Mining information from resume

Yogesh H. Kulkarni

Summary

This article demonstrates a framework for mining relevant entities from a text resume. It shows how separation of parsing logic from entity specification can be achieved. Although only one resume sample is considered here, the framework can be enhanced further to be used not only for different resume formats, but also for documents such as judgments, contracts, patents, medical papers, etc.

Introduction

Majority of world's unstructured data is in the textual form. To make sense of it, one must, either go through it painstakingly or employ certain automated techniques to extract relevant information. Looking at the volume, variety and velocity of such textual data, it is imperative to employ Text Mining techniques to extract the relevant information, transforming unstructured data into structured form, so that further insights, processing, analysis, visualizations are possible.

This article deals with a specific domain, of applicant profiles or resumes. They, as we know, come not only in different file formats (txt, doc, pdf, etc.) but also with different contents and layouts. Such heterogeneity makes extraction of relevant information, a challenging task. Even though it may not be possible to fully extract all the relevant information from all the types of formats, one can get started with simple steps and at least extract whatever is possible from some of the known formats.

Broadly there are two approaches: linguistics based and Machine Learning based. In "linguistic" based approaches pattern searches are made to find key information, whereas in "machine learning" approaches supervised-unsupervised methods are used to extract the information. "Regular expression" (RegEx), used here, is one of the "linguistic" based pattern-matching method.

Framework

A primitive way of implementing entity extraction in a resume could be to write the pattern-matching logic for each entity, in a code-program, monolithically. In case of any change in the patterns, or if there is an introduction of new entities/patterns, one needs to change the code-program. This makes maintenance cumbersome as the complexity increases. To alleviate this problem, separation of parsing-logic and specification of entities is proposed in a framework, which is demonstrated below. Entities and their RegEx patterns are specified in a configuration file. The file also specifies type of extraction method to be used for each type of the entity. Parser uses these patterns to extract entities by the specified method. Advantages of such separation is not just maintainability but also its potential use in other domains such as legal/contracts, medical, etc.

Entities Specification

The configuration file specifies entities to be extracted along with their patterns and extraction-method. It also specifies the section within which the given entities are to be looked for. Specification shown in the textbox below, describes meta data entities like Name, Phone, Email, etc. Method used to extract them is "univalue_extractor". Section within which these entities are to be searched is named "", it's a non-labelled section, like the initial few lines of the resume. Entities like Email or Phone can have multiple regular-expressions patterns. If first fails then the second one is tried and so on. Here is a brief description of the patterns used:

- Name: Resume's first line is assumed to have the Name, with an optional "Name:" prefix.
- Email: Is a word (with optional dot in the middle) then "@", then a word, dot and then a word.
- Phone: Optional International code in bracket, then digit pattern of 3-3-4, with optional bracket to the first 3 digits. For India number, it can be hard coded to "+91" as shown in the next entry.

```
<Config-Specifications>
  <Term name="Metadata">
    <Method name="univalue_extractor" section="">
      <Name>(^(Name:\s*)?(.+))</Name>
      <Email>((\w+[.|\w])*@(\w+[.])*\w+)</Email>
      <Email>([A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4})</Email>
      <Phone>((\+\d+\s*)?(\(?\d{3}\)?\D+\d{3}\D+\d{4}))</Phone>
      <Phone>((?:\(?\+91\)?)?\d{9})</Phone>
      <Address>Address: (.+)</Address>
    </Method>
  </Term>
</Config-Specifications>
```

- Python's 'etree' ElementTree library is used to parse the config xml into internal dictionary.
- Parser reads this specifications' dictionary and uses it to find entities from the text resume.
- Once an entity is matched it is stored as the node-tag, like Email, Phone, etc.

Like Metadata described above, Educational qualifications can be searched with a config below:

```
  <Term name="Education">
    <Method name="section_value_extractor" section="EducationSection">
      <Secondary>10th,X,Matric,SSC</Secondary>
      <HigherSecondary>12th,XII,HSC</HigherSecondary>
      <Diploma>CDAC, PGDBM</Diploma>
      <Bachelors>BE,B.E.,BTech,B.Tech,BS,Mechanical,Instrumentation,Civil</Bachelors>
      <Masters>ME,MTech,M.E.,M.Tech,MS</Masters>
      <PhD>Doctorate,PhD,Ph.D.,Ph.D</PhD>
    </Method>
  </Term>
```

- Method "section_value_extractor" of the parser is to be used and within section "EducationSection". It finds value within a section just by matching given words.
- If parser finds any of the words, say "10th" or "X" or "SSC", then those are the values extracted for the entity "Secondary", describing Secondary School level education.
- If parser finds any of the words, say "12th" or "XII" or "HSC", then those are the values extracted for the entity "HigherSecondary", describing Higher Secondary School level education.

Segmentation

The sections mentioned in the above code snippets are blocks of text, labelled such as SummarySection, EducationSection, etc. These are specified at the top of the config file.

- Method "section_extractor" parses the document line-wise and looks for section headings.
- Sections are recognised by keywords used for its headings. Say, for SummarySection, keywords are "Summary", "Aim", "Objective", etc.
- Once the match is found, state of "SummarySection" is set and further lines are clubbed under it, till the next section is found.

- Once the new heading matches, the new state of the next section starts, so on.

```
<Term name="Sections">
  <Method name="section_extractor">
    <SummarySection>Summary, Objective, Goal, Aim</SummarySection>
    <SkillSection>Skill, Skills</SkillSection>
    <EmploymentSection>Employment, History, Work, Experience</EmploymentSection>
    <EducationSection>Education</EducationSection>
    <HonorsSection>Honors, Rewards, Accomplishments</HonorsSection>
  </Method>
</Term>
```

Results

A sample resume is shown below:

```
Yogesh Kulkarni

Address: E1-32 State Bank Nagar, NCL, Pashan, Pune, India
Mobile: +91 989 025 1406
Email: kulkarniay@gmail.com

Summary:
12+ years in QA automation in various domains. Exploring a Senior Technical role.

Skills:
            Certifications: Certified Scrum Master, ISTQB
            Automation: Selenium, Selenium Grid, Cucumber, Jenkins, Robot, AutoIT, SilkTest, JMeter
            Programming: Python, Perl, Java

Experience:
Sep 2015    QA Manager, QA Services, Pune, India.
Till Date   - Responsible for managing a QA team: planning, execution of releases. Team size: 4.
                                    - UI automation and REST API automation using Selenium and Cucumber.
                                    - Implemented Continuous Integration using Jenkins and headless browser.
                                    - Python and shell scripting to automate QA tasks.

May2013     Team Lead, Repblic Solutions, Pune, India.
Aug 2015    - QA lead. Team size: 12
                                    - Held the responsibility of planning and execution of releases in addition to automation.
                                    - REST API and UI automation using Robot Framework (Selenium).
                                    - Python scripting to automate QA of scripts which dealt with a huge data repository.

Dec 2006    Senior Member of Technical Staff, Nice Software Laboratory, Pune, India.
Apr 2009    - QA lead. Team size: 4.
                                    - Planning test cycles on various platforms via Virtual machines (VM Images on ESX server).
                                    - Designed a framework for the automation of the testcases using AutoIT.
Aug 2004    System Engineer, CFO Systems, Pune, India.
Sep 2006    - Initiated and led the automation effort for the product using SilkTest.Team size: 4.
                                    - Perl scripting for a Report generator,which consolidated and publishing test results.
                                    - Also, led the activity of Manual QA of a client server and web application.

Nov 2002    QA Engineer, Vigilant Technologies, Pune, India.
Jul 2004    - Testing of a banking call center application. Team size: 3.
                                    - Perl scripting fora report generator and post testing analysis module.
                                    - Took up the responsibility as a developer along with testing for a period of six months.
Jul 2000    QA Engineer, More Consultants, Pune, India.
Oct 2002    - Testing Banking, HealthCare applications. Automation using Rational Robot, SilkTest.

Education:
1999 -00 Diploma in Advanced Computing, CDAC, Pune, India, Distinction.
1994 -99 BE (Instrumentation Eng), DY College of Eng, Pune, India, First Class.
1994 12th, HSC, Bhopal Vidyapeeth, Pune, India, Distinction.
1992 10th, SSC, Madras Highschool, Pune, India, Distinction.

Honors:
Multiple recognitions in the form of Spot awards.
```

Entities extracted are shown below:

```
Final Result:
 [
   {
     "Metadata": {
       "Name": "Yogesh Kulkarni ",
       "Email": "kulkarniay@gmail.com",
       "Phone": "+91 989 025 1406",
       "Address": "E1-32 State Bank Nagar, NCL, Pashan, Pune, India "
     },
     "Education": {
       "Diploma": "CDAC 1999-00 ",
       "Bachelors": "BE, Instrumentation 1994-99 ",
       "HigherSecondary": "12th, HSC 1994",
       "Secondary": "10th, SSC 1992"
     },
     "Skills": {
       "Certifications": "Certified Scrum Master, ISTQB ",
       "QA": "Selenium, Selenium Grid, Cucumber, Jenkins, Robot, AutoIT, SilkTest, JMeter ",
       "Programming": "Python, Perl, Java "
     }
   }
]
```

Implementation of the parser, along its config file and sample resume can be found at github.

End note

This article demonstrates unearthing of structured information from unstructured data such as a resume. As the implementation is shown only for one sample, it may not work for other formats. One would need to enhance, customize it to cater to the other resume types. Apart from resumes, the parsing-specifications separation framework can be leveraged for the other types of documents from different domains as well, by specifying domain specific configuration files.

Tags

Text Mining, Regular Expressions, Python

Author's brief info

Yogesh H. Kulkarni, after working in the field of Geometric Modelling for more than 16 years, has recently finished a PhD in it. He got into Data Sciences while doing the doctoral research and wishes to pursue further career in it now. He is keenly interested in Text Mining, Machine/Deep Learning and primarily uses Python stack for implementations. He would love to hear from you about this article as well as on any such topics, projects, assignments, opportunities, etc.

LinkedIn profile: https://www.linkedin.com/in/yogeshkulkarni/
Email and Phone: (can be found somewhere in this article itself!!)