Information Extraction Phase 2 Write-Up

Jacob Johnson

1 Information Extraction Task:

My system performs a relation extraction task to recognize Drug-Drug interactions in literature from the biomedical domain of Pharmacovigilance. (Herrera-Zazo et al) My system takes as input raw documents (eithin within the command line or from a file) and optionally, the set of gold entities within the document (If gold entities are not provided, then my system uses a trained spaCy Entity Extraction model). My system then outputs the set of extracted relation tuples, along with which type of relation it is labeled as and which sentence within the document the relation was extracted from. In accordance with the corpus labeling schemes of Herrera-Zazo et al, there are four labels for relations: "mechanism", which describes how drugs interact at a biological level, "effect", which describes what the interaction does to the patient, "advise", which is just a written encouragement not to allow the drugs the chance to interact, or "int", short for interaction, which simply describes that an interaction is observed between the drugs, without further detail.

2 Resources List:

All external python packages were already available on the CADE lab machines and can be installed using Python's pip package installer, but the urls to the documentation websites are included here.

1. spaCy (`https://spacy.io`): en_core_web_lg (spaCy's large web-trained English language model) was used for dependency parsing, lemmatizing, and vectorization. These vectors, in turn, came from GloVe Common Crawl. I also trained its NER capabilities to recognize Drug Entities in the case where gold Entities are not input with the document, where it achieved an F-score of .797.

2. numPy (`https://numpy.org`): the ndarray class was used to hold vectors produced by spaCy.

3. pandas (`https://pandas.pydata.org/`): the DataFrame class was also used to hold vectors produced by spaCy and given to sklearn models as training and input data.

4. sklearn (`https://scikit-learn.org`): the LogisticRegression class was used as the backbone of the machine learning for my system. Varying arrangements of classifiers (see the Technical Description section for details) were trained on the 300-dimensional dependency path vector aggregations produced from spaCy, and these vectors were input as test data.

5. From the python standard library, the pickle package was used to save and load machine learning models, pre-calculated vectors, and forms of the training, dev, and test data that were preloaded from their XML format, the xml package was used to parse the corpus annotations, and the re package was used for regular expressions.

6. The corpus was no longer available at the link provided in Herrera-Zazo et al, but it can be downloaded from `https://github.com/isegura/DDICorpus`

3 Technical Description:

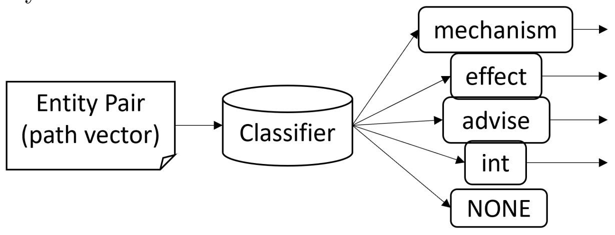
My system relies on a vector-semantics evaluation of the syntactic context (dependency path) that connects pairs of entities within a sentence. Lemmas of words along the dependency path were recorded, vectorized, weighted, and summed. After consulting with Professor Riloff, I determined to wait until phase 3 to include information pertaining to the labels of the dependencies, because my plan for phase 3 is to include several more features. See section 6 for more details.

Vectors were aggregated according to my Peak-Weight aggregation scheme: The vector of the highest element in the dependency tree (the "common syntactic ancestor" of the two entities) was given full weight, but weights were attenuated for path elements lower in the dependency tree, as a fraction of the number of elements on that side of the path. The intuition here was that possibly the word that connected the entities together syntactically would have the most important bearing on whether a syntactic context was indicative of a relation. See phase 1 for an example.

Three layouts composed of Logistic Regression Classifiers were trained for this task:

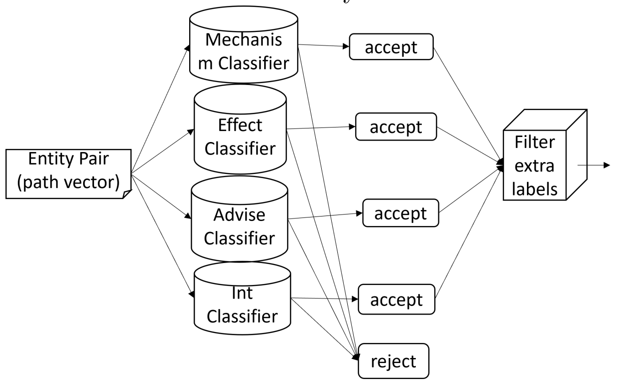
1. Multiclassifer:

   A single classifier was trained on all positive instances of related entities, plus varying amounts of negative (unrelated) examples. The classifier would output a category label or "none". In accordance with the findings of experiments on the devset, this classifier was trained with all positive examples, plus as many negative examples as it had positive examples. Conceptually, this is the baseline classifier layout.



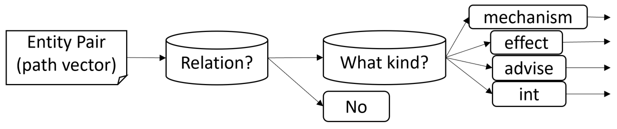
2. Multibinary:

   Four classifiers were trained, one for each relation label. Each classifier was trained on all positive instances of that relation type, plus all instances of other relation types and varying amounts of unrelated entity pairs as negative examples. Each classifier would output "true" or "false". If multiple classifiers accepted an input, all suggested labels would be ignored (similar to the collision resolution strategy from phase 1). An alternate option was explored, involving an additional classifer to arbitrate between proposed labels, but experiments on the devset suggested that this would not improve F-score. Those experiments also suggested that the "advise" classifier should be trained with all positive examples, plus 1.75 times as many unrelated examples, plus all the examples from other categories as additional negative examples, while the other three classifiers were trained with 2 times as many unrelated examples instead of 1.75. The intuition behind this classifier layout was that there might be less noise if the classifiers only have one class of interactions to be concerned with.



3. Pipeline:

A first classifier was trained to identify pairs of entity mentions as related or not related, and a second classifier was trained to label pairs of related entity mentions as an instance of a particular class of interaction. Experiments on the devset suggested that the initial classifier should be trained on as many unrelated examples as related examples. Both classifiers were trained on all positive examples of related entity mentions. The intuition behind this experimentation was that there probably is a lot of similarity across different classes of interactions, so training a classifier that needs to distinguish unrelated entity mentions from different classes of related entity mentions could be overwhelmed by noise; specializing one classifier to filter out unrelated entity mentions and another to distinguish between different labels could decrease the noisiness of the data at each step.



The label suggested by the classifier setup was reported for all pairs of detected or gold entities within each sentence.

4 Phase 2 vs Phase 1

Several additional but minor adjustments were made to the system:

1. Additional options were added to the demo and evaluate scripts to make them easier to work with. For evaluate.py, -test now automatically uses test data, -goldIgnore forces the system to use spaCy-predicted entities, and -mbin, -mclass, or -pipe can be used to select the classifier layout to be used. In demo.py, the same options to select the classifier layout are also available.

2. It was observed that many sentences in the corpus mention the same entity multiple times, which led the system to predict relations between an entity and itself, or multiple different relations between a pair of entities. In this version of the system, I added a simple filter to ignore any candidate relations between an entity and itself (as determined by case-insensitive string matching only), which will slightly improve recall. I started on, but have not implemented, additional improvements which aim to better handle multiple mentions of the same entity, see section 6 for more details.

5 Evaluation:

For the evaluation, Peak-Weighted vector aggregations were used. Precision, Recall, and F-score were calculated on a document-level basis. The scores for the best parameter set of each layout (as suggested by experiments with the devset, see subsections in section 3 for specifics), plus the best results of the phase 1 system are reported with input gold entities (g) and without, i.e., using spaCy's NER (s):

		mechanism			effect			advise			int			total		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
p1	s:	.235	.352	.282	.297	.217!	.251	.212	*.564*	.308	.236	.481	.316	.237	.361	.286
	g:	.255	.359	.298	.345	.212	.262	.218	**.564**	.315	.257	.519	.344	.255	.365	.301
mclass	s:	*.503*	.641	*.563*	.433	.603!	*.504*	.422!	*.491	*.454*	*.400*!!	*.423	.431!!	*.452*	.570!	*.504*
	g:	**.535**	.641	**.583**	.487	.587	**.532**	**.414**	*.500	**.453**	.329	*.442	.377	**.466**	.568	**.512**
mbin	s:	.458	.500!	.478	*.443*	.429	.435	*.440*!	*.464!	.451!	*.400*!!	*.500*	*.444*!	.441	.465!	.453
	g:	.479	.486	.483	**.500**	.429	.462	.412	*.445	.428	**.354**	**.538**	**.427**	.450	.460	.455
pipe	s:	.395	*.704*	.506	.362	*.656*	.466	.292	*.536!	.378	*.222!!	*.154	*.182!	.349	*.590*	.439
	g:	.432	**.711**	.537	.411	**.661**	.507	.299	*.527	.382	*.176	*.173	*.175	.374	**.594**	.459

There is a lot to unpack here, so in addition to bolding and italicizing, the graph also contains explanatory colors and symbols. Red/bold indicates the best entry in a column (with gold entities), while blue/italics indicates the best entry in a column (without gold entities). Generally these two were aligned.

An asterisk denotes that a score is lower than the corresponding score in phase 1. This only occurs in the "advise" and "int" classes, in which the phase 1 system was noted for outperforming its scores on the other classes, presumably on account of their relative infrequency.

An exclamation point indicates where the system appears to have achieved a higher score without gold entities than with them. Some of these (single exclamation points) have a very slight difference, which could potentially have just come about by chance, but there is a pattern around where the largest differences are observed (double exclamation point) – they are all in the "int" category. Recall: the way the classifiers are in their various layouts, the probabilities of an item being given a particular label are linked to their probabilities of being given any of the other available labels. Potentially, because the "int" category is the least populated, there is less cohesion among training examples, thus the classifiers were likely to mislabel entity mention pairs that were less cohesively identifiable as a member of a particular class of relations as "int" interaction instances, so having the full gold set of entity labels, including entities that were obscure enough for the spaCy Entity Extractor system to not recognize just meant more "int"s to mislabel, thus reducing the Precision and F-score. On the whole, including gold entities did help F-score, although once again to a surprisingly minimal degree. Potentially this is a random artifact of the particular train/dev/test split used on my data, so I intend to use cross validation for phase 3 to help against that. See section 6 for more details.

Taking all that into account, which of the three layouts worked best? "multiclassifier" had the best Precision and F-score, while "pipeline" had the best overall recall. "multibinary" appeared to be the most balanced across categories, which makes sense, considering it was designed to take each category into account separately. "pipeline" performed very well with the two common classes, but failed to outperform even the very naive statistical phase 1 model for the least common class. Furthermore, during experiments on the devset, all three of these layouts were roughly equal, so it's possible that the differences observable between these evaluations simply stem from the makeup of the test set. On the whole, the basic "multiclassifier" layout is probably the best, because, in addition to having achieved the highest score here, it is also the simplest to set up, needing to train only a single classifier.

6 Future Work (Plans for Phase 3):

1. As mentioned above, my phase 3 system will include more features beyond just word vectors. As suggested in my phase 1 feedback, I will include a representation of the dependency labels, in addition to the dependency path itself. I also intend to include at least features relating to entity order and the length of the dependency path.
2. Because I have been reporting extracted relations on a document-level basis, and because entities can have multiple mentions across the span of a complete document, it has been the case that multiple, different labels could be applied to the same pair of entities, with each label based on a different pair of mentions. Other systems have seen improvements by considering all mentions pairs of the same entities together, so I intend to include this in my phase 3 submission by having my model take all pairs of two entities into consideration at the same time.
3. To control for unusual results, I will make use of cross validation.

7 References:

Herrero-Zazo, M., Segura-Bedmar, I., Martínez, P., & Declerck, T. (2013). The DDI corpus: An annotated corpus with pharmacological substances and drug–drug interactions. Journal of biomedical informatics, 46(5), 914-920.