, as well as our own crawler that searches for social information on websites that are not social networks. It also saves links to all posts, publications, public messages and photos of found persons. So when performing search it is easier to understand is it correct person, when you see messages by this person.

Using APIs of well-known social networks

All big social networks have their API that allows third-party applications to use social networks using predefined classes and methods [Russell, 2013]. In our system we actually need three types of queries:

1) find person by name;

2) get all posts and photos of given person;

3) get person's friends.

As we designed it to find alumni, then item 3 becomes very handy. Alumni often add each other to their friends, so if we request his friends, probably we can find there another alumni that we are looking for. If we are looking for some group graduated then we can start from 1-3 persons and using social graph (Figure 1) to find the rest.

And we actually targeting at 3 types of information: 1) personal information (name, age, birth date, etc.); 2) posts, messages; 3) photos.

Also we can divide social networks by two groups: personal and work.

Major social networks APIs that we are using:

Social network	Personal information	Posts	Photos	Personal usage	Work usage
Facebook	+	+	+	+	+
Twitter	+	-	+	+	+
Google+	+	+	+	+	+
VK	+	+	+	+	-
LinkedIn	+	+	-	-	+
MoiKrug	+	-	-	-	+
YouTube	-	+	-	+	-
Instagram	-	-	+	+	-
GitHub	+	-	-	-	+

We created modules for each social network that uses API of this network. All this modules have the same program interface, so we can simply query all social networks one-by-one. When user queries for some person, our system responds with results from all data sources plus our own crawler results.

Named entity recognition

To search people over the Web we implemented named entity recognition subsystem. It can find people names on web pages. Named entity recognition involves the supervised training of a statistical model or more direct methods like dictionary matching or regular expression matching. We use named entity recognition algorithms to recognize names in texts that processing. The ability to recognize previously unknown entities is an essential part of our system. Most of existing systems are rule-based. It means that recognition is limited to pre-defined rules [Nadeau, 2007]. So such system should be able to add new rules. Basically, there are three learning methods to add new rules:

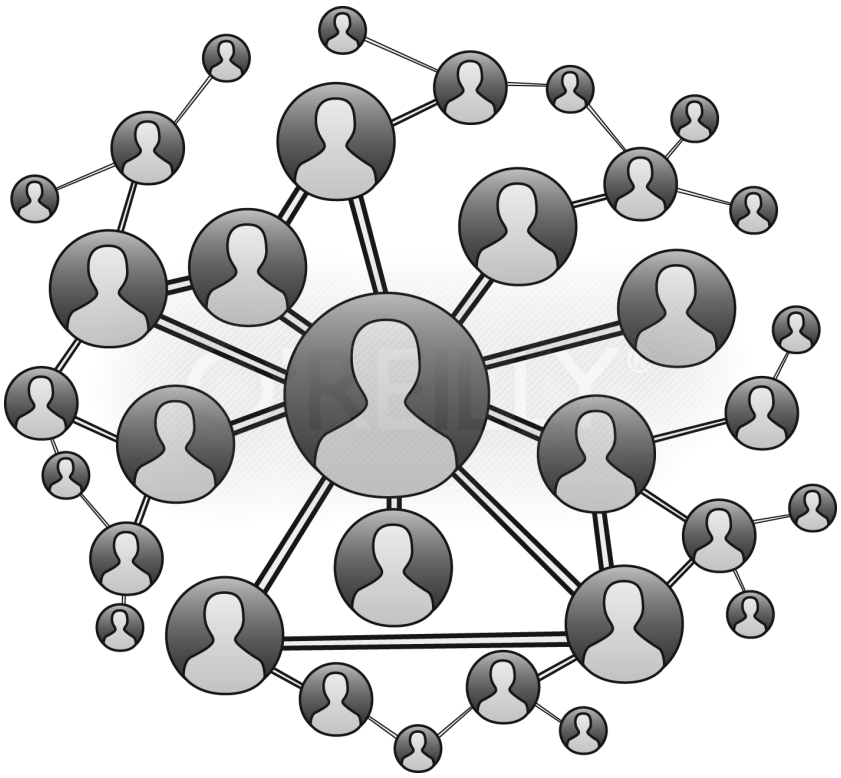

Figure 1:

1. Supervised learning

The idea of supervised learning is to study the features of positive and negative examples of named entity over a large collection of annotated documents and design rules that capture instances of a given type.

This method is the simplest, but it requires significant human resources and it is not very suitable for automatic system. In our system we use this method, but only when we need to make some corrections in our database (i.e. add non-common name).

2. Semi-supervised learning

It is relatively recent method. It also requires human interaction, but only at start. You need to provide starting data, like some common names. And then system will try to find similar names and add it to the database. When new names were found, system looks further for new names, but with bigger database, so it is able to find more new names. Function that extracts name parts is shown on Figure 2. It expects to have array of initial names (i.e. ["John", "Bill", "Mike", "Gates", "Queen"]) in input and text to extract names.

```
function extractNamesParts(text, startNames) {
    var words = text.split(' ');
    var newNames = [];

    for (var i = 0; i < words.length; i++) {
        if (startNames.indexOf(words[i]) === -1) {
            continue;
        }

        if (i > 0) {
            if (/^[A-Z]/.test(words[i-1])) {
                newNames.push(words[i-1]);
            }
        }

        if (i < words.lengh - 1) {
            if (/^[A-Z]/.test(words[i+1])) {
                newNames.push(words[i+1]);
            }
        }
    }

    return newNames;
}
```

Figure 2:

In addition to some lexical rules this method becomes quite powerful. For example, if phrase starts with Mr. then probably after goes name (Figure 3). Or if word ends on -ov, -ko then there is a possibility that this word is family name in some post-soviet countries. Figure 4 shows algorithm of extraction of such last names.

We use this method in out application and it gives satisfactory results. The good thing is that we already have lists of alumni from universities, so it is good starting point for this method.

3. Unsupervised learning

This method is based on clustering. It makes rules based on information gathered from clusters of information. This methods are deeply dependant on natural language analysis and forming of universal rules that are suitable in all situations.

Recognition and storage of personal information

Luckily, personal information is well-structured, so it can be easily recognized. Every person has such properties as name, age, birth date, live place, etc [Charu, 2011]. All this properties are well-described in open knowledge bases. We use FreeBase to get this information. We are using NodeJS as primary

```
var PREFIXES = [ 'Mr.', 'Mrs.', 'Ms.', 'Dr.' ];

function extractTitledNames(text) {
    var words = text.split(' ');
    var names = [];

    for (var i = 1; i < words.length; i++) {
        if (!/^[A-Z]/.test(words[i])) {
            continue;
        }

        if (PREFIXES.indexOf(words[i - 1])) {
            names.push(words[i]);
        } else if (i > 1 && names.indexOf(words[i - 1])
                  && PREFIXES.indexOf(words[i - 2])) {
            names.push(words[i]);
        }
    }

    return words;
}
```

Figure 3:

```
var POSTFIXES = [ /ko$/, /ov$/, /ev$/ ];

function extractPostSovietLastNames(text) {
    var words = text.split(' ');

    return words.filter(function (word) {
        if (!/^[A-Z]/.test(word)) {
            return word;
        }

        for (var i = 0; i < POSTFIXES.length; i++) {
            if (POSTFIXES[i].test(word)) {
                return true;
            }
        }

        return false;
    });
}
```

Figure 4:

```
var freebase= require('freebase');
freebase.description('person', {})
```

Figure 5:

language in our system and it has freebase package that allows us easily query FreeBase (Figure 5). Non-recognized information is also saved to a database and it is reviewed by supervisor.

We expecting that property title would be close in text to property value. Currenly our system recognizes such formats:

- Exact match
  Property name: Property value
- Exact match with HTML tags
  Property name: Property value
  *<h4>Property name</h4> Property value *
- HTML tables
  <table> <tr> <td>Property name 1</td> <td>Property name 2</td> </tr> <tr> <td>Property value 1</td> <td>Property value 2</td> </tr> </table>

Property name 1	Property value 1
Property name 2	Property value 2

Once information was recognized we store it at SQL database. SQL fits good for our purposes because data is well-structured. After data appeared in our database it becomes very simple to find any person. You need only to write SQL query.

Conclusion

In this paper we researched models, methods and approaches that we are using in our social search engine. We researched which data we can extract from social networks and how we can use it. We also described methods and algorithms of named entity recognition and recognition of names in texts. We also described approach of recognition of personal information using knowledge bases.

Bibliography

[Charu, 2011] Charu C. Aggarwal. 2011. Social Network Data Analytics (1st ed.). Springer Publishing Company, Incorporated.

[Nadeau, 2007] Nadeau, D. Sekine, S. (2007). A survey of named entity recognition and classification. Linguisticae Investigationes, 30, 3–26.

[Russell, 2013] Russell, M. A. (2013). Mining the Social Web, 2nd Edition. O'Reilly Media.

Authors' Information

Oleksandr Kuzomin *- Prof. Dr., Department of Informatics, Office of Innovations & Marketing, Kharkiv National University of Radio Electronics; Kharkiv, Ukraine; tel.: +38(057)7021515;*

e-mail: oleksandr.kuzomin@gmail.com

Major Fields of Scientific Research: General theoretical information research, Decision Making, Emergency Prevention, Data Mining, Business Informatics

Mariia Tkachenko *- PhD degree student, Kharkiv National University of Radio Electronics; Kharkiv; Ukraine; e-mail: mariia@tkachenko.io*

Major Fields of Scientific Research: General theoretical information research, Knowledge Discovery and Engineering, Data Mining, Business Informatics