Homework 8

Rebecca Iglesias-Flores, Shubhankar Patankar

April 3, 2020

1 Hearst Patterns

1.1 Regular Expressions

The following five Hearst patterns were implemented based on 'Automatic Acquisition of Hyponyms from Large Text Corpora' (Hearst, 1992):

1. `(NP_[\w\-]+ (, )?such as (NP_[\w\-]+ ? (, )?(and |or )?)+)`
2. `(NP_such_[\w\-]+ as (NP_[\w\-]+ ? (, )?(and |or )?)+)`
3. `((NP_[\w\-]+ ?(, )?)+(and |or )?NP_other_[\w\-]+)`
4. `(NP_[\w\-]+ (, )?including (NP_[\w\-]+ ? (, )?(and |or )?)+)`
5. `(NP_[\w\-]+ (, )?especially (NP_[\w\-]+ ? (, )?(and |or )?)+)`

Five additional patterns were implemented based on instances of hyponym-hypernym relationships observed in the Wikipedia corpus:

6. `(NP_[\w]+ (, )?has been (NP_[\w]+ ? (, )?(and |or )?)+)")`
7. `(NP_[\w]+ (, )?are otherwise known as (NP_[\w]+ ? (, )?(and |or )?)+))`
8. `(NP_[\w]+ (, )?is (NP_a_[\w]+ ? (, )?(and |or )?)+)`
9. `(NP_[\w\-]+ of (NP_[\w\-]+) (, )?includes (NP_[\w\-]+ ? (, )?(and |or )?)+)`
10. `(there is NP_[\w\-]+ (, )?possibly (NP_[\w\-]+ ? (, )?(and |or )?)+)`

1.2 Example Extractions Based on Patterns 6-10

6. – **Text**: 'For many years, the tavern has been a private residence and ironically, the principal mill has been a fine restaurant and bar.'
   – **Extracted Relationship(s)**: [('tavern', 'residence')]
7. – **Text**: 'The shelters are otherwise known as motorcycle sheds, bike sheds and motorbike sheds.'
   – **Extracted Relationship(s)**: [('sheds', 'shelters')]
8. – **Text**: 'Carsluith Castle is a ruinous tower house, dating largely to the 16th century.'
   – **Extracted Relationship(s)**: [('Castle', 'house')]
9. – **Text**: 'Typical wildlife of the North River includes Great Blue Heron, Wood Duck, Canada Goose, Belted Kingfisher, Baltimore Oriole, Painted Turtle, Common Snapping Turtle, Largemouth Bass, Sun Perch, Catfish, Eastern Cottontail Rabbit, White-tailed Deer, Raccoon, Opossum, Brown Bats, Freshwater Clams, Mink, Tiger Swallowtail and Ebony Jewelwing.'
   – **Extracted Relationship(s)**: [('Heron', 'wildlife'), ('Duck', 'wildlife'), ('Goose', 'wildlife'), ('Kingfisher', 'wildlife'), ('Oriole', 'wildlife'), ('Turtle', 'wildlife'), ('Turtle', 'wildlife'), ('Bass', 'wildlife'), ('Perch', 'wildlife'), ('Catfish', 'wildlife'), ('Rabbit', 'wildlife'), ('Deer', 'wildlife'), ('Raccoon', 'wildlife'), ('Opossum', 'wildlife'), ('Bats', 'wildlife'), ('Clams', 'wildlife'), ('Mink', 'wildlife'), ('Swallowtail', 'wildlife'), ('Jewelwing', 'wildlife')]
10. – **Text**: 'Along the lower right side of the shaft, there is a small animal, possibly a cat.'
   – **Extracted Relationship(s)**: [('cat', 'animal')]

1.3 Performance Analysis

1.3.1 BLESS 2011 Training Data Performance

	P	R	F
unlemmatized corpus + [1-5]	0.84	0.23	0.36
lemmatized corpus + [1-5]	0.81	0.29	0.43
unlemmatized corpus + [1-10]	0.75	0.30	0.43
lemmatized corpus + [1-10]	0.81	0.29	0.43

Table 1: Performance analysis of patterns from Hearst 1992 derived from the lemmatized Wikipedia corpus, and subsequently applied to the BLESS 2011 training data set for hypernymy extraction. Rows in the table differ from each other depending on the nature of the Wikipedia corpus (lemmatized vs. unlemmatized) used to derive initial hyponym-hypernym relationships, as well as the set of Hearst patterns ([1-5] vs. [1-10]) applied to the data sets.

Based on Table 1, when Hearst Patterns 1-5 are used, performance improves when the lemmatized Wikipedia corpus ($F = 0.43$) is used to derive hyponym-hypernym relationships relative to when the unlemmatized corpus ($F = 0.36$) is used. No gains beyond these are observed for a larger

Hearst pattern set when comparing the lemmatized and unlemmatized corpora (rows 3 and 4). When the set of Hearst patterns is expanded to include Patterns 6-10, performance improves if the unlemmatized corpus is used ($F = 0.36$ to $F = 0.43$). However, no gains are observed if the lemmatized corpus is used instead (rows 2 and 4).

1.3.2 BLESS 2011 Validation Data Performance

	P	R	F
unlemmatized corpus + [1-5]	0.70	0.25	0.37
lemmatized corpus + [1-5]	0.87	0.46	0.60
unlemmatized corpus + [1-10]	0.66	0.29	0.40
lemmatized corpus + [1-10]	0.87	0.46	0.60

Table 2: Performance analysis of patterns from Hearst 1992 derived from the lemmatized Wikipedia corpus, and subsequently applied to the BLESS 2011 validation data set for hypernymy extraction. Rows in the table differ from each other depending on the nature of the Wikipedia corpus (lemmatized vs. unlemmatized) used to derive initial hyponym-hypernym relationships, as well as the set of Hearst patterns ([1-5] vs. [1-10]) applied to the data sets.

Based on Table 2, when Hearst Patterns 1-5 are used, performance improves when the lemmatized Wikipedia corpus ($F = 0.60$) is used to derive hyponym-hypernym relationships relative to when the unlemmatized corpus ($F = 0.37$) is used. Similarly, when all 10 Hearst patterns are used, performance improves from the unlemmatized Wikipedia corpus ($F = 0.40$) to the lemmatized Wikipedia corpus ($F = 0.60$) as applied to the BLESS validation data. In addition, when the set of Hearst patterns is expanded to include Patterns 6-10, performance improves regardless of whether the unlemmatized (rows 1 and 3) or lemmatized (rows 2 and 4) corpora are used.

1.3.3 Performance Comparison for Patterns 1-5 on BLESS 2011 Validation Data

	P	R	F
lemmatized corpus + 1	1.00	0.36	0.53
lemmatized corpus + 2	1.00	0.07	0.13
lemmatized corpus + 3	0.82	0.32	0.46
lemmatized corpus + 4			
lemmatized corpus + 5	1.00	0.04	0.07

Table 3: Performance analysis of individual patterns from Hearst 1992 derived from the lemmatized Wikipedia corpus, and subsequently applied to the BLESS 2011 validation data set for hypernymy extraction. Note: No patterns matches are found in the validation data corresponding to Hearst Pattern 4.

Based on Table 3, Hearst Pattern 1 is the most useful when extracting hypernymy relationships from the data ($F = 0.53$). This is followed by Hearst Patterns 3 ($F = 0.46$), 2 ($F = 0.13$) and 5 ($F = 0.07$), in that order. No matches corresponding to Hearst Pattern 4 are found in the Wikipedia corpus implying that it is not well-suited to extract hypernymy relationships.

2. Dependency Paths: Information

New_wikipedia_deppaths.txt was used to extract WordPairs relationships and their associated dependency paths. For each WordPair, a Counter() was maintained in order to keep track of the number of forward or reverse relationships that WordPair was identified in. Below is a table of the top twenty most commonly occurring dependency paths with there associated hypernym/hyponym relationship counts throughout the new_wikipedia_deppaths.txt.

Dependency Path	Hypernym/Hyponym Dictionary Counter
1. X/NOUN/conj_<Y/NOUN/conj	Counter({'total': 14962, 'forward': 7865, 'reverse': 7097})
2. X/NOUN/pobj_<Y/NOUN/conj	Counter({'total': 8791, 'forward': 4567, 'reverse': 4224})
3. X/NOUN/dobj_<Y/NOUN/conj	Counter({'total': 4588, 'forward': 2480, 'reverse': 2108})
4. X/NOUN/nmod_<Y/NOUN/conj	Counter({'total': 2621, 'reverse': 1311, 'forward': 1310})
5. X/NOUN/pobj_<of/ADP/prep_<Y/NOUN/pobj	Counter({'total': 2385, 'reverse': 2199, 'forward': 186})
6. the/DET/det<_X/NOUN/pobj_<of/ADP/prep_<Y/NOUN/pobj	Counter({'total': 2108, 'reverse': 2007, 'forward': 101})
7. X/NOUN/appos_<Y/NOUN/conj	Counter({'total': 1894, 'forward': 1027, 'reverse': 867})
8. X/NOUN/conj_<Y/NOUN/conj_>and/CCONJ/cc	Counter({'total': 1410, 'forward': 724, 'reverse': 686})
9. X/PROPN/conj_<Y/PROPN/conj	Counter({'total': 1127, 'forward': 573, 'reverse': 554})
10. the/DET/det<_X/NOUN/pobj_<Y/NOUN/conj	Counter({'total': 958, 'forward': 497, 'reverse': 461})
11. X/NOUN/pobj_<Y/NOUN/conj_>and/CCONJ/cc	Counter({'total': 934, 'forward': 470, 'reverse': 464})
12. X/NOUN/attr_<Y/NOUN/conj	Counter({'total': 781, 'forward': 399, 'reverse': 382})
13. X/PROPN/pobj_<Y/PROPN/conj	Counter({'total': 721, 'forward': 385, 'reverse': 336})
14. X/NOUN/nsubj_<Y/NOUN/conj	Counter({'total': 696, 'forward': 353, 'reverse': 343})
15. X/NOUN/pobj_<of/ADP/prep_<Y/PROPN/pobj	Counter({'total': 660, 'reverse': 651, 'forward': 9})
16. X/NOUN/pobj_<Y/NOUN/appos	Counter({'total': 657, 'reverse': 390, 'forward': 267})
17. the/DET/det<_X/NOUN/dobj_<Y/NOUN/conj	Counter({'total': 653, 'forward': 351, 'reverse': 302})
18. X/PROPN/conj_<Y/NOUN/conj	Counter({'total': 620, 'forward': 346, 'reverse': 274})

19. X/NOUN/dobj_<of/ADP/prep_<Y/NOUN/pobj	Counter({'total': 618, 'reverse': 509, 'forward': 109})
20. the/DET/det<_X/NOUN/pobj_<of/ADP/prep_<Y/PROPN/pobj	Counter({'total': 598, 'reverse': 591, 'forward': 7})

Table 4. List of the "Top Twenty" dependency paths and their associated hypernym/hyponym (forward, reverse) relationships counts.

3. Dependency Paths: PR Analysis

Percentile	**WordPair**	**Dependency Path**
Max Min	wordpair_counts: 4805 Wordpair_counts: 1	sorted_dict_counts: 14962 sorted_dict_counts: 1
10th	('willow', 'trunk')	Ã¢a/ADV/nsubj<_X/NOUN/ccomp_<Y/NOUN/conj
20th	('willow', 'search')	Ã¢a/ADV/nsubj<_X/NOUN/ccomp_<Y/NOUN/conj
30th	('villa', 'incident')	Ã¢a/ADV/nsubj<_X/NOUN/ccomp_<Y/NOUN/conj
40th	('washer', 'oven')	Ã¢a/ADV/nsubj<_X/NOUN/ccomp_<Y/NOUN/conj
50th	('wardrobe', 'dresser')	Ã¢a/ADV/nsubj<_X/NOUN/ccomp_<Y/NOUN/conj
60th	('snake', 'scale')	Ã¢a/ADV/nsubj<_X/NOUN/ccomp_<Y/NOUN/conj
70th	('pig', 'head')	Ã¢833/NOUN/compound<_X/NOUN/pobj_<Y/NOUN/appos
80th	('lettuce', 'cucumber')	Ã¢the/DET/det<_X/NOUN/pobj_<Y/NOUN/conj
90th	('bus', 'driver')	Ã¢the/DET/det<_X/NOUN/pobj_<Y/NOUN/conj
95th	('missile', 'bomb')	wild/ADJ/amod<_X/NOUN/dobj_<Y/NOUN/conj
100th	('hospital', 'school')	X/NOUN/conj_<Y/NOUN/conj

Table 5. List of Percentiles of the "Relevant" WordPairs and "Relevant" Dependency Paths occuring after they were extracted and classified from new_wikipedia_deppaths.txt *Note*: the percentile WordPair and Dependency Path within the same row are not related (not linked), due to the fact that one WordPair could be mapped to several dependency paths in the new_wikipedia_deppaths.txt file.

Below is an image taken from new_wikipedia_deppaths.txt:

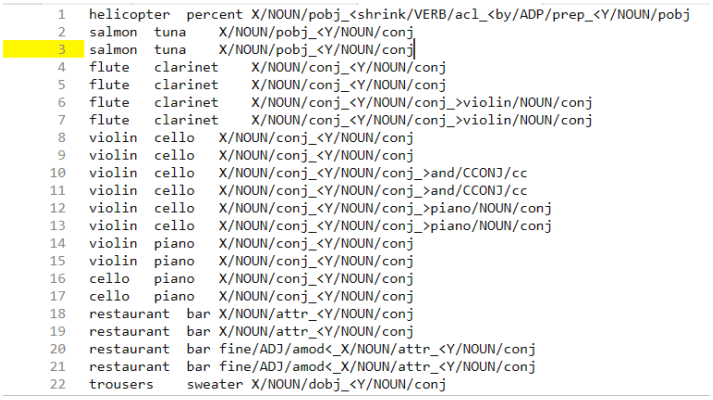

Image 1. Image taken directly from new_wikipedia_deppaths.txt

As you can see in image 1 above, any single WordPair can map to several dependency paths in new_wikipedia_deppaths.txt. In order to determine the "Relevant" path associated specifically to that WordPair, counts were taken of all the forward and reverse relationships that WordPair was seen in and then weighted (see Counter() of hypernym and hyponym relationships for each WordPair in Table 4). In order to single out the relevant dependency path for that WordPair, forward relationships were weighed higher (meaning they were given a lower threshold score of 0.65 in order to be automatically classified as a forward relationship). Several weighing schemes were experimented with between weighing reverse relationship counts higher, etc. but the final values of 0.65 for forward and 0.80 for reverse were used to make the final decision of whether to classify a WordPair as forward or reverse. The final results of those "Relevant Paths" and their associated percentiles can be seen in Table 5 above.

4. DIY: Method

In the same vein as in Snow et. al. (NIPS 2005), we used supervised classification with dependency path features. Our first feature was a concatenated word embedding for the word pair and our second feature was a one-hot-vector of the dependency path that word pair would most likely be identified as:

List of Features	
Concatenate WordPair Embedding	wordpair_embed = np.concatenate((w1_embed, w2_embed))
depPath one-hot-vector	np.zeros(len(unique_relevantPaths))

Table 6. List of features used in supervised classification

Both the new_wikipedia_deppaths.txt and the output from extractRelevantDepPaths.py (in Task 2) were used to generate a one-hot-vector for each WordPair. Each WordPair could be mapped to several dependency paths, so for each WordPair a list of those dependency paths were extracted from new_wikipedia_deppaths.txt and then used to identify which one of those paths was the ***most relevant*** based off the output "relevantDepPaths.txt" that we extracted from extractRelevantDepPaths.py in Task 2. If there was a match in Wikipedia for that WordPair, then the index argument was returned from its corresponding depPath in the unique relevant paths output in order to change the value of that position to 1 in the one-hot-vec for that WordPair.

5. DIY: Analysis

Several models were compared in classifying hypernym relations using only the word pair as a training example and predicting upon those wordpairs.

Models	F1 Score
1. **Perceptron(verbose = 1, max_iter = 5000)**	**47.37 %**
2. LinearSVC(max_iter=10000)	40.1 %
3. PassiveAggressiveClassifier(verbose = 1, loss = 'squared_hinge')	34.03 %
4. RidgeClassifier()	26.06 %
5. LinearSVC(max_iter=2000)	Failed Results
6. LinearSVC(max_iter=5000)	Failed Results

Table 7. List of models that were used to predict WordPair hypernym relationships and their associated F1-Scores on the test set.

One of the more interesting aspects of our study is the LinearSVC model whose iterations had to be increased to 10000 in order to get any results on the test set. The other max iterations timed out before results could be reported, but I'm glad I kept trying, because the Linear SVM model (see models .3- .5 in Table 7) ended up being one of the best performing models for this classification task performing at 40.1% ranking 2nd.

Our best performing model was the Perceptron(verbose=1, max_iter=5000) that beat out the next best model by almost 10%. This was a significant increase and completely divergent from the model I thought would perform the best based off the previous homework (PassiveAggressive).