Project Report

INFORMATION EXTRACTION FROM JOB POSTINGS AND RECOMMENDING THE MOST SUITABLE JOB BASED ON SIMILARITY

Course: Natural Language Processing (CSCI-GA.2590)

Professor: Ralph Grishman

By:

Ritu Sirkanungo (N11251912)

Sarthak Jain (N19848774)

Acknowledgements

We would like to thank Professor Ralph Grishman for offering us the opportunity to work on this very interesting Natural Language Processing project. We are grateful to Professor Grishman for his guidance and wonderful advice throughout our project. We have learnt immensely from this project.

Further, we would like to thank NYU for providing us with this opportunity.

Contents

Introduction

We all know that Job Postings are cluttered with a lot of information. Applying for jobs can be a tedious process. A computer science student when applying for jobs has to read through the whole job posting (which in some cases can span through multiple pages) and search for the relevant skills and other details of the job. The student has to focus on requirements listed and determine if he/she satisfies those requirements, else applying to that job will basically result in waste of time and resources.

We aim to aid people in this process by analyzing the cluttered information and extracting the most relevant information out of a job posting; and presenting it to a user in a format they can scan quickly.

We focused on the domain of Software Engineer and jobs of similar types for ex. software developer, senior programmer, data scientist, quality assurance engineer and so on.

Further based on the extracted information, we aim to recommend the most suitable job (out of a list of jobs) to the user based on a similarity measure with their resume/profile.

Problem Statement

The Project aims to extract the most relevant fields out of the job posting using natural language processing techniques of information extraction.

We focused on extracting the following list of fields-

Technical Languages ex. Java, Python, SQL etc.

Job Titles ex. Sofware Engineer, Senior Programmer

Experience (required/preferred) ex. 1 years

Degree ex. Bachelors

Tools ex. Github, Windows

Location ex. New York

Company Name ex. Oracle, American Research Institute

Further we will recommend the most suitable job using the TF-IFDF and cosine similarity measurements.

Data Sets

Source: www.indeed.com

Training Data:

-Job Postings collected by querying Indeed.com using the job title Software Engineer, different Locations.

-The Training data also contains Indeed.com job postings of similar kind of job titles like Software Developer etc.

Validation Data:

-The validation set was also collected from Indeed.com in a similar way, but it contains different job postings.

Test Data:

-The test set was also collected from Indeed.com in a similar way, but it contains job postings which are different from the training set and the validation set.

Tool kits and Libraries used:

Stanford NER (CRFClassifier)

Stanford Core NLP (`classifiers/english.conll.4class.distsim.crf.ser.gz`),

TF-IDF and cosine similarity scikit-learn libraries for calculating document similarity

Jsoup, jaunt and Java for web scraping of data.

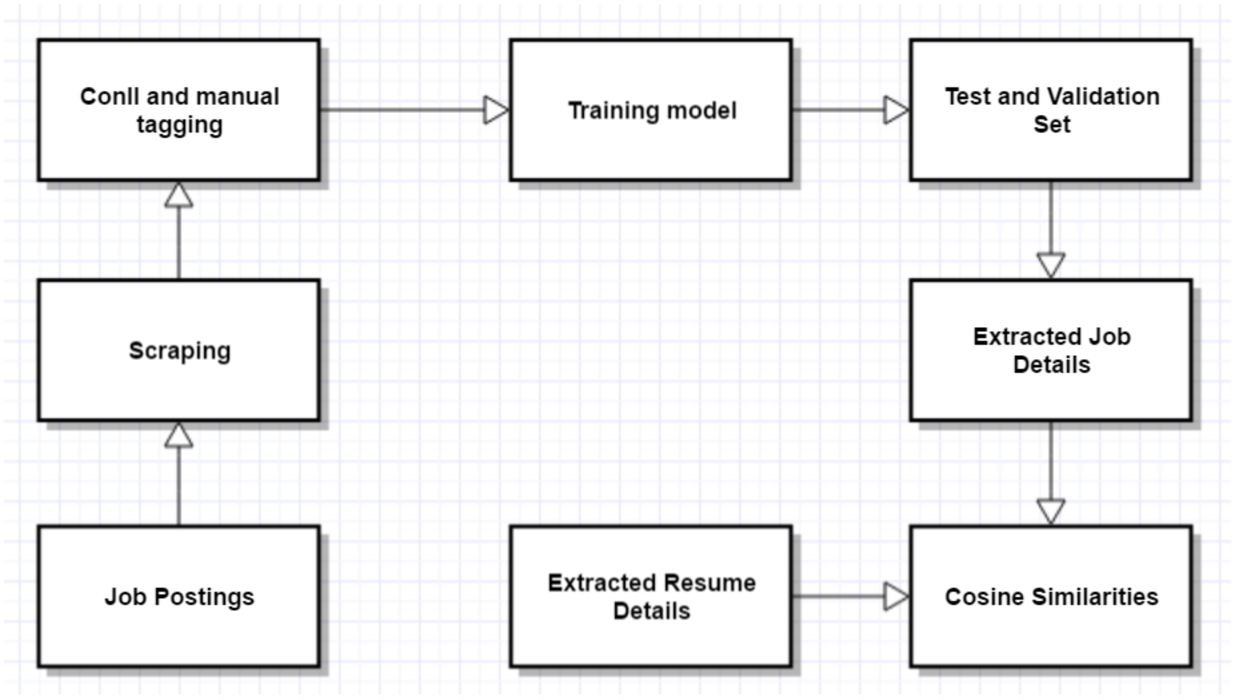
Java Eclipse Luna

Spyder

Note: Stanford NER uses Java 1.8 or higher

Methodology

Workflow:



Step 1:

-Procuring Data and Data Preprocessing/Data Cleaning

-We searched various sources for a good dataset of job postings, but could not find either a tagged or untagged dataset.

-We chose www.indeed.com as our data source. Indeed.com does not have an API for retrieving data.

-We wrote a web crawler/scraper in Java for scraping job postings from indeed.com

- We used the **jsoup** and **jaunt** libraries for the scraper.

Approach-

-Collecting URLS of job postings:

We tried to make an exhaustive list of locations containing- US States names, US state codes, US cities.

We iterated over the list of locations and queried indeed by generating queries of type:

`http://www.indeed.com/jobs?q=software+engineer&l=”+ tempLocation`

We collected all the links/URLS on the page which was the result of this query and on all the next pages to which this page had links.

-We now iterated over these collected URLS and retrieved the summary element from the page source using the code:

doc.getElementsByClass("summary")

-This gave us the text of the job posting. The text retrieved was devoid of any XML or HTML tags so we did not have to perform any further data preprocessing on it.

-We were able to collect roughly 15000 job postings

File containing code for the scraper:

NLPScrapingDataCleanClass.java

-We also extracted the fields **company** and **location** for each job posting and stored them in a different file as we thought they might be useful in cross validating the tagged output of the CONLL tagger.

Step 2:

Identifying training, validation and test Corpus

We used 1000 files for the training set.

We used 100 files each for the validation and test sets.

We were careful that there should not be an overlap between these 3 sets.

Step 3:

Procedure

-We used the Stanford NER(name entity recognition)-CONLL tagger for tokenizing and tagging the job summary files in order to prepare the training data set. At this stage we had a file for each job posting and we tagged them individually.

- Now we had the CONLL tagged files.

-Manual tagging using custom made tags generated by us. This was done on the same files using java code.

<u>MANUAL TAGS USED:</u>

LANG –for languages

TITLES – for job titles

EXP – for experience

DEGREE- for degree type

TOOLS – for tools

O – other tag

-Now we had the files with three columns-> **token CONLLtags manualTags**

- Now we merged the two types of tags.

-Approach for merging:

We replaced the CONLL tagged column with the manual tags based on the conditions: If there is a manual tag, final tag is set to the manual tag. If the manual tag is O and the CONLL tag is not O then the tag is CONLL tag.

-Now we had training files with two columns-> **token mergedTags**

All the training files were combined/appended to generate a single training file.

-Stanford ner Training

Trained ner over set of 1000 files (**token mergedTags**)

We generated a trained model by passing the training files (**token mergedTags**) to the **Stanford-ner-crf.CRF Classifier**

<u>Properties file for the classifier:</u>

```
#location of the training file

trainFile = file1000.train

#location where you would like to save (serialize to) your

#classifier; adding .gz at the end automatically gzips the file,

#making it faster and smaller

serializeTo = job1000-ner-model.ser.gz


#structure of your training file; this tells the classifier

#that the word is in column 0 and the correct answer is in
```

```
#column 1
map = word=0,answer=1


#these are the features we'd like to train with
#some are discussed below, the rest can be
#understood by looking at NERFeatureFactory
useClassFeature=true
useWord=true
useNGrams=true
#no ngrams will be included that do not contain either the
#beginning or end of the word
noMidNGrams=true
useDisjunctive=true
maxNGramLeng=6
usePrev=true
useNext=true
useSequences=true
usePrevSequences=true
maxLeft=1
#the next 4 deal with word shape features
useTypeSeqs=true
useTypeSeqs2=true
useTypeySequences=true
wordShape=chris2useLC
```

-The fields which are true are the features tried during the training.

-Validation

-We performed validation over a validation set of 100 files by passing them through the training model.

-To prepare the validation set we generated merged tags for the validation set using the CONLL and manual tagging.

ValidationSet file(**token mergedTags**)-> trained model -> file(**token mergedTags tagsPredictedByModel**)

We calculated the accuracy over the validation set by scoring the tags predicted against the mergedTags

Validation Problem and resolution:

To ensure the accuracy was correct, we had initially planned to evaluate the manually tagged files by hand, but due to time constraints and the large training set we could only perform the evaluation on approximately 50 files. We believe that the training set can be completely corrected if gone over by hand and this would increase the accuracy of the predicted tags.

-**Testing**

We used a test set of 100 files(**tokens**) -> applied trained model -> file (**token predictedTagsByModel**)

Now, After getting the predicted tags over the test set, we applied the process to get the mergedTags:

100 file(token) -> conll tagger(summary files)-> conll tag-> outputConll

manual tagger(conllTaggedOutput)-> files with three columns-> token conllTags manualTags

merging tags-> two column-> token merged tags

We then scored the predicted tags against the manual tags in the test files using the scorer.

Step 4:

Extraction of information from the tagged test files

We extracted information from the tagged test files as follows:

Extracting Languages : all tokens tagged as **LANG**

Extracting Tools : all tokens tagged as **TOOLS**

Extracting Degree: all tokens tagged as **DEGREE**

Extracting Job titles: all tokens tagged as **TITLES**

Extracting Experience: all tokens tagged as **EXP**

Extracting Location: all tokens tagged as **LOC**

Extracting Organization: all tokens tagged as **ORG**

Special Case:

We noticed that the CONLL tagger was tagging a lot of other tokens as organizations. We also noticed a pattern that singular tokens in the middle of the file were not the organizations, and the organizations usually came in a sequence or in the starting of the file. Using this pattern and a helper list of organizations which was collected, we extracted only those organization tags which matched these patterns; and whose at least one word matched an organization name in the helper file.

Using this we extracted the final organization information.

<u>Sitemap of files:</u>

Basic:

Scraping File

Basic Conll Tagging

Manual Tagging

Merge Basic Manual

Training : Files 1000 -> to generate a single file for all 1000 files

Files To Property -> to set the property for Stanford ner

Run Training File (shell sript) -> to train model

Run Validation File (shell script) -> for validation testing

valdation scorer -> to calculate score

run Testing File (shellp script) -> for accuracy on testing set

test Validation Scorer -> to calculate accuracy on testing model

cosine similarity (python file + scikit library for tf-idf and cosine similarity)

Extra files (Used as filters):

filter test set -> filter into token-tags format after tagging done by model

filter relevant tags -> for cosine similarity

form docs on extracted file -> create documents of extracted files for cosine similarity

Helper Files:

Languages -> for manual tagging

Titles -> for manual tagging

Tools -> for manual tagging

Company name list -> special case (extraction of organization names)

Evaluation and Results

Precision recall and f1 score in the model:

P, Precision,

R, Recall,

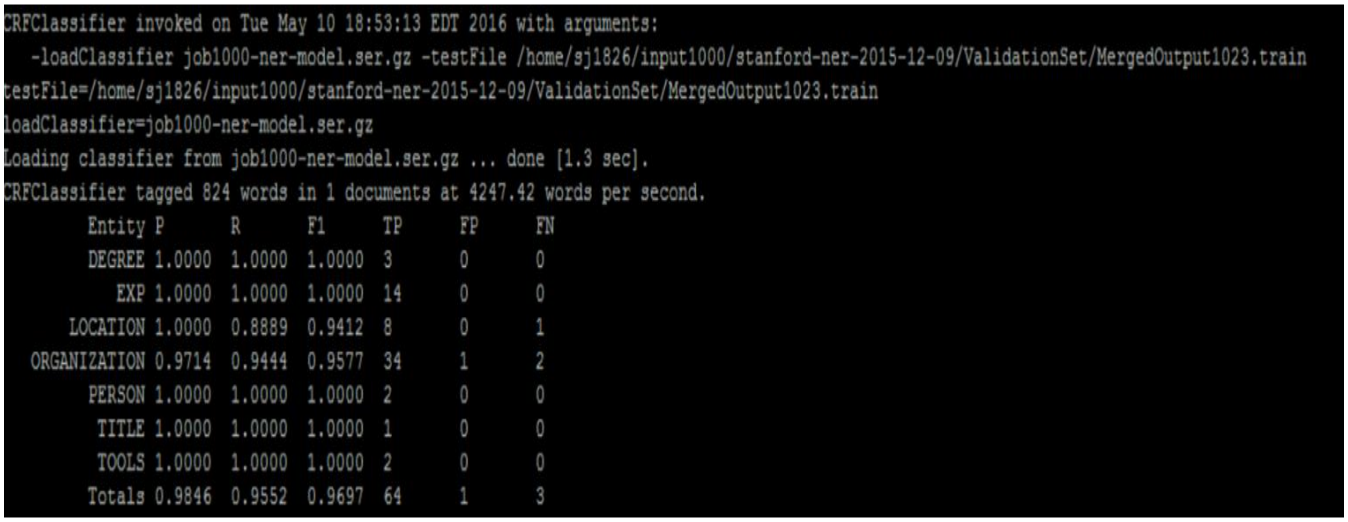
F1, F-score

TP, true positives, the ones it found which are correctly classified

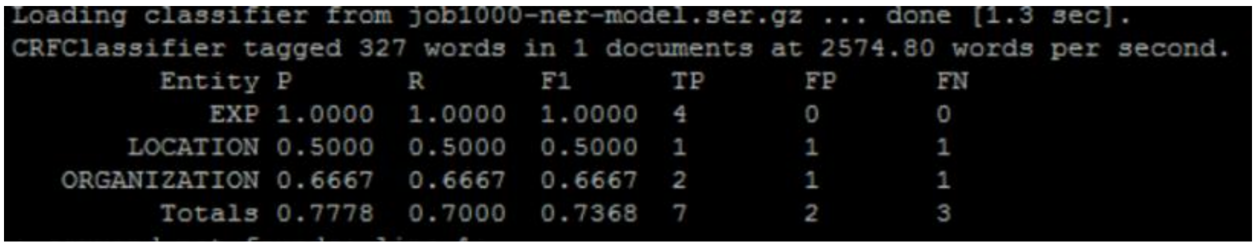
FP, false positives, the ones that it thought was entities, but it was not.

FN, false negatives, it didn't think these were entities, but they actually were

Screen shots of all the measures:



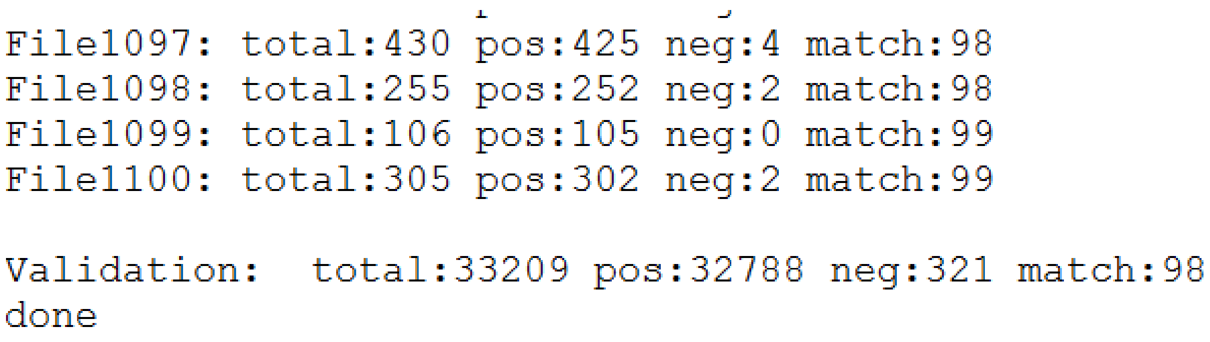
```
CRFClassifier invoked on Tue May 10 18:53:13 EDT 2016 with arguments:
   -loadClassifier job1000-ner-model.ser.gz -testFile /home/sj1826/input1000/stanford-ner-2015-12-09/ValidationSet/MergedOutput1023.train
testFile=/home/sj1826/input1000/stanford-ner-2015-12-09/ValidationSet/MergedOutput1023.train
loadClassifier=job1000-ner-model.ser.gz
Loading classifier from job1000-ner-model.ser.gz ... done [1.3 sec].
CRFClassifier tagged 824 words in 1 documents at 4247.42 words per second.
         Entity P       R       F1      TP      FP      FN
         DEGREE 1.0000  1.0000  1.0000  3       0       0
            EXP 1.0000  1.0000  1.0000  14      0       0
       LOCATION 1.0000  0.8889  0.9412  8       0       1
   ORGANIZATION 0.9714  0.9444  0.9577  34      1       2
         PERSON 1.0000  1.0000  1.0000  2       0       0
          TITLE 1.0000  1.0000  1.0000  1       0       0
          TOOLS 1.0000  1.0000  1.0000  2       0       0
         Totals 0.9846  0.9552  0.9697  64      1       3
```



```
Loading classifier from job1000-ner-model.ser.gz ... done [1.3 sec].
CRFClassifier tagged 327 words in 1 documents at 2574.80 words per second.
         Entity P       R       F1      TP      FP      FN
            EXP 1.0000  1.0000  1.0000  4       0       0
       LOCATION 0.5000  0.5000  0.5000  1       1       1
   ORGANIZATION 0.6667  0.6667  0.6667  2       1       1
         Totals 0.7778  0.7000  0.7368  7       2       3
```

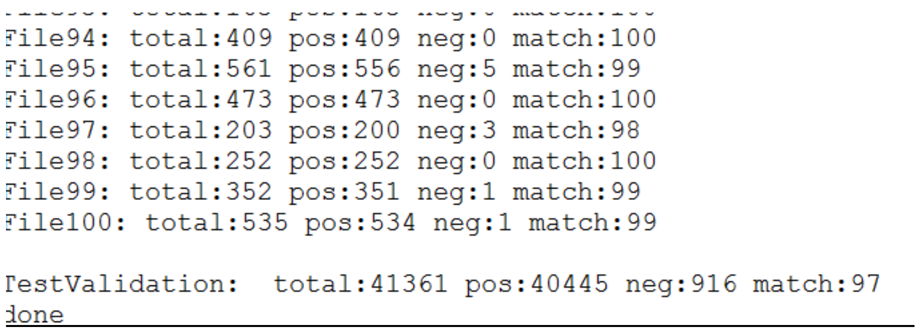
<u>For model trained with 1000 files:</u>

Validation socre on 100 validation files: 98%



```
File1097: total:430 pos:425 neg:4 match:98
File1098: total:255 pos:252 neg:2 match:98
File1099: total:106 pos:105 neg:0 match:99
File1100: total:305 pos:302 neg:2 match:99

Validation:  total:33209 pos:32788 neg:321 match:98
done
```

Score on test set of 100 files: 97%



```
File94: total:409 pos:409 neg:0 match:100
File95: total:561 pos:556 neg:5 match:99
File96: total:473 pos:473 neg:0 match:100
File97: total:203 pos:200 neg:3 match:98
File98: total:252 pos:252 neg:0 match:100
File99: total:352 pos:351 neg:1 match:99
File100: total:535 pos:534 neg:1 match:99

TestValidation:  total:41361 pos:40445 neg:916 match:97
done
```

<u>TFIDF and Cosine similarity</u>

-We calculated the TFIDF and cosine similarity using sci-kit learn libraries and Spyder.

The cosine similarity was calculated between some content which was extracted from a resume and some of the model tagged job postings from test set.

-An example of content extracted from resume and information extracted from job postings:

Extracted resume:

```
NAME: Nicole Johnson.

DEGREE: Master,degree.

LANGUAGES: Java,Perl,Adobe,SQL,Oracle,MySQL, HTML5, REST.

TOOLS:  Windows,Linux,Spring,  Subversion,  Apache  Tomcat,  MVC,  Visio,
AJAX, Bootstrap, MVC.
```

```
EXPERIENCE: 12 years of experience.
```

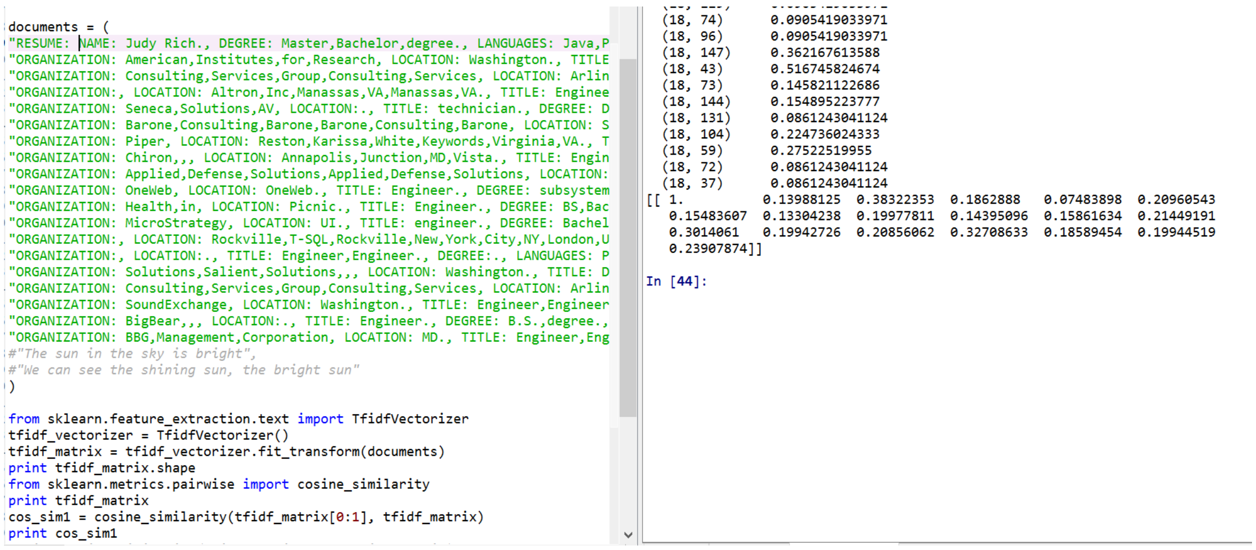
Information from job posting tagged by model:

```
ORGANIZATION: Consulting,Services,Group,Consulting,Services

LOCATION: Arlington,VA.

TITLE: Engineer,Engineer,Engineer.

DEGREE:
Master,degree,substitute,substitute,Bachelor,degree,substituted,degree
,degree,Master,Master.

LANGUAGES:
Java,Perl,Ruby,Python,HTML,C,C++,ColdFusion,Adobe,SQL,Oracle,MySQL.

TOOLS:
PTO,Windows,Linux,Apache,TomCat,Spring,JUnit,Hibernate,Agile,CVS,Eclip
se,Cucumber,Agile,SCRUM.

EXPERIENCE:
per,year,10,4,experience,/,/,experience,one,experience,one,experience,
one,Experience,Experience,Experience,Experience,12,more,years,experien
ce,two,2,years,experience,4,years,experience,12,years,experience,more,
years,Experience.
```

So, we basically tried to see information extracted from which job posting is most similar to the resume content. The job posting whose information has the highest similarity will be the most suited to the candidate and we recommend this job posting to the candidate. We calculate similarity measure with using tf-idf and cosine similarity.

We can see In the screenshot below that the file corresponding to the cosine similarity **0.383** is most similar to the resume.

An example of cosine similarity measurement:



```
documents = (
"RESUME: NAME: Judy Rich., DEGREE: Master,Bachelor,degree., LANGUAGES: Java,P
"ORGANIZATION: American,Institutes,for,Research, LOCATION: Washington., TITLE
"ORGANIZATION: Consulting,Services,Group,Consulting,Services, LOCATION: Arlin
"ORGANIZATION:, LOCATION: Altron,Inc,Manassas,VA,Manassas,VA., TITLE: Enginee
"ORGANIZATION: Seneca,Solutions,AV, LOCATION:., TITLE: technician., DEGREE: D
"ORGANIZATION: Barone,Consulting,Barone,Barone,Consulting,Barone, LOCATION: S
"ORGANIZATION: Piper, LOCATION: Reston,Karissa,White,Keywords,Virginia,VA., T
"ORGANIZATION: Chiron,,, LOCATION: Annapolis,Junction,MD,Vista., TITLE: Engin
"ORGANIZATION: Applied,Defense,Solutions,Applied,Defense,Solutions, LOCATION:
"ORGANIZATION: OneWeb, LOCATION: OneWeb., TITLE: Engineer., DEGREE: subsystem
"ORGANIZATION: Health,in, LOCATION: Picnic., TITLE: Engineer., DEGREE: BS,Bac
"ORGANIZATION: MicroStrategy, LOCATION: UI., TITLE: engineer., DEGREE: Bachel
"ORGANIZATION:, LOCATION: Rockville,T-SQL,Rockville,New,York,City,NY,London,U
"ORGANIZATION:, LOCATION:., TITLE: Engineer,Engineer., DEGREE:., LANGUAGES: P
"ORGANIZATION: Solutions,Salient,Solutions,,, LOCATION: Washington., TITLE: D
"ORGANIZATION: Consulting,Services,Group,Consulting,Services, LOCATION: Arlin
"ORGANIZATION: SoundExchange, LOCATION: Washington., TITLE: Engineer,Engineer
"ORGANIZATION: BigBear,,, LOCATION:., TITLE: Engineer., DEGREE: B.S.,degree.,
"ORGANIZATION: BBG,Management,Corporation, LOCATION: MD., TITLE: Engineer,Eng
#"The sun in the sky is bright",
#"We can see the shining sun, the bright sun"
)

from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
print tfidf_matrix.shape
from sklearn.metrics.pairwise import cosine_similarity
print tfidf_matrix
cos_sim1 = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)
print cos_sim1
```

```
  (18, 74)      0.0905419033971
  (18, 96)      0.0905419033971
  (18, 147)     0.362167613588
  (18, 43)      0.516745824674
  (18, 73)      0.145821122686
  (18, 144)     0.154895223777
  (18, 131)     0.0861243041124
  (18, 104)     0.224736024333
  (18, 59)      0.27522519955
  (18, 72)      0.0861243041124
  (18, 37)      0.0861243041124
[[ 1.          0.13988125  0.38322353  0.1862888   0.07483898  0.20960543
   0.15483607  0.13304238  0.19977811  0.14395096  0.15861634  0.21449191
   0.3014061   0.19942726  0.20856062  0.32708633  0.18589454  0.19944519
   0.23907874]]

In [44]:
```

Note:

Please see some examples of job postings and their extracted information further down in the report, as the problems and extensions are based on their observations.

Problems faced: Analysis and Resolutions

Lack of a proper training set. We could not hand check the training set completely so the generated training set might have had some wrongly tagged tokens.

Tools being tagged as organizations because in the training set they were not tagged as TOOLS. Resolutions: Make the list of tools exhaustive. Step completed and trained the model again.

Correctly extracting company names

Used generalized list for job titles during manual tagging which can be made more exhaustive.

Summary does not contain the organization name sometimes, so this field does not have any value in some extracted files.

Programming problem faced:

Out of heap space error as number of files to process are huge

Used HPC cluster to train and test the model as processing of files were consuming large amount of time on our machines.

Possible Extensions and Future Work

Experience can be matched with skills i.e some job postings list experience required according to skill(language or tool).

Proposed approach. Relate the LANG,TOOLS tags to the EXP tags when these come in the same sentence. This can be done during extraction.

Finding out difference between required and preferred qualifications.

Extracting more fields like salary.

Extracting strength of skill required.

For ex. basics of java, expert in java.

System is scalable, can be trained on a larger training set and classes/entity set can be increased in size.

More features can be tried to improve accuracy of model.

Some Sample job postings and their outputs

Sample 1:

Overview: The American Institutes for Research (AIR) is a leading professional services firm with a growing software engineering and product development team. We design and build things that are inspiring and make a real impact in the online testing industry. We are currently seeking a Contract Software Engineer to join our team in Washington, DC. Some of our ground-breaking work includes: advanced computer-adaptive algorithms (only one that's peer-approved in the country) mobile support for the user interfaces learning management systems with social media features user interfaces that are universally accessible to people with or without disabilities innovative, machine-scorable items Responsibilities: The Contract Software Engineer will be an integral part of the software engineering and product development team within AIR. This group of professionals provide custom software solutions for our clients as well as internal support systems. Much of the work we perform is new development, so the right candidate will have the skills needed to perform full life-cycle software development. This includes participation in requirements gathering, application and database design, system documentation, writing and unit-testing efficient code, and deployment. Qualifications: B.S. and/or M.S. in Computer Science.

Will consider other education if candidate has work experience that shows strong understanding of programming. At least 4 years of professional software development experience. 2-3 years professional experience utilizing C#, ASP.Net, Javascript and XML Experience supporting and troubleshooting applications programmed in C# and .NET web-based technologies and familiarity with deploying applications to IIS6 and IIS7 Experience writing SQL queries in SQL Server 2005/2008 Required experience: Professional Software Development: 4 years

Extracted Information 1 :

ORGANIZATION: American,Institutes,for,Research

LOCATION: Washington.

TITLE: Engineer,Engineer.

DEGREE: B.S..

LANGUAGES: C#,XML,C#,SQL,SQL.

TOOLS: ASP.Net.

EXPERIENCE: one,experience,4,years,experience,2-3,years,experience,Experience,Experience,experience,4,years.

Sample 2:

Consulting Services Group, LLC (www.csg-llcusa.com) Consulting Services Group (CSG) is a niche provider of intelligence related support services. Clearance Requirement: Active TS/SCI clearance is required. Benefits: Salary We offer competitive compensation packages with plenty of opportunity for advancement. Health Insurance CSG will pay 100% of the employee's health insurance premium. Family insurance options are available through payroll deduction. Dental Insurance CSG will pay 100% of the employee's insurance premium. Family insurance options are available. PTO (Paid Time Off) CSG grants a total of 80 PTO hours per year. We recognize 10 federal holidays (4 floating). Software Engineer – Fellow I Arlington, VA All levels of Software Engineer will have the skills listed below. Each level may have additional education, skill and/or experience requirements. The Software Engineer develops, maintains, and enhances complex and diverse software systems (e.g., processing-intensive analytics, novel algorithm development, manipulation of extremely large data sets, real-time systems, and business management information systems) based upon documented requirements. Software Engineering tasks include: Design and develop new software. Maintain existing software and resolves software problem reports. Modify existing software to add new features. Perform individually and as part of a team. Utilize software development and software design methodologies appropriate to the development environment and in compliance with customer and industry lifecycle development approaches. Develop, execute and document software testing, including test cases. Perform software requirement analysis, derivation and allocation. Provide input to the software components of system design to include hardware/software trade-offs, software reuse, application security, and use of Commercial Off-the-shelf (COTS)/Government Off-the-shelf (GOTS) in place of new development. Provide input to hardware (COTS/GOTS/capacity/etc.) recommendations. Prepare software documentation, including user guides. Required skills include: U.S. Citizenship required. Demonstrated experience and/or familiarity with one of more of the following: Java, Perl, Ruby, Python, HTML, C, C++, .NET, ColdFusion, Adobe, etc. Demonstrated experience and/or familiarity with one of more of the following: SQL, Oracle, MySQL, JBDC, etc. Demonstrated experience and/or familiarity with one of more of the following: Windows, Linux, Apache, TomCat, Ozone Widgets, Spring, Struts, JUnit, Hibernate, etc. Desired skills include: Experience with Agile development methodology. Experience with software configuration management and control tools (e.g., CVS, Eclipse, ClearCase, etc.) Experience with automated testing tools (e.g., RSpec, Cucumber, etc.) Experience: Twelve (12) or more years of software development experience required. [A Master’s degree in a related discipline may substitute for two (2) years of experience. A PhD may substitute for four (4) years of experience.] Bachelor’s degree in Software Engineering, Computer Science or a related discipline is required. [Twelve (12) years of experience (for a total of twenty-two (22) or more years) may be substituted for a degree.] Advanced degree (Master’s) is desired. Additional Responsibilities and/or skills: Develop and provide input to system architecture design. Act as subject matter expert (SME) to project. Additional Requirements: Experience leading multiple software development teams. Certifications: Agile software development methodology certification (e.g. SCRUM Master) highly desired . DoD 8570.1-M Compliance at IAT Level I (e.g., Certified Information Systems Security Professional (CISSP)) certification highly desired Consulting Services Group provides equal employment opportunities (EEO) to all employees and applicants for employment without regard to race, color, religion, sex, national origin, age, disability, or genetics. For additional information about Consulting Services Group or the position, please email John at jreidy@csg-llcusa.com.

Extracted Information 2 :

ORGANIZATION: Consulting,Services,Group,Consulting,Services

LOCATION: Arlington,VA.

TITLE: Engineer,Engineer,Engineer.

DEGREE:
Master,degree,substitute,substitute,Bachelor,degree,substituted,degree
,degree,Master,Master.

LANGUAGES:
Java,Perl,Ruby,Python,HTML,C,C++,ColdFusion,Adobe,SQL,Oracle,MySQL.

TOOLS:
PTO,Windows,Linux,Apache,TomCat,Spring,JUnit,Hibernate,Agile,CVS,Eclipse,Cucumber,Agile,SCRUM.

EXPERIENCE:
per,year,10,4,experience,/,/,experience,one,experience,one,experience,one,Experience,Experience,Experience,Experience,12,more,years,experience,two,2,years,experience,4,years,experience,12,years,experience,more,years,Experience.

Sample 3:

Job Description MicroStrategy is looking for a creative and talented engineer who is passionate about solving challenging problems and building world-class software. We are evolving our industry leading platform and we need a UI ninja who can take our BI self-services products to the next level. As part of a scrum team, you will work on developing the Tools our customer use to create compelling dashboards to analyze their data. Job responsibility Work with a highly motivated scrum team to deliver the next generation of MicroStrategy products Build high-performance and highly reusable front end components and system Contribute to code review and unit test Cooperate with other team members to achieve the highest productivity Proactively participate in every aspect of the entire software development lifecycle of feature development, including input on specifications, designs, implementation, test design, test implementation, optimization, and delivery. Qualifications Bachelor’s Degree, Master’s Degree in Computer Science, Software Engineering, or related field. Experience developing software applications and features in an Agile environment. Experience with performance benchmarking and optimization Must excel in a team-driven environment, while having the freedom to manage your workload independently. Work across the development stack; you may specialize in one core technology such as Java or JavaScript but you are comfortable learning new technologies and have skills in many different technologies to draw on. Experience working with GitHub, responsive design and RESTful APIs Proficient in Object-Oriented JavaScript programming Strong command of web standards, HTML5, CSS-layouts, DOM scripting, cross-browser compatibility & browser degradation strategies Good communication skills and the ability to work as part of a team. Job Type: Full-time

Extracted Information:

ORGANIZATION: MicroStrategy

LOCATION: UI.

TITLE: engineer.

DEGREE: Bachelor,Degree,Master,Degree.

LANGUAGES: Java,JavaScript,JavaScript,HTML5.

TOOLS: scrum,Agile,RESTful,APIs.

EXPERIENCE: Experience,Experience,one,Experience.

Sample 4:

Job Category: Full Stack Developer Position Title: Systems Engineer 4 Location: Washington, DC Description: Salient CRGT is looking for a Full Stack Systems Engineer to work in conjunction the Red Hat Open Shift Enterprise (OSE) 3.0 engineer during the architecture development, instantiation, and migration of pilot applications. This engineer will be a member of the Enterprise Architecture team within the Technical Solutions Office. CLEARANCE LEVEL: POSITION of TRUST/SUITABILITY REQUIREMENT: Please be aware that this position requires a U.S Government Public Trust/Suitability determination. Applicants who accept a conditional offer of employment may be subject to government security investigation(s) and must receive a favorable suitability determination for access to U.S Government information. JOB RESPONSIBILITIES: Help design, enhance, and maintain the CI and Continuous Delivery (CD) tooset with a focus on OSE. Work closely with infrastructure, networking and security teams in ongoing enhancement and support of OSE. Develop back-end software. Develop automated tests. Provide Tier 2 production support for OSE and associated applications. Participate in architecture reviews and technology standards definition. EDUCATION REQUIREMENTS: N/A CERTIFICATION REQUIREMENTS: N/A EXPERIENCE/SKILLS REQUIRED: Demonstrate prototyping as an integral and regular step in the design process of conceptualizing, refining, an inventive solution. Experience with build tools and continuous integration systems like Jenkins, Bamboo, etc. Experience with source code control tools like GIT, SVN, GitHub, etc. Strong experience with the full web stack (HTML5, CSS3, JavaScript Frameworks, etc), backend technologies and design patterns (Java 7, multithreading, servlets, Spring, Hibernate/ JDBC, RESTFul Frameworks, etc.). Experience with dependency management (Maven, NuGet, Gradle, Bower, etc.) Experience with one or more SQL databases required. Understanding of security requirements, OS hardening, penetration testing, and container isolation. Strong experience in Red Hat Linux and Windows administration. Experience actively developing and deploying applications. TRAVEL REQUIRED: N/A PHYSICAL REQUIREMENTS: N/A DESIRED QUALIFICATIONS : S. in Computer Science or related discipline Experience in software engineering, architecture, technical operations Red Hat Certified Architect (RHCA) or Red Hat Certified JBoss Developer (RHCJD) Strong Agile experience If you feel you are qualified for this position, please go to http://www.salientcrgt.com/careers /* *to apply. Salient Federal Solutions (Salient) is a leading provider of information technology, engineering and intelligence analysis services to agencies in the intelligence, defense, homeland security, and cyber domains. Salient is proud to be an Equal Employment Opportunity/AAP employer and maintains a Drug-Free Workplace. Salient prohibits discrimination against employees and qualified applicants for employment on the basis of race, color, religion, sex (including pregnancy), age, disability, marital status, national origin, veteran status, or any other classification protected by applicable discrimination laws. Salient also participates in E-Verify. Click here to learn about the E-Verify Program . For more information on Salient Federal Solutions, Inc., please visit us at www.salientcrgt.com . Required experience: Full Stack Development: 7 years

Extracted Information 4:

ORGANIZATION: Solutions,Salient,Solutions,,

LOCATION: Washington.

TITLE: Developer,Engineer,Engineer,engineer,engineer,Architect,Developer.

DEGREE:.

LANGUAGES: HTML5,JavaScript,Java,SQL.

TOOLS: Enterprise,Enterprise,Jenkins,GIT,Spring,Hibernate,JDBC,Maven,Linux,Windows,Agile.

EXPERIENCE: 4,2,Experience,Experience,experience,7,/,Experience,Experience,one,experience,Experience,Experience,experience,/,experience,7,years.

Sample 5:

BBG Management Corporation is looking to hire a Software Engineer in Ft.Meade, MD areas Expected Years of Experience: 2 years or more Clearance Requirement: An active TS/SCI w/Polygraph clearance Required Experiences: Minimum 2 years of experience as a Software Engineer, including software development, integration, requirement analysis, installation, enchantment, evaluation, problem diagnosis and testing Minimum 2 years of developing software for Windows and Linux operating systems Minimum 2 years of experience using Java programming language Extensive experience in developing and implementing test scripts plans Extensive experience in writing and generating technical documents for a project Knowledgeable about Subversion, REST service, Eclipse and Ant/Gradle Required Duties: Tackle customer's challenging problems related to big data Cooperate with teams to construct a system which ingests unstructured data, extracts entities, presents it for analytic, using UI and API Build Java software which ingest new data sources and process them. If you feel that your experiences make you a good candidate for this position, please contactAndrea Mills atandrea.mills@bbgmanagement.com

Extracted Information 5:

ORGANIZATION: BBG,Management,Corporation

LOCATION: MD.

TITLE: Engineer,Engineer.

DEGREE:.

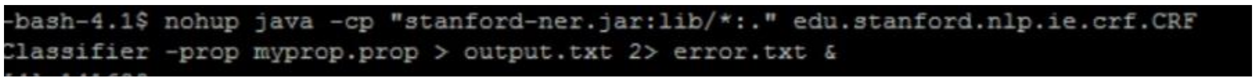
LANGUAGES: Java,REST,Java.

TOOLS: Windows,Linux,Eclipse.

EXPERIENCE: Experience,2,years,2,years,experience,2,years,2,years,experience,experience,experience.

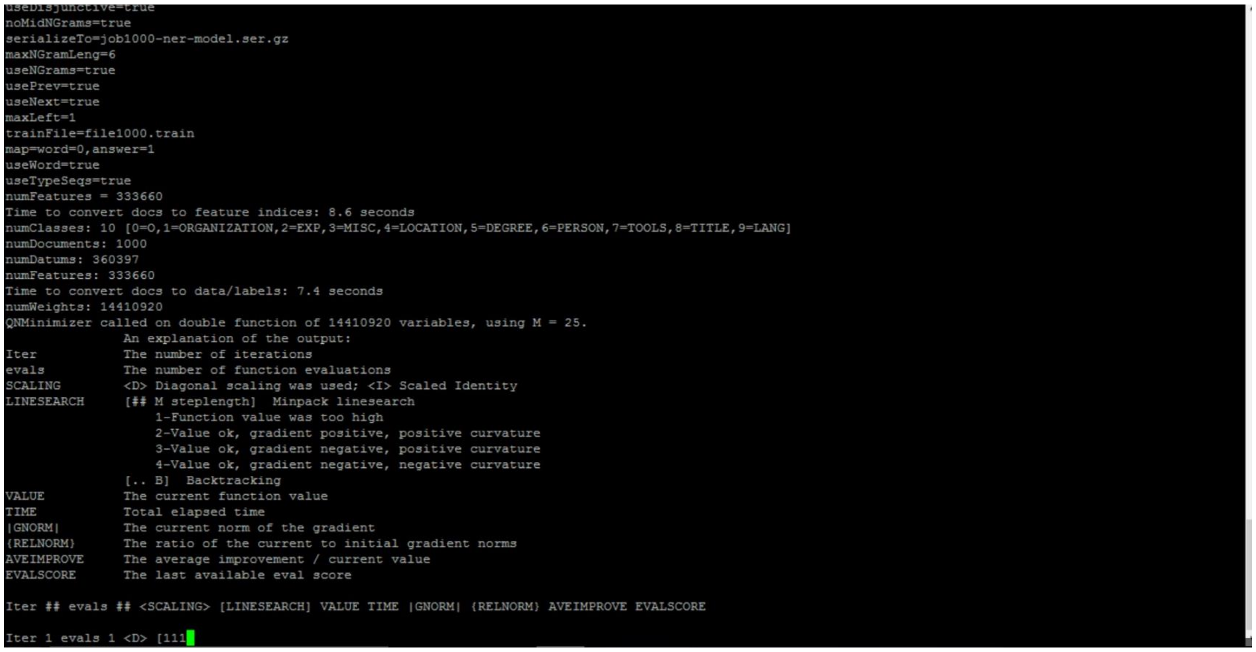
Some screenshots:

How to train the classifier:



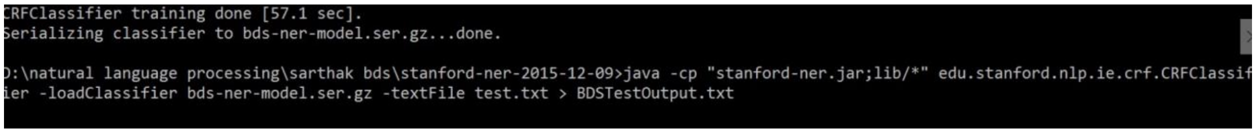
```
-bash-4.1$ nohup java -cp "stanford-ner.jar:lib/*:." edu.stanford.nlp.ie.crf.CRF
Classifier -prop myprop.prop > output.txt 2> error.txt &
```

Classes created when classifier trains the model using training set:



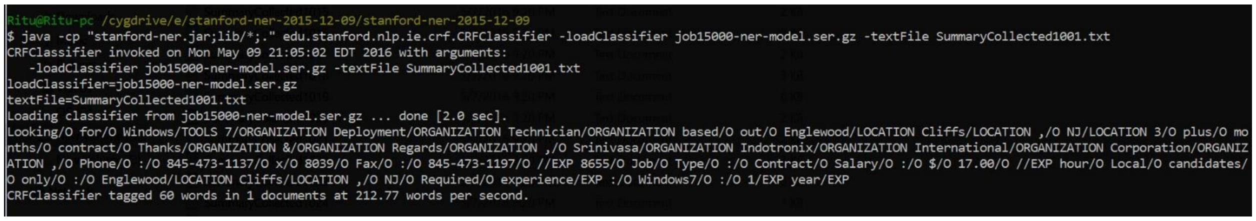
```
useDisjunctive=true
noMidNGrams=true
serializeTo=job1000-ner-model.ser.gz
maxNGramLeng=6
useNGrams=true
usePrev=true
useNext=true
maxLeft=1
trainFile=file1000.train
map=word=0,answer=1
useWord=true
useTypeSeqs=true
numFeatures = 333660
Time to convert docs to feature indices: 8.6 seconds
numClasses: 10 [0=O,1=ORGANIZATION,2=EXP,3=MISC,4=LOCATION,5=DEGREE,6=PERSON,7=TOOLS,8=TITLE,9=LANG]
numDocuments: 1000
numDatums: 360397
numFeatures: 333660
Time to convert docs to data/labels: 7.4 seconds
numWeights: 14410920
QNMinimizer called on double function of 14410920 variables, using M = 25.
               An explanation of the output:
Iter           The number of iterations
evals          The number of function evaluations
SCALING        <D> Diagonal scaling was used; <I> Scaled Identity
LINESEARCH     [## M steplength]  Minpack linesearch
                   1-Function value was too high
                   2-Value ok, gradient positive, positive curvature
                   3-Value ok, gradient negative, positive curvature
                   4-Value ok, gradient negative, negative curvature
               [.. B]  Backtracking
VALUE          The current function value
TIME           Total elapsed time
|GNORM|        The current norm of the gradient
{RELNORM}      The ratio of the current to initial gradient norms
AVEIMPROVE     The average improvement / current value
EVALSCORE      The last available eval score

Iter ## evals ## <SCALING> [LINESEARCH] VALUE TIME |GNORM| {RELNORM} AVEIMPROVE EVALSCORE

Iter 1 evals 1 <D> [111
```

How to test your file on trained model:



```
CRFClassifier training done [57.1 sec].
Serializing classifier to bds-ner-model.ser.gz...done.

D:\natural language processing\sarthak bds\stanford-ner-2015-12-09>java -cp "stanford-ner.jar;lib/*" edu.stanford.nlp.ie.crf.CRFClassif
ier -loadClassifier bds-ner-model.ser.gz -textFile test.txt > BDSTestOutput.txt
```

Tagging on one of the test inputs:



```
Ritu@Ritu-pc /cygdrive/e/stanford-ner-2015-12-09/stanford-ner-2015-12-09
$ java -cp "stanford-ner.jar;lib/*;." edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier job15000-ner-model.ser.gz -textFile SummaryCollected1001.txt
CRFClassifier invoked on Mon May 09 21:05:02 EDT 2016 with arguments:
   -loadClassifier job15000-ner-model.ser.gz -textFile SummaryCollected1001.txt
loadClassifier=job15000-ner-model.ser.gz
textFile=SummaryCollected1001.txt
Loading classifier from job15000-ner-model.ser.gz ... done [2.0 sec].
Looking/O for/O Windows/TOOLS 7/ORGANIZATION Deployment/ORGANIZATION Technician/ORGANIZATION based/O out/O Englewood/LOCATION Cliffs/LOCATION ,/O NJ/LOCATION 3/O plus/O mo
nths/O contract/O Thanks/ORGANIZATION &/ORGANIZATION Regards/ORGANIZATION ,/O Srinivasa/ORGANIZATION Indotronix/ORGANIZATION International/ORGANIZATION Corporation/ORGANIZ
ATION ,/O Phone/O :/O 845-473-1137/O x/O 8039/O Fax/O :/O 845-473-1197/O //EXP 8655/O Job/O Type/O :/O Contract/O Salary/O :/O $/O 17.00/O //EXP hour/O Local/O candidates/
O only/O :/O Englewood/LOCATION Cliffs/LOCATION ,/O NJ/O Required/O experience/EXP :/O Windows7/O :/O 1/EXP year/EXP
CRFClassifier tagged 60 words in 1 documents at 212.77 words per second.
```

References

http://cs.nyu.edu/courses/spring16/CSCI-GA.2590-001/TermProject.html

http://cs.nyu.edu/courses/spring16/CSCI-GA.2590-001/BagOfWords.pdf

http://cs.nyu.edu/courses/spring16/CSCI-GA.2590-001/MaxEnt.pdf

http://cs.nyu.edu/courses/spring16/CSCI-GA.2590-001/LearningToExtract.pdf

Speech and Language Processing, Daniel Jurafsky and James Martin, Prentice-Hall, Second edition, 2009

http://scikit-learn.sourceforge.net/stable/

Www.stackoverflow.com

www.scikit-learn.com

http://nlp.stanford.edu/software/crf-faq.shtml

http://www.informit.com/articles/article.aspx?p=2265404

http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/crf/CRFClassifier.html

https://www.youtube.com/watch?v=42NQ5ORqQIM

https://medium.com/@klintcho/training-a-swedish-ner-model-for-stanford-corenlp-part-2-20a0cfd801dd#.4eyjbi2ou

http://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction-tf-idf-part-i/