CS838 DATA SCIENCE PROJECT

Information Extraction

Tuan Dinh
dinh5@wisc.edu

Sam Gelman
sgelman2@wisc.edu

Varun Sah
varun.sah@wisc.edu

February 8, 2017

CONTENTS

SUMMARY

Documents

Source of documents	Amazon
Type of documents	Product descriptions (text)
Product category	Laptops
Entity type extracted	Brand Names Eg: Intel, Dell, Nvidia
Number of documents	373
Number of mentions marked up	1053

Set I : Training Set

Number of documents in set I	242
Number of mentions in set I	663

Set J : Test Set

Number of documents in set J	131
Number of mentions in set J	390

Classifier :

Classifier M : 1st 5-fold cross validation (CV)	Random Forests	
	Precision (CV average)	0.848261857986
	Recall (CV average)	0.852228425779
	F1 (CV average)	0.849845416441
Classifier M : debugging + 5-fold cross validation	Random Forests	
	Precision (CV average)	0.928561118375
	Recall (CV average)	0.679200642862
	F1 (CV average)	0.782959835822
Classifier X : run on complete set I	Random Forests	
	Precision (Set I)	0.934065934066
	Recall (Set I)	0.661478599222
	F1 (Set I)	0.77448747152
Classifier X : before postprocessing : run on set J	Random Forests	
	Precision (Set J)	0.921052631579
	Recall (Set J)	0.538461538462
	F1 (Set J)	0.67961165048
Post-processing	None	
Classifier Y : final classifier : run on set J	Random Forests	
	Precision	0.921052631579
	Recall	0.538461538462
	F1	0.67961165048

1 DOCUMENTS

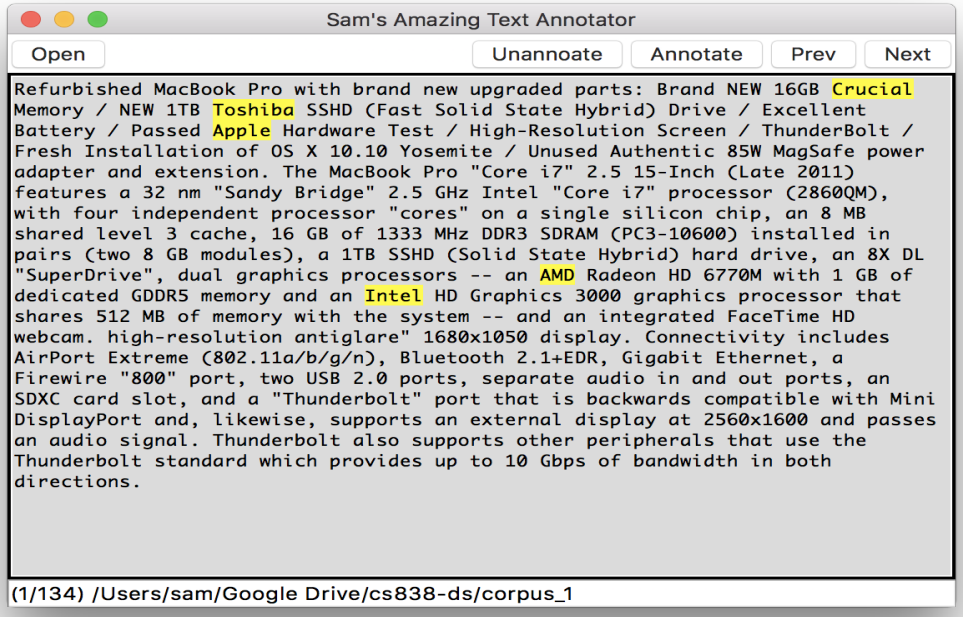
We collected product descriptions for 373 laptops from the Amazon website. We had written python scripts for crawling product descriptions during the data acquisition phase (stage 1) of the project.

1.1 ENTITY TO BE EXTRACTED

We decided to extract brand names from the documents. The brand names that we intended to extract included laptop manufacturing companies like Apple, Asus, Alienware, HP, Dell, Microsoft, Samsung, Lenovo, Toshiba etc. The brand names also included laptop hardware and accessory manufacturers like AMD, Intel, Nvidia, Waves, B&O, SteelSeries, Corning etc. An important distinction to be made is between brand names and product lines. For instance, we considered Microsoft a brand but not Surface Pro or Windows. Similarly, Nvidia is considered a brand, whereas GeForce is not.

1.2 ENTITY ANNOTATION

We wrote a python program that provided us a graphical user interface to make the task of annotating brand names within documents easier. Following is a screenshot of the annotator:



The annotator requires Python 3 to run. The user is expected to put all text files that need to

be annotated in a single directory. On running the annotator, the user is required to choose the directory which contains the files of interest. A word/phrase can be annotated by highlighting the word(s) and pressing the "a" key or clicking the "annotate" button. The annotations are saved in a separate .annot file placed in the same directory. The .annot file contains the start and end points of the annotations per line. The annotation is saved when the user clicks "next" or "prev" to move to the next or previous document in the directory.

2 Dataset Preparation

2.1 Feature Extraction

We computed the following 11 features for each positive and negative example.

- First letter uppercase (T/F)
- The entire word uppercase (T/F)
- Length of the word (Int)
- Part of a sequence of words, each of which starts with uppercase letters (T/F)
- Position of the word in the sequence of words that start with uppercase letters (Int)
- Next word has numerals (T/F)
- Number of occurrences of this word in text file (Int)
- All characters are alphabetic (T/F)
- Previous word is "the" (T/F)
- Previous word is "from" (T/F)
- Previous word is "by" (T/F)

We selected the features using a combination of intuition and experience from annotating the files. For example, feature 9 (Previous word is "the") was added after we saw that many company names were preceded by "the", as in "The Apple MacBook ..." and "The Samsung Galaxy S7..."

2.2 Generating Positive Instances

Generating positive instances was straightforward after we had labelled all the 373 documents of set I. We identified brand names using the coordinates mentioned in the <filename>.txt.annot files. We then computed the above features for each annotated word for each text file.This resulted in a total of 1053 feature vectors of positive instances.

2.3 Generating Negative Instances

We generated twice as many negative examples as positive examples, as was suggested in class. Negative examples were generated per file based on the number of positive examples within that file. For example, if there were three positive examples in a file, we generated six negative examples.

We realized that picking good negative instances, wherein some negative examples are deceptively similar to positive examples, was key to simulating a real-world scenario. We ensured a fair distribution of negative examples by using a probability based approach. Negative examples were selected with the following probabilities.

- 10% First letter uppercase
- 20% Neighboring words of annotated brand names
- 30% No numeric characters
- 40% Any other un-annotated words

Neighboring words and words with first letter uppercase were particularly important for our domain since brand names are often surrounded by capitalized names of products or product lines. By including them as negative instances, we ensured that the classifier's performance estimates provided in this report can potentially be extrapolated to be applicable in real-world applications.

3 Classification

Our aim was to create an acceptable classifier that could identify brand names. We defined the acceptability criteria on the basis of precision and recall thresholds that needed to be achieved. The thresholds were set at 90% for precision and 50% for recall.

3.1 Selecting the best classifier

To choose the most appropriate classifier M, we ran 5-fold cross validation on the following models:

- Linear Regression
- Logistic Regression
- SVM
- Decision Tree
- Random Forest

3.1.1 SELECTING INITIAL CLASSIFIER M

Classifier	*Precision*	*Recall*	*F1*
Linear Regression	0.676208916295	0.790956238645	0.728252674644
Logistic Regression	0.683920480008	0.810389077605	0.741089090163
SVM	0.737895741942	0.514278708592	0.605085398653
Decision Tree	0.794464022207	0.824951727279	0.809097754076
Random Forest	0.848261857986	0.852228425779	0.849845416441

We picked Random Forest as classifier M based on highest precision. But since Random Forest's precision value was not acceptable ($\geq 90\%$) we had to debug our methodology.

3.1.2 DEBUGGING CLASSIFIER M

- Debug 0: Vary the number of estimators of Random Forest (10, 30, 50, 100):
  - Result: choose 30
  - Precision: 0.86
  - Recall: 0.83
- Debug 1: Increase threshold to 0.8
  - Precision: 0.947058823529
  - Recall: 0.636363636364
- Debug 2: Remove the least important features of RandomForest
  - Feature importance : [0.18641615, 0.19031936, 0.21027666, 0.07001555, 0.11511065, 0.02940783, 0.08617387, 0.03997453, 0.02172997, 0.0043303, 0.00352656, 0.04271857]
  - previous_word_is_by: 0.00352656
  - previous_word_is_from: 0.0043303
  - Result: remove "previous_word_is_by"
  - No improvement in precision after Debug 1. Discard Debug 2.
- Debug 3: Investigate false positives and false negatives
  - Result: relabel some data to get rid of inconsistencies

Since some of the training data was modified at the end of Debug 3, we needed to re-run cross validation on all the models considered initially. We retrained all models using the best threshold 0.8 (Debug 1), 30 classifiers for random forests (Debug 0) and the newly labeled training data (Debug 3).

3.1.3 SELECTING FINAL CLASSIFIER M

Results of 5-fold cross validation after debigging were as follows:

Classifier	*Precision*	*Recall*	*F1*
Linear Regression	0.681111659507	0.790598613386	0.73123448801
Logistic Regression	0.679166705565	0.81933382087	0.742114089871
SVM	0.730119164833	0.524802751288	0.60942728957
Decision Tree	0.813479883162	0.818430210828	0.815535850267
Random Forest	0.928561118375	0.679200642862	0.782959835822

We picked Random Forest as the best classifier based on highest precision and proceeded to train it on the entire training set (without cross validation).

3.2 TRAINING THE CLASSIFIER X

We then trained our Random Forest classifier on all training-set instances (without cross-validation) and obtained the following results:

The 4 most important features were found to be:

- first_letter_uppercase
- word_len
- num_of_occurrences_of_word
- previous_word_is_the

Classifier	*Precision*	*Recall*	*F1*
Random Forest	0.934065934066	0.661478599222	0.77448747152

Since both the recall (0.661478599222) and precision (0.934065934066) of Random Forest classifier on the training set were acceptable ($\geq$ 50% and $\geq$ 90% respectively), Random Forest was confirmed as our classifier X, which was to be tested on test documents of set J.

3.3 TESTING THE CLASSIFIER X

Classifier	*Precision*	*Recall*	*F1*
Random Forest	0.921052631579	0.538461538462	0.67961165048

Since both the recall (0.538461538462) and precision (0.921052631579) of our Random Forest classifier on the test set were acceptable ($\geq$ 50% and $\geq$ 90% respectively), the classifier was confirmed to be our classifier Y. There was no requirement of exploring any post-processing rules.