Authors:
Srinivas suri
Ilayda Onur

Part 1:

Basic Experiments:

1) Hearst Pattern 1 with no lemmatization:

Train F1-score: .09

Validation F1-score: .13

2) Hearst Pattern 1 with lemmatization:

Train F1-score: .17

Validation F1-score: .3

Analysis for 1,2:

It can see be seen that using lemmatization improved the validation performance from .13 to .3 .

Several Things tried out just for fun and to explore the dataset in more detail:

Experiments done to understand the dataset in depth:

3) Pattern 1 only with Lemmatization on hearts.txt but trained on non lemmatized wiki corpus:

Basically, we generated hearst.txt on the non lemmatized wiki corpus but once we generated hearsts.txt .We lemmatized the hypernyms and hyponyms generated.

Train F1-score: .18

Validation F1-score: .21

4) Pattern 1 With only 2nd column lemmatised but trained on non lemmatized wiki corpus:

Basically, we generated hearst.txt on the non lemmatized wiki corpus but once we generated hearsts.txt .We lemmatized only the words in the second column(This was just to explore the dataset in more depth without any specific reason as to why we lemmatized only the second column and not the first column as well)

Train: .23

Validation: .26

Experiments with all the 6 Hearst Patterns:

5) All 6 Hearst patterns on lemmatized corpus with all words of the second column:
Train:
Precision:0.7890625 Recall:0.2885714285714286 F1:0.4225941422594142
Val:
Precision:0.7142857142857143 Recall:0.35714285714285715
F1:0.4761904761904762

6) All 6 Hearst patterns on lemmatized corpus but with the last word of second column:
Train:
Precision:0.8691588785046729 Recall:0.26571428571428574
F1:0.40700218818380746
Val:
Precision:1.0 Recall:0.35714285714285715 F1:0.5263157894736842

7) All 6 Hearst patterns on lemmatized corpus but with the first word of second column:
Train:
Precision:0.8666666666666667 Recall:0.18571428571428572
F1:0.3058823529411765
Val:
Precision:0.8888888888888888 Recall:0.2857142857142857
F1:0.43243243243243246

8) All 6 Hearst patterns on lemmatized corpus on the last two words in the second tab of hearts.txt:
Train:
Precision:0.8015873015873016 Recall:0.2885714285714286 F1:0.4243697478991597
Val:
Precision:0.7142857142857143 Recall:0.35714285714285715
F1:0.4761904761904762

Analysis for 5,6,7,8:
The recalls values seem to be low compared to the precision values. This means that the True True's were predicted as False and since the precision is also high, this means that the dataset has more True 'False' values than True 'True' values. This also means that our model failed to recognise True 'True' values

9) Added additional 6 hearst patterns:

```
("(NP_\w+ (, )?like (NP_\w+ ? (, )?(and |or )?)+)", "1"),
("(NP_\w+ (, )?particularly (NP_\w+ (, )?)+(and |or )?NP_\w+)","2"),
```

("(NP_\w+ (,)?specially (NP_\w+ (,)?)+(and |or)?NP_\w+)","3"),
("(NP_\w+ (,)?similarly (NP_\w+ (,)?)+(and |or)?NP_\w+)","4")

Train:
Precision:0.7744360902255639 Recall:0.29428571428571426
F1:0.42650103519668736

Val:
Precision:0.7692307692307693 Recall:0.35714285714285715
F1:0.48780487804878053

Analysis for 9: Pattern 1 turned out to be the most useful pattern. It also makes sense logically as we are more likely to find the word ‘like’ than words like specially,similarly or particularly. We thought similarly would also be seen , but we found very few instances of the pattern with similarly in it. So, first additional “1” pattern turned out to be useful while the fourth additional pattern turned out to be the worst with the second, third finding patterns;but not as frequently as the pattern 1 or not as less frequently as pattern “4”

Part 2:

Some of the most occurring dependency paths:
-forward:

- **X/PROPN/conj_<Y/PROPN/conj**
- **X/NOUN/conj_<Y/NOUN/conj**
- **X/NOUN/dobj_<Y/NOUN/conj**
- **X/PROPN/nsubj>_castle/PROPN/nsubj>_be/VERB/ROOT_<Y/NOUN/attr**
- **X/NOUN/pobj_<Y/NOUN/conj**
- **X/PROPN/pobj_<Y/NOUN/appos**
- **X/PROPN/nsubj>_hotel/PROPN/nsubj>_be/VERB/ROOT_<Y/NOUN/attr**

-reverse:

- **X/NOUN/pobj_<as/ADP/prep_<Y/NOUN/pobj**
- **X/NOUN/nsubj>_weapon/NOUN/nsubj>_be/VERB/ROOT_<Y/NOUN/attr**
- **X/NOUN/pobj_<Y/NOUN/appos**
- **X/NOUN/pobj_<include/VERB/prep_<Y/NOUN/pobj**

Most common Hearst Patterns are:

1. (NP_\w+ (,)?such as (NP_\w+ ? (,)?(and |or)?)+) - Most common
2. (NP_\w+ (,)?especially (NP_\w+ ? (,)?(and |or)?)+) - Less common than 1 but more common than others

Part 3:

Model 1:

We have combined the hypernym-hyponym predictions generated from both the best hearst pattern model and the deppaths pattern model.

Train:

Precision:0.4874141876430206 Recall:0.6085714285714285 F1:0.5412960609911055

Val:

Precision:0.32075471698113206 Recall:0.6071428571428571
F1:0.41975308641975306

Combining both models gave us an improvement in the test F1-score from **33.24** for best Hearst pattern to **34.62** after combining both the hypernym-hyponym predictions

From closely observing the difference between the test set predictions for hearst vs heart+deppaths combined, we observe that hearst+deppaths is able to identify some words better than just the plain hearst. For example, Hearst patterns failed to recognise alligator as creature however, hearst+deppaths has identified alligator as creature. The same case is observed with dagger, artefact as well. It's interesting that hearst+deppaths combined dagger as an artifact.

Model 2:

In our second model, we have used pre-trained word2ved model.
In model1, we have gathered the hypernym-hyponym words from the deppaths and hearst and predicted True if a (hypernym-hyponym) pair from the test set existed in these files. In this model, we also take into consideration the hypernym-hyponym words from the test set where word2vec compares the similarity of these words and it predicts they are hypernym-hyponyms if the words are above a threshold level. Basically, If either hearst+deppaths or word2vec considers the words as hypernym-hyponym , then we consider the word as hyponym-hypernym pairs. The results for similarity of thresholds ranging from 0 to 0.9 are presented in the table below (If the threshold is 0.5 then all the words in the test set with a similarity are considered as hypernym-hyoponym pairs from a word2ved perspective):

Threshold	Word Count	Dataset	precision	recall	f1-score
>=0	3417	Train	.487	.608	.541
>=0	3417	dev	.320	.320	.41
>=0.1	2196	train	.487	.608	.541
>=0.1	2196	dev	.320	.320	.41
>=0.2	1389	train	.487	.608	.541
>=0.2	1389	dev	.320	.320	.41
>=0.3	1000	train	.487	.608	.541
>=0.3	1000	dev	.320	.608	.41
>=0.4	595	train	.487	.608	541
>=0.4	595	dev	.32	.607	.419
>=0.5	293	train	.48	.608	.541
>=0.5	293	dev	.32	.607	.541
>=0.6	119	train	.48	.608	.541
>=0.6	119	dev	.32	.60	.41
>=0.7	37	train	.46	.608	.52
>=0.7	37	dev	.32	.607	.41
>=0.8	6	train	.48	.60	.54
>=0.8	6	dev	.32	.60	.41
>=0.9	0	train	.48	.60	.54
>=0.9	0	dev	.32	.60	.41

Analysis of the above table:
All the Thresholds yielded similar results. We suspect that the wordvec didn't play a very crucial role in determining the relations or the hearst+deppaths were good enough without adding the word2vec to the model.

Adding wordvec predictions to model 1 gave us an improvement in the test F1-score from **34.62** for best Hearst pattern to **34.85**. This is our final submission model where we have leveraged both word2vec and hearst+deppaths models.