CS784: Data Models and Languages

Project Stage II

Brand Name Extraction

Date: March 14, 2016

Group Member Information
Artsiom Hovarau: hovarau@cs.wisc.edu
Apul Jain: apul@cs.wisc.edu
Zhuowei Cai: zhuoweic@cs.wisc.edu

Website: http://pages.cs.wisc.edu/~zhuoweic/index.html

In this project stage we analyzed electronic product names and developed an extractor to extract brand names from the product names. For this task, we used dictionary-based approach. We randomly sampled 350 product item from a total of 10K products and split this set of items (Set S) into two sets: Set I and Set J. Set I consists of 200 product items while Set I consists of 120. Set I was used as a development set to develop our extractor while Set J was used to test how well our extractor behaves.

Extractor Algorithm:

In this stage, we used dictionary-based approach. Initially we are given a dictionary of 8442 brand names. We first expanded this dictionary with brand names extracted from the development set I. Our extraction algorithm consists of following steps:

Step 1: Data Acquisition:

Sample data to form set S. This data will be split into two sets - development set I and test set J. We sampled 350 records to form set S and then split it into 200 and 150 for Set I and Set J, respectively. We then manually extracted the brand names for each of the product names. After this step, we had records in the following format:

Format of product records:
`prod.id?prod_name?brand_name`

Note that for some records we couldn't figure out the brand names from the product name, these were labeled as records having missing brand names (represented by an empty string).

We also used the dictionary D of plausible brand names given to us.

Step 2: Analysis of brand names from the development set I:

Dictionary Expansion:
Parse the development set (Set I) and pull out the brand names. Merge these extracted brand names with our dictionary D.

Step 3: Brand name extraction:

Dictionary lookup:
Given a product name, we do a substring matching for each plausible brand name in the product name. If we find no match, then the brand name extracted is empty. If we have more than one collisions in the brand name, we infer its brand name using the rules in the precedence corresponding to their order. We will elaborate the rules that are used in the section below.

<u>Step 4:</u> Calculation of Precision and Recall

To calculate precision and recall, we defined False Positive (FP), True Positive (TP), True Negative (TN), False Negative (FN) as follows:

Actual brand name	Extracted brand name	Classification
ABC	ABC	True Positive (non-empty match with non-empty brand name)
ABC	XYZ	False Positive (non-empty mismatch with non-empty brand name)
“”	ABC	False Positive (non-empty mismatch with empty brand name)
ABC	“”	False Negative (empty mismatch with non-empty brand name)
“”	“”	True Negative (empty match with empty brand name)

Precision = $\frac{TP}{TP+FP}$

Recall = $\frac{TP}{TP+FN}$

<u>Results:</u>

Based on our algorithm, we saw following results:

Precision: 96.06%
Recall: 92.42%

Rules to handle multiple plausible brand names match:

Assumption: Brand name is a consecutive sequence of words. We made this assumption so that we can do substring matching between the brand name and the product name.
Rule 1: Brand name closer to the beginning first
The basic idea behind this rule is that we prefer brand name match which appears closer to the beginning of the product name.

Example:
`ASUS T100TAM Transformer Book Intel Atom 2GB Memory 64GB SSD 10.1 2-in-1 Brushed Aluminum Notebook Windows 8.1 + Micro?ASUS`

In this case as you can see, there are two plausible brand names based on dictionary lookup: `ASUS` and `Intel`. In order to correctly extract brand name, we must prefer `ASUS` and not `Intel`. Based on this rule, we see that `ASUS` appears before `Intel` so our algorithm would pick `ASUS` rather than `Intel`. This also follows our general observation that usually product names start with brand name when we go for shopping on e-commerce websites like Amazon or Walmart (since they are the most important attribute of a product).

`match_index(“ASUS”) < match_index(“Intel”)` ⇒ `brand_name = “ASUS”`

Rule 2: Longest matching brand name first:

This rule handles cases where brand names itself consists of multiple parts and each part individually can refer to a brand name in itself, which is complementary to Rule 1.

Example:
`11038058?Belkin Mobile Retractable USB Mouse - Black?Belkin Mobile`

For this example, we see that both Belkin and Belkin Mobile are valid brand names based on the dictionary lookup. Our algorithm would extract Belkin Mobile because its length is larger than length of Belkin alone.

`len(“Belkin Mobile”) > len(“Belkin”)` ⇒ brand_name = `“Belkin Mobile”`

Rule 3: Suffix rule:

We maintain a suffix array:

```
SUFFIXES = ["Inc.", "Corp.", "Incorporation", "Corporation", "Technology", "Ltd.", "Limited"]
```

This rule says that for each plausible brand name, search for brand_name + <suffix> in the product name, and if we find a match we return brand_name + <suffix> as the extracted brand name. This is essentially an extension of Rule 2.

Example:

`37241921#Perf-Moto?`**`Biltwell Inc.`**` Bonanza Solid Helmet Gloss Black LG?`**`Biltwell Inc.`**

For this product, we found only “`Biltwell`” in our dictionary. But based on the suffix rule, we would search for “`Biltwell + <suffix in SUFFIXES>`” and we will hit a match for “**`Biltwell Inc.`**” So our algorithm would return “`Biltwell Inc.`”

```
match(“Biltwell” + “ Inc.”) == true ⇒ brand_name = “Biltwell Inc.”
```

Analysis of Precision/Recall further improvement:

We see that precision and recall mainly depend upon False positives and False negatives. To understand effect of False positives/negatives on precision/recall, we observed two factors affecting our precision:

Reason for no-further-improvement in precision/recall:

1. Brand name mismatch due to multiple similar brand names (for precision):

Example: False positive due to wrong brand name prediction
`42397735#TigerDirect?Kingston ValueRAM - DDR3 - 16 GB - DIMM 240-pin - 1600 MHz / PC3-12800 - CL11 - 1.5 V - registered with parity - ECC?Kingston ValueRAM`

True brand name: `Kingston ValueRAM`
Predicted brand name: `Kingston`

In this case `Kingston` is present in our dictionary but not “`Kingston ValueRAM`”. So this results in false positive by our algorithm.

2. Brand name not found in dictionary (for recall first, then for precision):

Example: False negative due to missing brand name in dictionary

`20850274?Winslow TV Stand, for TVs up to 46, Espresso?Winslow`

True brand name: `Winslow`
Predicted brand name: `””`

In this case no brand name from our dictionary is present in the product name. So our algorithm fails to extract `Winslow` and results in false negative. In fact, we consider using a **non-brand-name dictionary** to get rid of the common words from the product name. But the size of the non-brand-name dictionary is huge, and it turns out that it does not work quite well.

Solution:
We finally came up with another idea that will significantly reduce the size of the dictionary we need and could possibly offer better performance (we did not have time to test this idea though). That is, to maintain a **product dictionary**. For example, this dictionary consists of the name of the product like

TV Stand
Case
Refrigerator
...

And usually the brand name will be the substring that immediately precedes the name of the product. So it is possible turn to the product dictionary in order to get ultimate performance and even pull up the above parameters in the future. This also seems to be quite a reasonable heuristics to use.

Appendix:

Code to illustrate our extraction rules: [Please refer to actual code for complete details]

```
# Extract brand name from product name.
# Expecting the input data (development) and test files
# containing record in the form of
#
# id?product_name?brand_name
# ...
# The second argument corresponds to the dictionary used.
# ---------------------------------------------------------
def extractor(datafile, dictfile, testfile):

  # Dictionary Expansion: merge brand names in the development set with the given dictionary
  merge(dictfile, datafile)

  # do the actual testing
  extracted_brand = list()
  for rec in test:
    match = ""
    index = len(rec[PROD])
    product_name = " " + rec[PROD].lower() + " "
    for brand in dict:
      guess_brand = " " + brand.lower().strip() + " "

      try:
        pos = product_name.index(guess_brand)
      except ValueError:
        pos = index + 1

      id = get_id([rec])

      # Rule 1: Leading brand name match is preferred
      if (pos < index) or ((pos == index) and (len(match) < len(brand))):
        index = pos
        match = brand

      # Rule 2: Longest matching brand name is preferred
      if ((pos == index) and (len(match) < len(brand))):
        index = pos
        match = brand

    # Rule 3: Suffix rule: try finding suffixes
    for suffix in SUFFIXES:
      if (" " + match + " " + suffix + " ").lower() in product_name:
        match = match + " " + suffix
        break

    extracted_brand.append(match)


  # calculate precision and recall
  compute_pr(get_id(test), get_truth(test), extracted_brand)
```

Output of our extractor:

tp, brand: BTI, extracted: BTI id: 41183261#TEKENVY
tp, brand: Panasonic, extracted: Panasonic id: 40208059
tp, brand: AEARO, extracted: AEARO id: 40855323
fp, brand: , extracted: Socket id: 40603745#Monoprice Inc
tp, brand: Intel, extracted: Intel id: 38106727
tp, brand: Urban Factory, extracted: Urban Factory id: 41194394
tp, brand: Turtle Beach, extracted: Turtle Beach id: 17298980
tp, brand: EVEREADY, extracted: EVEREADY id: 10242709#Tonzof
tp, brand: Canon, extracted: Canon id: 41509264
tp, brand: Kanex, extracted: Kanex id: 30655254
tp, brand: HP LaserJet Pro, extracted: HP LaserJet Pro id: 42397494
tp, brand: Dell, extracted: Dell id: 41177272
tp, brand: Ricoh, extracted: Ricoh id: 41447001
tp, brand: Dell Optiplex, extracted: Dell Optiplex id: 41053806#US Micro
tp, brand: Peerless, extracted: Peerless id: 11961377
tp, brand: EDGE, extracted: EDGE id: 42398246#TigerDirect
tp, brand: Belkin, extracted: Belkin id: 40986263#TEKENVY
tp, brand: Xerox, extracted: Xerox id: 13056832
tp, brand: TRENDnet, extracted: TRENDnet id: 23596846
tp, brand: Symantec, extracted: Symantec id: 41091471
tp, brand: Middle Atlantic, extracted: Middle Atlantic id: 11465425#Wayfair
tp, brand: Dataproducts, extracted: Dataproducts id: 19311056
tp, brand: DAYTON, extracted: DAYTON id: 43066779#Zoro
tp, brand: Digital, extracted: Digital id: 11043750#Walmart.com
tp, brand: EVERCOOL, extracted: EVERCOOL id: 41034431#TigerDirect
tp, brand: Ricoh, extracted: Ricoh id: 9135600
tp, brand: Corlink, extracted: Corlink id: 42450969#TigerDirect
tp, brand: Iluv, extracted: Iluv id: 21668710#UnbeatableSale.com
tp, brand: Niles, extracted: Niles id: 40623151#OneCall
tp, brand: Dual, extracted: Dual id: 40500640
tp, brand: Cocoon, extracted: Cocoon id: 40871588#TEKENVY
tp, brand: Comprehensive, extracted: Comprehensive id: 41192877#TEKENVY
tp, brand: Dell Optiplex, extracted: Dell Optiplex id: 41314969
tp, brand: Microsoft, extracted: Microsoft id: 39745594#Tech For Less Inc
tp, brand: Jabra, extracted: Jabra id: 11980344#Walmart.com
tp, brand: Tripp Lite, extracted: Tripp Lite id: 13212909#Tech For Less Inc
tp, brand: Gefen, extracted: Gefen id: 41248183
tp, brand: SteelSeries, extracted: SteelSeries id: 32114184
tp, brand: HP ProLiant, extracted: HP ProLiant id: 40804269#TigerDirect
tp, brand: Microsoft, extracted: Microsoft id: 11331573
tp, brand: Corlink, extracted: Corlink id: 42462431
tp, brand: EDGE, extracted: EDGE id: 42394281#TigerDirect
fn, brand: Multi-Tech, extracted: id: 40672898#TigerDirect
fp, brand: Apple iPhone, extracted: Apple id: 33152928#Tech For Less Inc
tp, brand: Upg, extracted: Upg id: 21618720#UnbeatableSale.com
tp, brand: Xerox, extracted: Xerox id: 13056887
tp, brand: Farenheit, extracted: Farenheit id: 41879033
tp, brand: Wiremold, extracted: Wiremold id: 23327073
tp, brand: Incipio, extracted: Incipio id: 42531056
tp, brand: GN, extracted: GN id: 40818772#TEKENVY
tp, brand: Hp, extracted: Hp id: 41439155
tp, brand: HP, extracted: HP id: 42945872#O.co

fn, brand: THINKSERVER, extracted: id: 40564304#TigerDirect
tp, brand: Cambridge Audio, extracted: Cambridge Audio id: 40623036#OneCall
tp, brand: BUFFALO, extracted: BUFFALO id: 42579402#TigerDirect
tp, brand: JILL-E, extracted: JILL-E id: 42462547#TigerDirect
tp, brand: Belkin, extracted: Belkin id: 932042
tp, brand: Innovera, extracted: Innovera id: 14917605#Shoplet
tp, brand: Energizer, extracted: Energizer id: 876218
tp, brand: HP, extracted: HP id: 4365698#UnbeatableSale.com
fn, brand: Riptidz, extracted: id: 17164147#Circuit City
tp, brand: Unibrain, extracted: Unibrain id: 29945338#UnbeatableSale.com
tp, brand: ASUS, extracted: ASUS id: 40593240#TigerDirect
tp, brand: Netgear, extracted: Netgear id: 41258838#TEKENVY
tp, brand: iLive, extracted: iLive id: 16541531
tp, brand: Gear Head, extracted: Gear Head id: 42557704
tp, brand: Comprehensive Cable, extracted: Comprehensive Cable id: 21862748
tp, brand: UPG, extracted: UPG id: 21618688
tp, brand: Lexmark, extracted: Lexmark id: 5799659#Mega Retail Store
tp, brand: Cisco, extracted: Cisco id: 32504280
tp, brand: Corlink, extracted: Corlink id: 42450749#TigerDirect
tp, brand: Corlink, extracted: Corlink id: 42458841
tp, brand: Sangean, extracted: Sangean id: 11039688#Walmart.com
tp, brand: Belkin Mobile, extracted: Belkin Mobile id: 11038058
tp, brand: Kensington, extracted: Kensington id: 14235463#Mega Retail Store
tp, brand: Comprehensive, extracted: Comprehensive id: 11962020
tp, brand: Tripp Lite, extracted: Tripp Lite id: 41193329#TEKENVY
fn, brand: Kingsons, extracted: id: 41285878
tp, brand: STARTECH, extracted: STARTECH id: 13073243#UnbeatableSale.com
tp, brand: Monster Cable, extracted: Monster Cable id: 24278318#Music123
tp, brand: Corlink, extracted: Corlink id: 42462595
tp, brand: INSTEN, extracted: INSTEN id: 43025577#O.co
tp, brand: Biltwell Inc., extracted: Biltwell Inc. id: 37241921#Perf-Moto
tp, brand: Pyle, extracted: Pyle id: 40985592#TEKENVY
tp, brand: Peerless, extracted: Peerless id: 11331690
tp, brand: VCOM, extracted: VCOM id: 32782402#Tech For Less Inc
fn, brand: Griffin, extracted: id: 40984162
tp, brand: Corlink, extracted: Corlink id: 42450699
tp, brand: Energizer, extracted: Energizer id: 872063#Walmart.com
tp, brand: PNY, extracted: PNY id: 26504638
tp, brand: Majesco, extracted: Majesco id: 24412709#Tech For Less Inc
tp, brand: 3M, extracted: 3M id: 41002428#TigerDirect
fn, brand: Okidata, extracted: id: 23140695#UnbeatableSale.com
tp, brand: SIIG, extracted: SIIG id: 41192735
tp, brand: Case Logic, extracted: Case Logic id: 42558134#TEKENVY
tp, brand: Antenna, extracted: Antenna id: 12548336
fp, brand: Intel Xeon, extracted: Intel id: 41437338#TigerDirect
fn, brand: Spin-Clean, extracted: id: 40623265
tp, brand: Monster, extracted: Monster id: 29749088
tp, brand: Sennheiser, extracted: Sennheiser id: 40695348#OneCall
tp, brand: INSTEN, extracted: INSTEN id: 42953012
tp, brand: Boss, extracted: Boss id: 23852095#UnbeatableSale.com
tp, brand: Ironkey, extracted: Ironkey id: 22237843#UnbeatableSale.com
tp, brand: Dell, extracted: Dell id: 41679951#US Micro
tp, brand: StarTech.com, extracted: StarTech.com id: 41493582#TEKENVY
tp, brand: GreatShield, extracted: GreatShield id: 41961943#SF Planet

tp, brand: V7, extracted: V7 id: 40870909
tp, brand: Seidio, extracted: Seidio id: 41981072
tp, brand: Corsair, extracted: Corsair id: 42508283#TigerDirect
fn, brand: Cocoa Touch, extracted: id: 11089046#Walmart.com
tp, brand: APC, extracted: APC id: 40871410
fp, brand: Dell Latitude, extracted: Dell id: 41177187
tp, brand: Idatalink, extracted: Idatalink id: 42517022#HappEshopper
tp, brand: Innovera, extracted: Innovera id: 14922688
tp, brand: Digi, extracted: Digi id: 41195472
tp, brand: LaCie, extracted: LaCie id: 11016993#Walmart.com
tp, brand: C2G, extracted: C2G id: 40984286#TEKENVY
tp, brand: PNY, extracted: PNY id: 40987111
tp, brand: Xerox, extracted: Xerox id: 42509754
tp, brand: Corlink, extracted: Corlink id: 42462509#TigerDirect
tp, brand: Intel, extracted: Intel id: 42814082#TigerDirect
tp, brand: Pelican, extracted: Pelican id: 41493819#TEKENVY
tp, brand: Belkin, extracted: Belkin id: 8223016
tp, brand: Tripp Lite, extracted: Tripp Lite id: 11077951
tp, brand: Dataproducts, extracted: Dataproducts id: 19311008
tp, brand: roocase, extracted: roocase id: 42398267#TigerDirect
fn, brand: Ambir, extracted: id: 17046055
tp, brand: Amped Wireless, extracted: Amped Wireless id: 40999925#TigerDirect
tp, brand: Acer, extracted: Acer id: 40871293#TEKENVY
fp, brand: Kingston ValueRAM, extracted: Kingston id: 42397735#TigerDirect
tp, brand: MSI, extracted: MSI id: 40871056
tp, brand: V7, extracted: V7 id: 41812441
tp, brand: Middle Atlantic, extracted: Middle Atlantic id: 11462848#Wayfair
tp, brand: Creative Concepts, extracted: Creative Concepts id: 21576697#UnbeatableSale.com
fn, brand: Winslow, extracted: id: 20850274
tp, brand: Memorex, extracted: Memorex id: 11013678#Walmart.com
tp, brand: GN, extracted: GN id: 40869983
precision: 0.96062992126 recall: 0.924242424242
true positive: 122 true negative: 13
false positive: 5 false negative: 10