



CS 6390

Relation Extraction Project

 Detecting Drug-Drug Interactions 

By Jacob Johnson

Task: Drug-Drug Interaction Extraction

- Taking multiple medications at the same time can lead to unexpected side effects
- *Pharmacovigilance* is the task of monitoring for and protecting against accidentally causing these effects
- If an NLP system can detect and extract Drug-Drug Interactions in the many medical papers that are constantly being published, it could help prescribers, patients, and researchers

Detection and Labeling

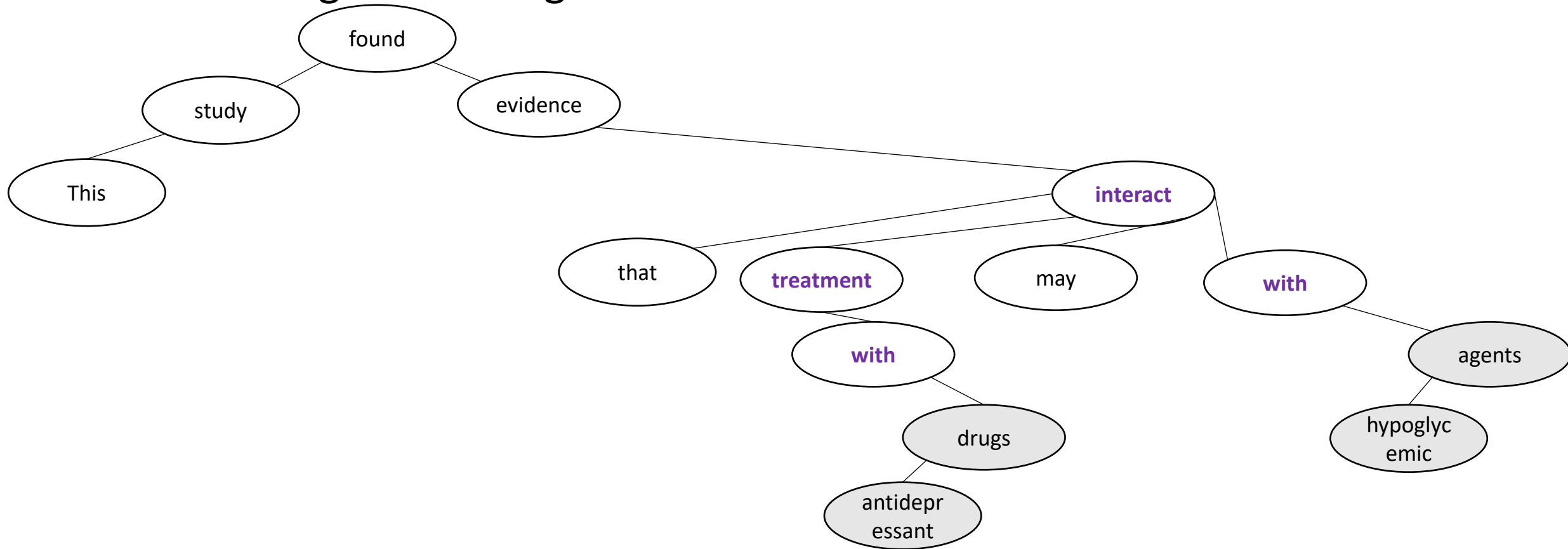
- The Corpus defines four types of drug-drug interactions:
- “Mechanism”: how (biologically) do they interact
- “Effect”: what does this do to the patient
- “Advise”: encouraging not to prescribe both drugs
- “Int”: They just interact

Overall Approach

- Create some vector representation for a pair of extracted entities
 - Updated between phase 2 to 3
- Detect and label
 - Updated between phase 1 to 2

Phase 1 Representation

- Used GloVe vectors for words along the dependency path, aggregated with a weighted average



Vector Aggregation Experiments

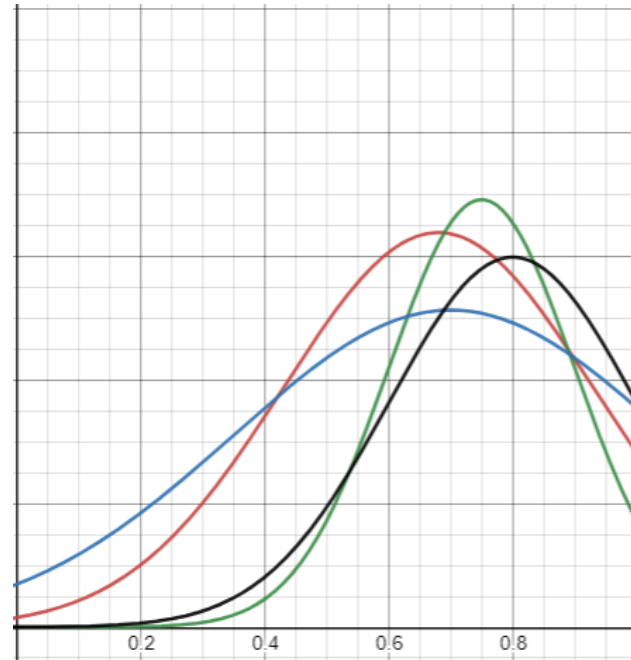
- Vectors of words along the dependency path were summed according to one of the following aggregation schemes
- “Peak” – the highest element in the dependency parse tree has the highest weight
- “End” – the lowest elements in the dependency tree have the highest weight
- “Uniform” – the elements were weighted equally

Phase 1 Decision Making

- Assign a test vector the label of the category whose gold training vectors' average similarity to the test vector is highest, is over some threshold, and whose similarity distribution to the test vector is statistically dissimilar to that of the other categories



Too low – the highest mean similarity isn't high enough



Too close – the distribution from relation with maximal average similarity blends into the other distributions



Just right – the label whose gold examples produced the black distribution wins

Phase 1 Results

Mechanism			Effect			Advise			Int		
P	R	F	P	R	F	P	R	F	P	R	F
.349	.374	.361	.471	.169	.249	.178	.473	.259	.129	.403	.195

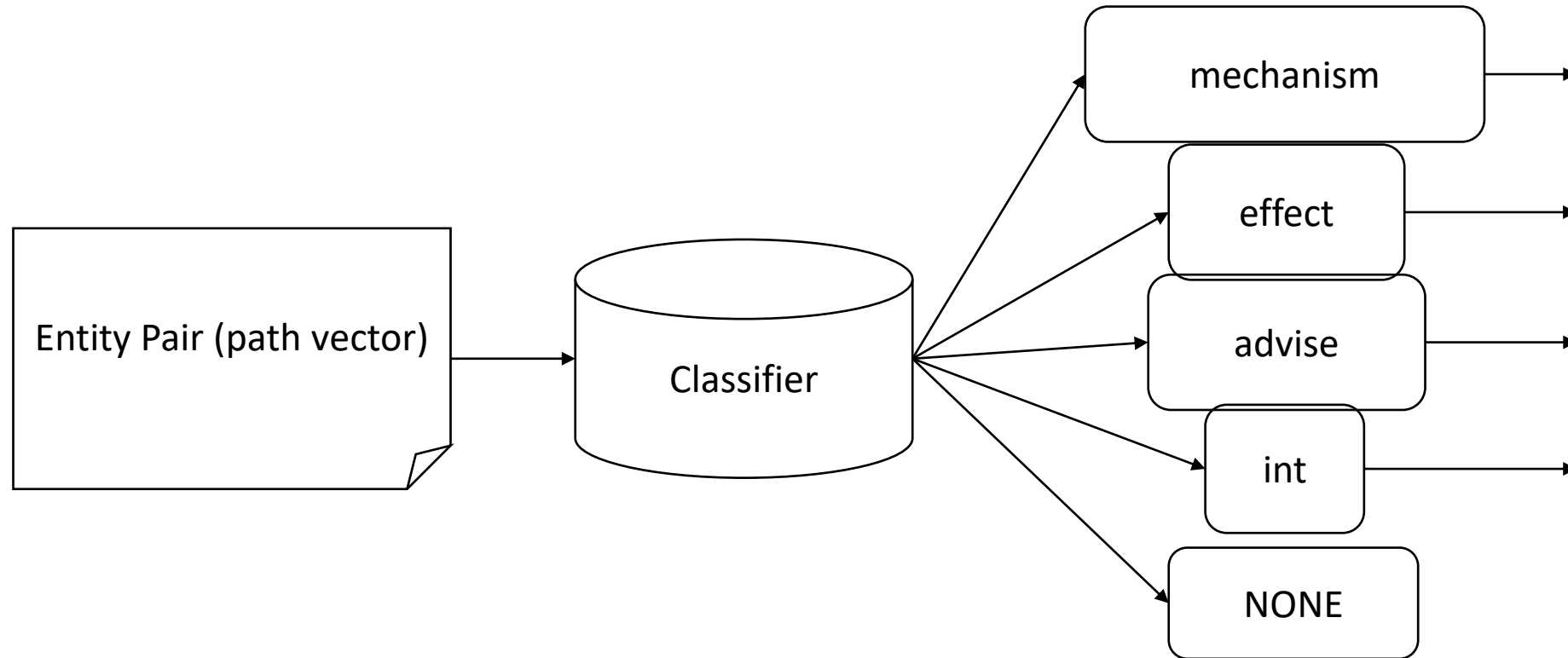
Total		
Precision	Recall	F-Score
.251	.313	.279

Best results were obtained by using a cutoff of .65 : the mean cosine similarity must be at least .65 to be accepted
No major difference was observed between aggregation schemes, but I kept the Peak scheme because it seemed to me like the most intuitive (possibly no difference was observed because the decision making mechanism was so blunt)

Phase 2 Decision Making Upgrade

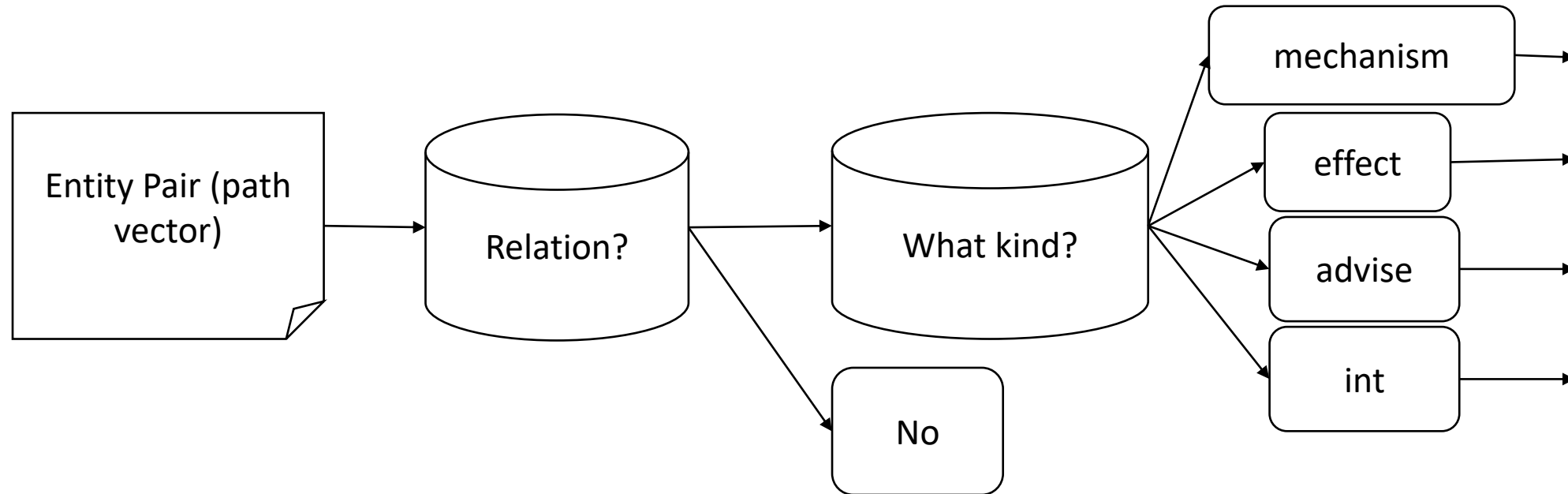
- The statistical method proposed in Phase 1 is very crude, let's use an actual classifier instead
- Trained an sklearn LogisticRegression on the same context vectors
- Experimented with several layouts, including a single multiclassifier, and a pipeline to first detect interaction, then apply a label

Multiclassifier – decide between labels and unrelated all at once



This layout had the highest evaluation, and it is the simplest to train

Pipeline – first decide whether the drugs are interacting, then label the interaction if needed



According to some evaluations that I did after turning in Phase 2, detecting a relation is harder than labeling them. My phase 2 model that had this structure scored .493, .770, and .601 Precision, Recall and F-score on interaction detection and .768, .578, and .660 on labeling. If I'd had the foresight to better consider how this system could actually be used, I probably would have liked this system better for how it's more tunable, and I can decide to ignore the labels altogether if its use case demanded

Phase 2 Results

Mechanism			Effect			Advise			Int		
P	R	F	P	R	F	P	R	F	P	R	F
.405	.671	.505	.395	.657	.494	.272	.545	.362	.296	.368	.328

Total		
Precision	Recall	F-Score
.363	.620	.457

Best results were obtained using the single multiclassifier setup
Phase 2 almost universally improved over the evaluation of Phase 1

Dependency Label “Embeddings” (Phase 3)

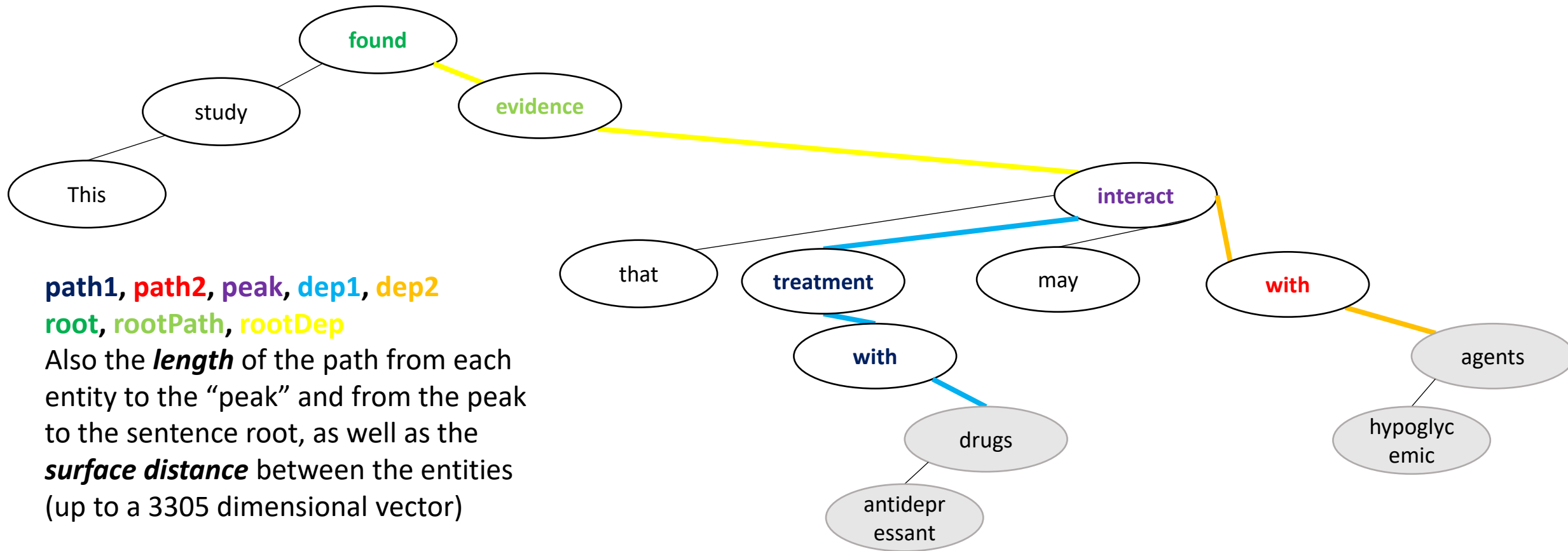
$$\begin{bmatrix} 1 \\ 2 \\ \vdots \\ 299 \\ 300 \\ 301 \\ 302 \\ \vdots \\ 599 \\ 600 \end{bmatrix}
 \begin{bmatrix}
 [1 & \color{red}{2} & \cdots & 44 & 45] \\
 w_{1,1} & \color{red}{w_{2,1}} & \cdots & w_{44,1} & w_{45,1} \\
 w_{1,2} & \color{red}{w_{2,2}} & \cdots & w_{44,2} & w_{45,2} \\
 \vdots & & & & \\
 \vdots & & & & \\
 \vdots & & & & \\
 \vdots & & & & \\
 \vdots & & & & \\
 \vdots & & & & \\
 w_{1,599} & \color{red}{w_{2,599}} & \cdots & w_{44,599} & w_{45,599} \\
 w_{1,600} & \color{red}{w_{2,600}} & \cdots & w_{44,600} & w_{45,600}
 \end{bmatrix}$$

An Embedding-Like representation for dependency labels was generated by training an sklearn LogisticRegression to predict the dependency label given the concatenation of a word and its head. The dependency label was then associated with the corresponding 600-length column of weights.

The highlighted column, for example, is the weight for the second of the 45 dependency labels

Phase 3 Representation Upgrade

- Many additional features



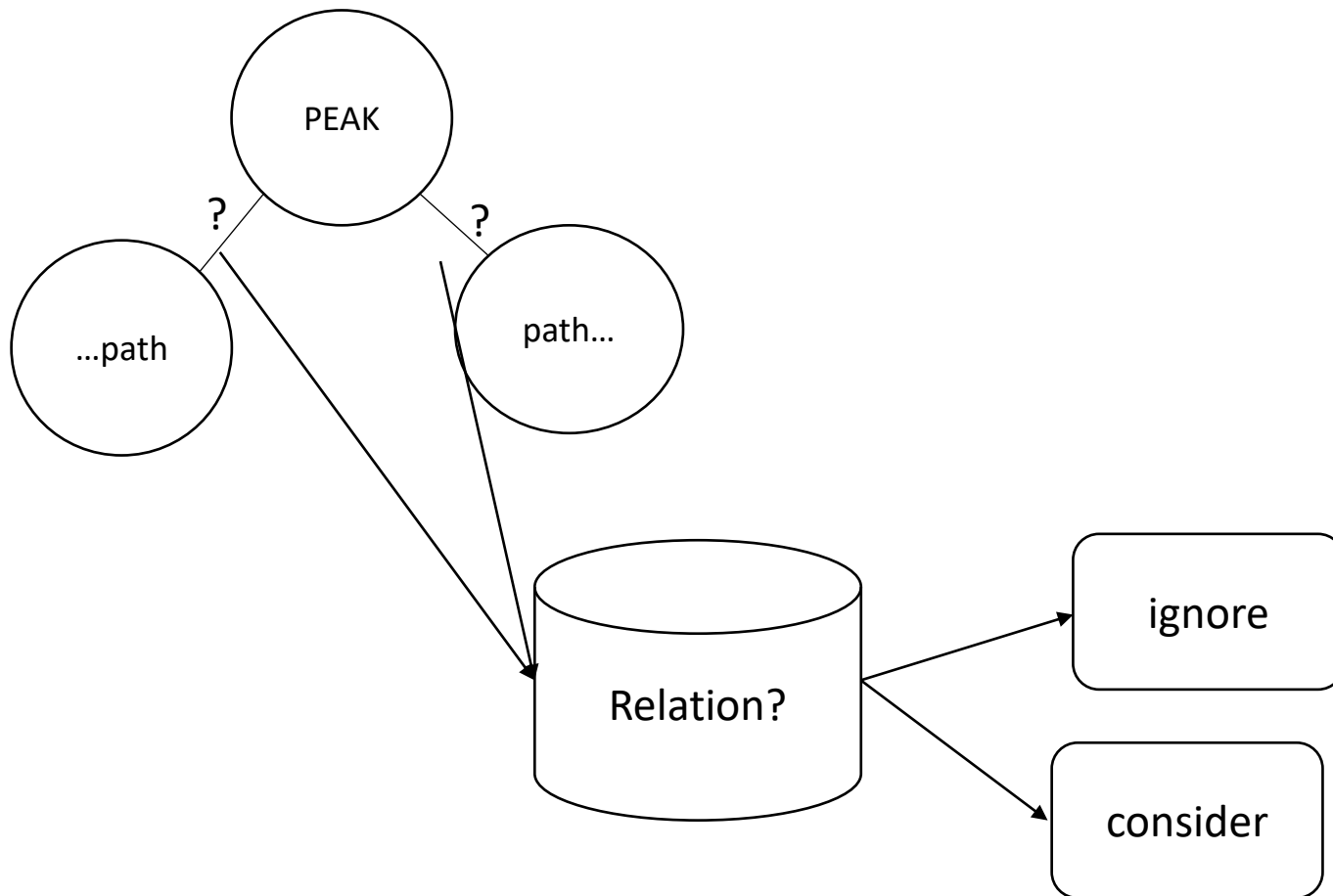
Clustering Contexts (Phase 3)

“ABC ... GHI ... DEF ... GHI. DEF ... GHI. GHI ... DEF.”

ABC-DEF [1,2, ..., 3304,3305]	ABC-GHI [1,2, ..., 3304,3305], [1,2, ..., 3304,3305]	DEF-GHI [1,2, ..., 3304,3305], [1,2, ..., 3304,3305], [1,2, ..., 3304,3305], [1,2, ..., 3304,3305]
----------------------------------	--	--

Instead of considering each mention pair individually, the mention pairs cooccurring within a sentence were collected across a document, and then considered all at once.

Cross-Phrase Relation Rejection Filter (Phase 3)



I trained a classifier on the dependency labels that connect the rest of each path to the “peak”. This filtered out on average 18% of entity mentions, but only 3% of related entity mentions, so it’s a cheap way to stop a mention before it gets completely processed by the main classifier

Phase 3 Results

Mechanism			Effect			Advise			Int		
P	R	F	P	R	F	P	R	F	P	R	F
.504	.442	.471	.528	.595	.560	.495	.512	.503	.523	.431	.472

Total		
Precision	Recall	F-Score
.514	.520	.517

The most important features, according to an exhaustive ablation on the devset, were path2, peak, root, dep1, path1, dep2. The actually best-evaluated system used path1len, path1, dep1, peak, path2, path2len, root, rootPath, and rootDep

Universally improved upon Phase 2 in precision, generally had a lower recall, but also generally had a higher F-score, as well as the best and most rounded overall evaluation

Demo 1 – The Example Sentence

```
[Phase3]$ python3 demo.py This study found evidence  
that treatment with antidepressant drugs may  
interact with hypoglycemic agents
```

```
antidepressant drugs, hypoglycemic agents, effect
```

```
[Phase3]$
```

(correct relation extracted, but incorrect label
applied – should be int)

Demo 2 – A Corpus Document

```
[demo]$ cat Melphalan.txt
```

DRUG INTERACTIONS

There are no known drug/drug interactions with oral ALKERAN
Vaccinations with live organism vaccines are not recommended
in immunocompromised individuals

Nalidixic acid together with high-dose intravenous melphalan
has caused deaths in children due to haemorrhagic
enterocolitis.

Impaired renal function has been described in bone marrow
transplant patients who were conditioned with high-dose
intravenous melphalan and who subsequently received
cyclosporin to prevent graft-versus-host disease

```
[demo]$
```

Demo 2 - Output

```
[Phase3]$ python3 demo.py ../demo/Melphalan.txt  
melphalan, cyclosporin, effect  
[Phase3]$
```

(completely extracted one relation, but missed
another one: Nalixidic acid, melphalan, effect)

Analysis

- My system still fails to capture many cues to drug-drug interaction:
 - “ABC and DEF were tested...”
- It also is running into problems with the parse quality, it would be interesting to see if/how using a parser that was trained on more technical data (instead of web data) would affect the evaluation
- A real system used in this task should probably optimize for Precision, rather than F-score

Lessons Learned

- A lot about evaluating ML systems, including ablation testing, and especially about how Cross-Validation can help expose a “lucky” test set
- Sometimes a small addition can make a big difference (the cross-phrase filter) or a big thing can make a small difference (having 600-dimensional dependency label vectors)
- It is difficult to represent Syntax and Semantics in a way that meaningfully captures a sentence!

Resources Used

- sklearn (scikit-learn.org): the LogisticRegression classifier was the backbone of my systems
- spaCy (spaCy.io): used the en_core_web_lg for dependency parsing and to get GloVe vectors (and also to extract drug entity mentions when gold mentions weren't given)
- numPy (numpy.org) and pandas (pandas.pydata.org): used the ndarray and DataFrame classes to hold vectors
- pickle, xml, re, and statistics packages from python standard library
- (All available on CADE machines or through pip)