Name Context Extraction for Web Pages

Wei Enxun & Sui Qingyu

SJTU ADAPT Lab

2011-11-02

Section 1

Introduction

Introduction

Nowadays, the Internet has become the largest information source for people. We use search engine to find web pages about *entities*, which means name of companies, people, organizations, and so on.

But for many web pages, it's always not true that the whole page is talking about one entity. It will help if we extract the context which related to some certain entity.

Introduction (cont'd)

Name Context Extraction aims to extract the context related to an named entity from a web page. It could be a preprocessor for search engine or name disambiguation to improve their performance.

We need a web page and a named entity as input, and output some plain-text in the web page which are related to the named entity.

Section 2

Algorithm

Overview

The *Name Context Extraction* algorithm is consisted of these four parts:

Overview

The *Name Context Extraction* algorithm is consisted of these four parts:

- Vision-based Page Segmentation

Overview

The *Name Context Extraction* algorithm is consisted of these four parts:

- Vision-based Page Segmentation
- Block Semantic Vector Extraction

Overview

The *Name Context Extraction* algorithm is consisted of these four parts:

- Vision-based Page Segmentation
- Block Semantic Vector Extraction
- Block Semantic Similarity Calculation

Overview

The *Name Context Extraction* algorithm is consisted of these four parts:

- Vision-based Page Segmentation
- Block Semantic Vector Extraction
- Block Semantic Similarity Calculation
- Block Selection

Page Segmentation by VIPS

In this part, we need to break the web page into several blocks, each block talking about one topic.

Actually, we simply use **VIPS** to split the web page. We could control the block size by modify the granularity parameter.

Some Page Segmentation Examples

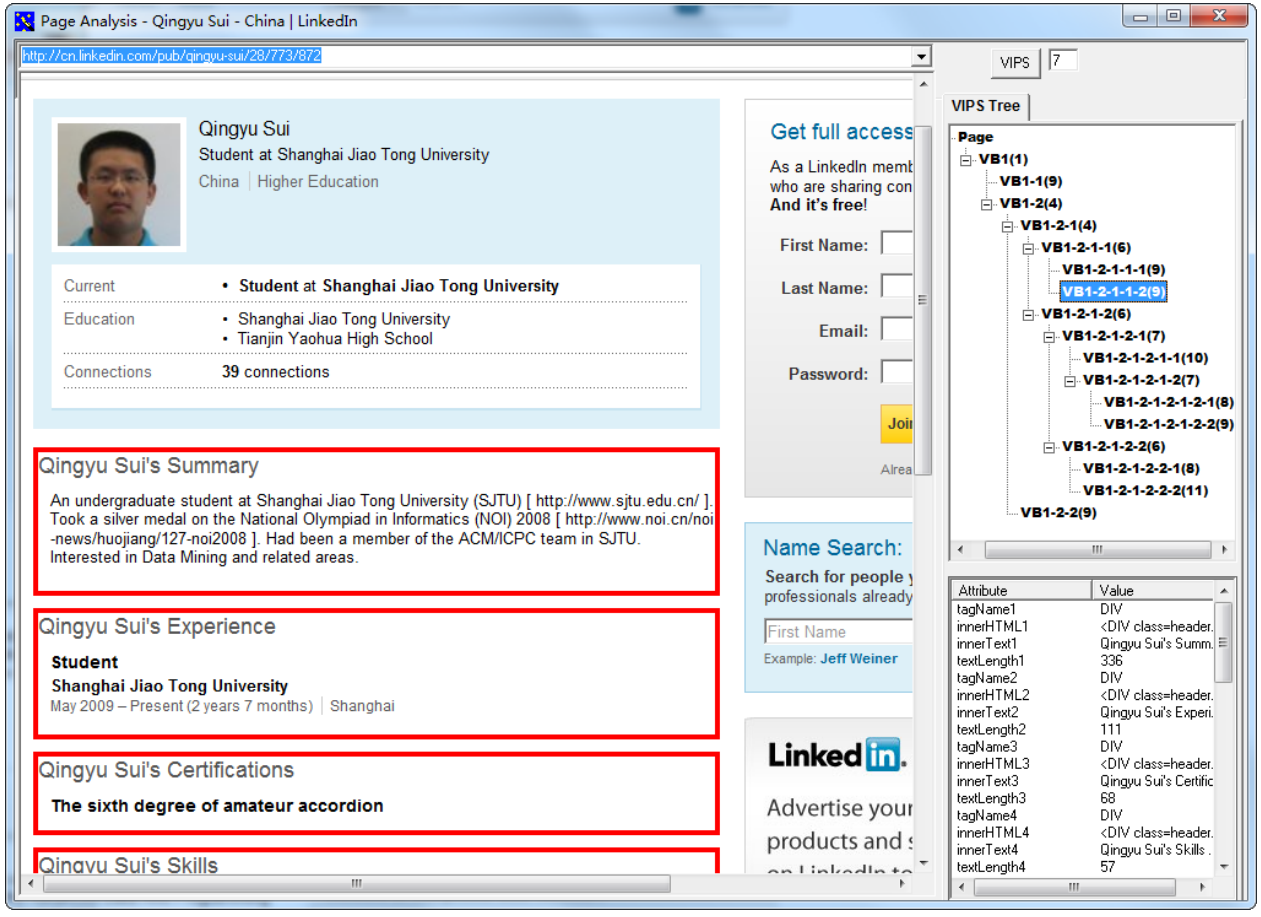

Figure: Page Segmentation by VIPS

Wikipedia Knowledge Base

Wikipedia Knowledge Base is a knowledge base consists of many concepts, each concept correspond to a page on Wikipedia.

The knowledge base can be described as a directed graph, and there is an edge from concept A to concept B only if concept A appears in the article of concept B.

Concept in Wikipedia Knowledge Base

Each concept has two properties: *rank* and *concept vector*.

- *Rank* means the popularity of a concept, or simply considered as Google PageRank on the graph described below.
- *Concept vector* of concept C is a vector indicates the set of concept which C appears in their Wikipedia pages.

Calculate Rank of Concepts

We could use the famous formula, which is also used to calculate Google PageRank:

$$R = dMR + (1 - d)V$$

since there is no difference between the concept graph and the PageRank graph.

Calculate Concept Vector of Concepts

The *concept vector* V_C is a vector of concept C, with each dimension correspond to a concept in the Wikipedia knowledge base.

In the vector V_C, the non-zero dimensions correspond to the concept which C "points to" in the graph, other dimensions are all zeros.

In the concept vector $V_C = (C_1, C_2, \ldots, C_n)$,

$$V_C(C_i) = R(C) \cdot P_{C,C_i}$$

where $R(C)$ is the rank of concept C, and P_{C,C_i} is the transitive probability from C to C_i.

In other words, if there is an edge from C to C_i, we have a $R(C)$ in dimension C_i of vector V_C, otherwise 0 in this dimension.

Block Vector

As *concept vector* indicates the relationship between a single concept and all concepts in the knowledge base, *block vector* is a similar vector which indicates the relationship between a single block and all concepts.

Concept Selection from Block

However, before calculate the block vector, we should pick up the concepts we could understand in this block, which consist a set T.

We implemented a simple algorithm here. First of all, as the block is HTML structured, we extract all text information S from the block. Here S is consisted of several words split by space, that:

$$S = (W_1 W_2 \dots W_n)$$

Concept Selection from Block (cont'd)

Then we run the code below to extract concepts in S, with the output T.

```
T ← ∅
// i points to the end of current phrase
for i ← n ... 1:
    // j points to the start of current phrase
    for j ← 1 ... i:
        if S[j...i] exists in KnowledgeBase:
            T ← T + S[j...i]
            // let i skip current word
            i = j
```

At last, we will have block concept set T, including a lot of concepts in the knowledge base.

Concept Extraction Example

For example:

$$S = (\textit{The quick brown fox jumped over a lazy dog})$$

And the knowledge base KB have these concepts:

$$KB = \{(\textit{dog}), (\textit{lazy dog}), (\textit{lazy}), (\textit{jumped over}), (\textit{jumped}), (\textit{fox}), (\textit{brown fox}), (\textit{quick brown})\}$$

Concept Extraction Example

For example:

$$S = (\textit{The quick brown fox jumped over a lazy dog})$$

And the knowledge base KB have these concepts:

$$KB = \{(\textit{dog}), (\textit{lazy dog}), (\textit{lazy}), (\textit{jumped over}), (\textit{jumped}), (\textit{fox}), (\textit{brown fox}), (\textit{quick brown})\}$$

Then the block concept set T is:

$$T = (\textit{lazy dog}), (\textit{jumped over}), (\textit{brown fox})$$

We find that the word “quick brown” wasn’t extracted, because “brown fox” had been extracted firstly.

Block Vector

Now we have a block concept set T that:

$$T = C_1, C_2, \ldots, C_n$$

and each concept C_i has a concept vector V_{C_i}. So the block vector BV is defined as:

$$BV = V_{C_1} + V_{C_2} + \cdots + V_{C_n}$$

Cosine Similarity of Vectors

The Euclidean Dot Product formula tells us:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \, \cos\theta$$

So we could calculate $cos\theta$ easily:

$$\cos\theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|}$$

The result is called *cosine similarity* of the two vectors, ranged from -1 to 1.

Block Similarity

With the block vector we had before, we could calculate the cosine similarity between two blocks.

Notice that block vector are always non-negative, so the cosine similarity ($\cos\theta$) is also non-negative, ranged from 0 to 1.

Cluster Analysis

Clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar to each other than to those in other clusters.

There are several different algorithms for clustering, such as **Hierarchical clustering**, **K-means clustering** and **Affinity Propagation**. In our project, we choose **Affinity Propagation** as our clustering algorithm.

Block Selection

Firstly, we run AP on the similarity matrix between blocks which we got before. Then we will have several disjoint clusters, each containing some blocks.

After that, we pick up all clusters which has at least one block including the named entity. Text in these clusters are treated as output.

Section 3

Performance

Performance of Name Context Extraction

We have a demo to show the result of our name context extraction algorithm.

Some examples:

Demo for Name Context Extraction

Figure: Name Context Extraction Demo

Demo for Name Context Extraction (cont'd)

Figure: Name Context Extraction Demo

Performance over GRAPE

GRAPE is a name disambiguation program, which works on **WEPS'07** dataset.

We use name context extraction as the preprocessor for GRAPE, comparing with the original algorithm. The result is shown below:

Algorithm	Purity	Inverse Purity	F_0.5	F_0.3
Original GRAPE	0.73	0.77	0.73	0.74
GRAPE with NCE	0.75	0.79	0.76	0.77

Table: Performance of GRAPE with Name Context Extraction

Thanks for Watching

Thanks for your watching!