Finding PICO: Extracting participant information from clinical trial abstracts

Tiffany Wong tcwong@mit.edu
Hansa Srinivasan hansa@mit.edu
Jing Wen jingw01@mit.edu
Nat Sothanaphan nsothana@mit.edu

Abstract

Evidence Based Medicine relies on systematic reviews, comprehensive synthesis of all published evidence that addresses precise clinical questions. An aspect of systematic reviews involve defining a Population/Problem, Intervention, Comparator and Outcome (a PICO criteria) for each relevant report. While this is a very crucial step, it is also extremely time consuming. Our vision was to automate this step by develop machine learning models that would be able to extract PICO annotations from published abstracts. In this paper, we discuss the steps we took to build our machine learning models including: (1) preprocessing data using majority voting, word2vec and other methods (2) selecting n-gram, CRF and CNN models (3) modifying the given models inputs, outputs, loss and evaluation functions. We also present our results for extracting population annotations and propose further works to extract the remaining PICO annotations.

1. Introduction

Evidence Based Medicine (EBM) optimizes decision making by using evidence from published papers. Comprehensive synthesis of all published evidence that addresses precise clinical questions is the backbone of EBM. Many of those that perform systematic reviews following the PICO framework by annotating Population/Problem, Intervention, Comparator and Outcome for each relevant report. While this is a very cruicial step, it is also extremely time consuming. A main limitation of EBM is the time lag between when research is published and when the intervention is properly applied. [18]

In the past, Conditional Random Field classifiers have been used to identify entities e.g. including treatment groups and outcome, and produced working results.[11] While CRFs achieve the current state-of-the-art performance in this area, neural nets have been revolutionary in many other machine learning problems. Neural nets have begun to outperform other machine learning techniques in various areas of natural language processing, such as some sentence classification problems [5].

In this paper, we explore the use of a Convolutional Neural Network (CNNs) model to identify population related phrases and sentences. We choose to work with CNNs because they have greater flexibility and generality in modeling data than CRFs, and have the potential to outperform CRFs. CNNs have the additional benefit of fewer parameters than conventional feed-forward, fully connected neural nets, making them easier to train and less prone to overfitting. Given the limited data we have, we aim to investigate the performance of CRFs vs CNNs in the task of extracting participant phrases from clinical abstracts.

2. Related work

Similar work has been done to extract information from published abstracts and have obtained desirable results in the biomedical text understanding field. We summarize them in this section. In addition, we review popular information retrieval techniques, such as Conditional Random Field (CRF) and Convolution Neural Network (CNN).

2.1. PICO extraction

A lot of work has been done in the biomedical text mining space and built the foundation for PICO extraction. Pyyalo et al. [14] applied word2vec on PubMed texts to compute vector representations that tailored to biomedical texts, and made the vector representations available to public.

In addition, Genia Tagger, implemented in Tsuruoka and Tsujii [19], is trained with GENIA corpus, a corpus consisting of over 2000 MEDLINE abstracts. It provides part-of-speech tagging and named entity recognition specifically for biomedical texts.

Even with these resources, automating identification of PICO is not a trival task because each component of PICO has different characteristics and requires extra effort. Recent research has focused on improving the performance of specific components, such as calculating absolute risk reduction. Summerscales et al. [11] designs a system that first identifies treatment groups and outcomes, then extracts integers from identified phrases, and lastly calculates summary statistics for each trials. They employ standard methods to first identify candidate sentences, such as shallow semantic parsing used by Paek et al. [4]. Then, they apply a Conditional Random Field classifier to identify entities, i.e. treatment groups and outcomes. Based on their previous work [12], they select useful features, such as word itself, its part of speech tag, features generated from its neighboring words, for the classifier. Lastly, they extract integers from entities and compute the absolute risk reduction.

Wallace et al. [2] focuses on improving the identification of the arms, or groups, in trials by incorporating coreference features. Coreference is particularly useful here because arms are usually mentioned and referred. Corefence chains helps algorithms to learn from patterns and possibly identify arms better. Through an experimentation, new models with coreference features boost F1 score by 9.3%.

2.2. CRF for information extraction

Structured prediction is very useful for natural language process tasks because we often want to make predictions not only based on the word itself but also neigboring predictions. Conditional Random Field (CRF), a probabilistic method for structured prediction, has been popular within the NLP community for this reason. Linear-chain CRF is considered a conditional version of Hidden Markov Model, just as Logistic Regression a conditional version of Naive Bayes. Instead of trying to model $P(\mathbf{y}, \mathbf{x}) = P(\mathbf{x}|\mathbf{y})P(\mathbf{y})$, CRFs model directly the conditional distribution, i.e. $P(\mathbf{y}|\mathbf{x})$ [17]. Hence CRFs are able to model the sequence of predicted variables without modeling the sequence of hidden variables and thus allow a lot of flexibilities in the models. Since CRFs can also perform multi-label classification, the application of Linear-chain CRFs in natural language processing is quite wide, from shallow parsing [15] to named-entity recognition [9]. Skip-chain CRF [16], a more complicated version of CRF, can model long-distance dependencies and be suitable for information extraction.

CRF has also been used frequently in the biomedical text mining. In addition Summerscales et al. [11] discussed above, both Chowdhury and Lavelli [1] and Leanman and Gonzalezto [7] developed NER systems based on CRF and achieved desirable results. Since our task of identifying PICO components are very similar to an NER task, we use CRF as our base model.

2.3. CNNs and deep learning for information extraction

Deep learning has been revolutionary in the areas of computer vision, speech recognition, data prediction, machine translation and for a variety of other machine learning problems. As demonstrated by Yao *et al.*, deep neural nets provide promising results for biomedical named entity recognition, approaching state of the art performances with minimal feature engineering compared to traditional methods [6]. Recurrent neural networks (RNNs), and specifically long short term memory networks (LSTMs), have become commonly used to model sequences and to perform sequence to sequence translation and sequence prediction. Lample *et al.* demonstrate that while deep neural nets have not beat state of the art performances, which are typically achieved using CRFs, the neural nets' performances do not fall far short and have the advantage of not needing feature engineering [3].

Convolutional neural networks (CNNs) have also had success in modeling sentences and prediction problems. Using CNNs, Kalchbrenner *et al.* successfully model sentences [10], Kim improves upon state-of-the-art performance on several sentence classification tasks [5], and Francis-Landau *et al.* achieve state-of-the-art performance on multiple entity linking datasets by capturing semantic similarity [8]. These successes lend credence to the notion that CNNs and neural nets could be utilized to perform the prediction of participant phrases in abstracts.

3. Technical approach

This section describes the data, the prepossessing pipeline and the implementations of the MaxEnt, CRF and CNN models.

3.1. Data

Our data is a collection of 5000 medical abstracts from Professor Byron Wallace. This corpus is newer is larger than others previously used.

The data was tagged by Amazon Mechanical Turk workers. The plan was to tag Participants (P), Interventions (I), and Outcomes (O). However, the Outcomes results are not finished as of now. Moreover, the Interventions prediction seems to require separate optimizations and approaches from Participants. Due to time constraints we focus on extracting Participant phrases in this paper. The tagging was crowdsourced, with roughly 5-10 people tagging each abstract.

3.2. Prepossessing

In order to combine the data into the *gold annotations*, we employed the following scheme.

First, we took the union of every worker's annotations for the abstract. That is, we tagged a token as `P` (for Participants) if and only if at least one worker tagged it as `P`.

Second, we separated the union annotations into *intervals* (contiguous chunks of `P` tags), and chose only those intervals that are made up of at least 3 workers' intervals to be in the gold annotations. The rationale is that each of

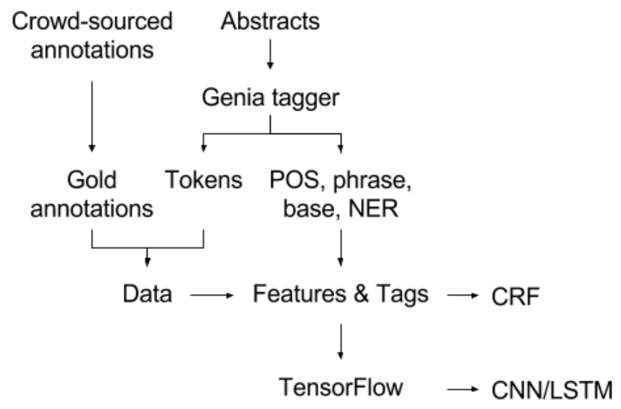

Figure 1. Diagram of the data prepossessing system. Shows the flow of data to features, and to classifiers.

our interval identifies a certain "phrase" that could potentially be a Participants phrase, and a consensus from at least 3 workers gives evidence that it is so.

However, this naive approach has problems, as the quality of the workers' annotations are not very good. Specifically, some workers tagged too many tokens as P by including other parts of the sentences as Participants, resulting in gold annotations which usually were too long. To remedy this, we trimmed the intervals down as follows. For each interval I which overlaps with annotations from exactly k workers, we took only tokens in the interval I which at least $k/2$ workers tagged as P to be in the final gold annotations. This scheme proves adequate for reasonable gold annotations for our 5000 abstracts.

Finally, we considered tokenizing. We first tokenized the abstracts by whitespaces. However, we found that we needed extra information such as POS tags and named entity tags, and we decided on using GENIA tagger, [1] which was trained on biomedical texts, to give us the information. However, GENIA tagger tokenizes the text differently from simply using whitespace, sometimes considering punctuations and parentheses as separate tokens. Therefore, we run GENIA tagger first to tokenize the abstracts and to give us the extra information. Then, we could run our aggregation scheme above to generate the gold annotations for our final dataset.

Our system can thus be summarized by the flowchart in Figure 1.

It is also worth noting that our task is slightly different from that presented in [11]. In that paper, the task was to identify PICO elements in only the sentences which contain numbers, in order to summarize the outcome results, while our task is to identify Participants in the whole abstracts. We will discuss this later in the Results section.

[1] http://www.nactem.ac.uk/GENIA/tagger/

3.3. Final Dataset

The cleaned dataset we used consists of the 5000 abstracts provided by Dr. Wallace, tokenized using the GENIA tagger, with gold annotations for participant phrases aggregated from multiple Amazon Mechanical Turk annotations. The abstracts were split randomly and divided into train, development (dev), and test sets. The training set consists of 70% or 3500 of the abstracts, the dev set has 20% or 1000 abstracts, and the test set has 10% or 500 abstracts.

3.4. Evaluation scheme

To evaluate our models we calculate the precision, recall and f1 score of their predictions. The models are evaluated at the level of each token. This means, for a given abstract the precision would be the number of tokens correctly predicted to be part of the participant phrase divided by the number of tokens predicted to be part of the participant phrase.

3.5. MaxEnt Model

We implemented a simple MaxEnt model in order to establish a very basic baseline performance for the task of extracting participant phrases from abstracts. The python library scikit-learn's logistic regression implementation was used to train the MaxEnt models. A uni-gram and trigram model were implemented and tested. One-hot vectors were used to encode words and concatenated to form the trigrams. The models' regularization parameters were tuned on the dev set for best performance.

3.6. CRF Model

We implemented a Conditional Random Field classifier, which is the state-of-the-art classifier for our task at the moment, in order to compare its performance with Convolutional Neural Network.

Following the work in [11], we gave our tokens the following features:

- Features based on the token identity: one-hot features, word2vec features. Word2vec embeddings were from word vector file induced from Pubmed [13]. Also applied to the neighboring 3 tokens on each side of the token in question.
- Features based on GENIA tagger:
    - The token's POS tag, IOB tag, and named entity tag;
    - The type of phrase containing the token, e.g. noun phrase;
    - Whether it is the last token of the phrase;
    - The above features but applied to the neighboring tokens;
    - Whether each of the neighbors is in the same phrase as the token in question.

- Features based on the syntax:
  - Whether the token is inside parentheses;
  - Whether all letters of the token are capitalized;
  - Whether only the first letter of the token is capitalized;
  - The above features but applied to the neighboring tokens.

We began by using only one-hot features and incrementally increasing the number of features. For each set of features, we experimented by iterating over pairs of values for L1 and L2 regularization parameters, chosen from the set $\{0, 0.001, 0.01, 0.1, 1\}$. Then we chose the pair that performs best on the development set (this process is called *grid search*). For the optimal pair, we also adjusted the number of iterations of CRF training by increments of 25 to select the best value. Due to the constraint of resources, we did not implement cross-validation, and we kept in mind that randomness exists in our result.

As the number of features increases, so does the computation time, and so for our top models, we only picked L1 and L2 values from the set $\{0.01, 0.1, 1\}$ and did not adjust the number of iterations in training.

In order to understand what kinds of mistakes the CRF made, we performed the following analyses.

1. Unseen words analysis. Based on the way we split train, develop, and test dataset, 34% of the words in develop dataset are unseen, and 26% of the word in the test dataset are unseen. This may explain the slightly better F1 score of test dataset. To address potential issues associated with unseen words, we included word embeddings induced from PubMed [14].

2. Interval analysis. We separated the gold tags and our predicted tags of the development set into intervals, contiguous chunks of `P` tags. Then, we counted the number of gold intervals which are 1) Identical to; 2) Subintervals of; 3) Superintervals of; 4) Overlapping with (and not belonging to any previous groups), a predicted interval; and 5) Non-overlapping with any predicted interval. The rationale is to see whether the mistakes were mostly originating from the CRF not being able to identify Participants phrases at all, or from it being able to identify phrases but with incorrect boundaries (e.g. predicted intervals being too small or too large).

3. Sentence level analysis. Our current approach is to train the CRF model at abstract level with the belief that abstracts are different and abstract-level context are very useful for our task. However, the description of participants/patient is usually included in a sentence, and may follow certain patterns. Hence, the abstract-level model may be too complicated and noisy, while local-level prediction may be enough and may achieve better results. Therefore, we also trained our models at sentence level, and results are summarized in the result section.

4. Noun phrase analysis. Most Participants tags belong to noun phrases, but we did not utilize this information in our predictions. Therefore, we experimented restricting our evaluations (precision, recall, and F1 scores) to only tokens belonging to noun phrases. If the results are significantly different, then they signal that we were not using this information properly.

5. Comparison with previous work. We also had access to the dataset employed in [11] via Professor Wallace. We trained and tested our model on a subset of the dataset in order to compare the performance of our model with that of the model in the paper. We also tested the model trained on our 5000 abstracts on this previous dataset, and vice versa, in order to evaluate the similarities between the two datasets. Results are summarized in the Results section.

3.7. CNN Model

We also implemented a Convolutional Neural Network (CNN) model in an attempt to explore better classifiers. We chose a CNN model because it was more likely to converge with the limited amount of training data in our dataset. While LSTMs lend themselves better to sequence prediction, they require large training datasets and more computational power. Due to these limitations, we first explore CNNs for participant extraction, and plan to explore LSTMs in the future.

We adapted Denny Britz's sentence classification CNN model into three models that all aimed to predict participant tags: CNN by abstract, CNN by word, CNN with word2Vec [2].

The CNN architecture consists of an embedding layer of size M, followed by three convolutional layers, each with M filters, ReLU non-linearity and a max-pooling layer. The three convolutional layers have filter sizes of 3, 4 and 5, in that order. These are followed by a dropout layer with dropout probability 0.5 and last is a softmax layer. The input and the sizes of the layers - parameter M - are changed in the various models. This architecture can be easily visualized in Table 3.7.

The CNN models were trained on a 2015 Macbook Pro with 16 GB of RAM and an 3.1 GHz Intel Core i7 processor. This hardware proved to be a constraint on our models, limiting the size of our neural net and the size of the input.

3.7.1 CNN by Abstract

The first attempt at adapting Britz's sentence classification CNN architecture involved feeding classifying entire abstracts at a time. The input to the CNN consisted of an entire abstract as a vector where each word is represented

[2]`https://github.com/dennybritz/cnn-text-classification-tf`

General CNN Architecture
input (Vector of length N)
embedding layer - M units
conv3- M filters ReLU non-linearity maxpool
conv4- M filters ReLU non-linearity maxpool
conv5- M filters ReLU non-linearity maxpool
dropout (0.5)
softmax

Table 1. CNN model architecture. Input consists of a vector of length N. The embedding layer consists of M units and the convolutions layers comprise M filters. These parameters are changed in the variations of this model when we perform classification by abstract, by word and with word2vec.

by a unique integer. The CNN then outputted a binary vector with the same length as the input, where 1 corresponds to the word at the the same location of the input being part of a participant phrase. 0 means the word at the corresponding location is not part of a participant phrase. The length of the input vector was 78. Abstracts that were shorter have 0s as padding at the end of the vector. M, the number of units in the embedding layer and the number of filters was 128.

3.7.2 CNN by Word

We then implemented a simpler model, inputting a word and its preceding and following k words, and predicting only the tag for the center word. This simpler prediction task was better suited to the smaller architecture of the CNN. Each word is represented with an unique integer from 1 to V where V is the size of the vocabulary. The input vector to the CNN with k neighbor words is $2k + 1$ in length. The CNN was trained with $M = 64$ units in the embedding layer and filters.

3.7.3 CNN with Word2Vec

In order to solve the problem of unseen words, words that are present in the dev and test sets but not the training set, we attempted to use word2vec embeddings. These word vectors were downloaded, and were induced from Pubmed files by Pyysalo *et al.* [13]. These word vectors were 200 in length, far too long to use as input into the CNN. We reduced the vectors to length 10 and represented each word and its neighbor with its reduced word vector and its unique integer. These were scaled to have the same range. However, the CNN with this size of input and with $M = 128$ was unable to train on our hardware. Due to memory constraints, we were unable to train and investigate the effects of word embeddings.

4. Results and discussion

4.1. MaxEnt

The MaxEnt unigram and trigram models were used to generate a tag for each token, identifying it as either part of a participant phrase or not. This simple model tagged tokens n-grams independently and did not use Viterbi algorithm to find the optimal tag sequence. This was to provided a quick baseline against which we could evaluate the CRF and CNN model's performances. The trigram model performed substantially better than the unigram model, providing an F1 score of 0.414169 on the test set vs 0.251637 from the unigram model. The full results are provided in Table 2.

Table 2. Precision, Recall and F1 scores for the unigram and trigram MaxEnt models.

	Prec.	**Rec.**	**F1**
Unigram	0.579103	0.160742	0.251637
Trigram	0.466212	0.372579	0.414169

4.2. CRF

We started by using only one-hot encodings on the tokens themselves with no neighbors ($Base$), and we already observed a big improvement in F1 score with regard to MaxEnt model.

Then we experimented to add one-hot encodings of neigboring words on both sides ($Base - Neighbor$), and observed another increase in F1 score. Along with our diagnosis of the models and understanding the types of mistakes made by models, we added new features, adjusted prediction level (abstract level or sentence level), and tailored the evaluation algorithm.

Our best model uses the features described in Section 3.6, with 200 iterations of training, and regularized by $L1 = 1, L2 = 0.01$. The results are summarized in Table 3.

Table 3. Accuracies of our CRF models

	Dev set			**Test set**		
	Prec.	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Base	0.701	0.392	0.503	0.699	0.391	0.501
Base-Neighbor	0.746	0.476	0.581	0.755	0.482	0.588
Best model	0.775	0.552	0.645	0.783	0.527	0.630

Moreover, we found that changing the set of features in our best model resulted in similar accuracies, as long as we used a large enough number of features. The best accuracies are approximately 64%-64.5% for the development set and 62.5%-63% for the test set.

Our analyses of the mistakes of our model reveal the following.

1. Unseen words analysis. Since 34% of the words in development dataset are not seen in the training dataset, we introduce vector representation for each word with a moderate size of the vector length due to memory constraint of the computer. Therefore, words that are not seen in the training dataset will still have a non-zero feature vector. Comparing a model that only includes one-hot encoding for the word and its neigboring words and a model that includes both one-hot encoding and word2vec for the word and its neigboring words, we observe 0.3% increase in F1 score, mainly coming from the increase in Recall. With word2vec, we are able to identify higher proportion of words associated with `P`, and hence we also observe a decrease in precision.

Table 4. Accuracies of our one neigbor one-hot encoding model with or without w2v (dev set)

	Prec.	**Rec.**	**F1**
Only one-hot encoding	0.746	0.476	0.581
One-hot encoding and w2v	0.710	0.497	0.584

Currently, we reduced word vectors to a length of 30, and we were able to improve our model performance slightly. It could be improved more if we had computing resources to use the full vector. Since we do not observe a boost in the score, we continue to investigate other issues that may be responsible for the relatively low accuracies.

2. Interval analysis. The interval analysis shows that a large number of mistakes originate from the CRF not being able to detect Participants phrases at all, rather than detecting them with incorrect boundaries. Specifically, for our best model, 8127 out of 22102 gold tokens (37%) belong to intervals which do not overlap with any predicted interval at all. The results are summarized in Table 5.

Table 5. Analysis of gold intervals

	Intervals	**Tokens**
Identical	928	7269
Subinterval	277	2662
Superinterval	245	3645
Overlapping	35	399
Non-overlapping	1209	8127
Total	**2694**	**22102**

Considering that our Recall errors (1-Recall) are in the range of 45%-47%, this accounts for much of our errors. However, our techniques for detecting unseen phrases, such as word2vec, only succeeded in bringing down the number of Non-overlapping tokens from 10021 (with one-hot features) to 8127. At present, we do not know how to reduce this kind of mistakes further.

3. Sentence level analysis. Under the assumption that local level information would be enough, we trained our models at sentence level. Comparing the accuracies of these models, we observed that sentence-level model could have slightly higher or lower precision than abstract-level model, but higher recall, resulting F1 scores that differed by less than 1%. Because the differences were small, we decided to stay with abstract-level model.

4. Noun phrase analysis. Evaluating our best model only on noun phrases, we found that the resulting scores are only slighly higher. Specifically, the Precision, Recall, and F1 scores are 0.789, 0.573, and 0.664 on the development set, and 0.795, 0.545, and 0.647 on the test set. These scores are only 1.5%-2% higher than the original values. Thus, we concluded that we did not lose significant information by training and predicting the model on whole abstracts, and not restricting to only noun phrases.

5. Comparison with previous work. We used 135 abstracts from the dataset in [11], separated into 95 for training, 27 for development, and 13 for testing. The parameters are the same as in our best model.

The optimal hyperparameters are found to be 200 iterations, with regularizations $L1 = 0.1, L2 = 0.01$. The results are shown in Table 6.

Table 6. Accuracies of our best CRF model trained on the previous dataset

	Prec.	**Rec.**	**F1**
Dev set	0.923	0.404	0.562
Test set	1.000	0.420	0.592

The best system in the paper [11] achieves Precision, Recall, and F1 scores of 0.82, 0.71, and 0.76, respectively. Hence, our model achieves higher precision, lower recall, and lower F1 scores overall. We suspect that the lower scores are partly due to the fact that the task in [11] aims to extract Participants information from only sentences with numbers. However, we aim to extract Participants from the whole abstracts, and so did not include any number-related features as in [11] at all.

To confirm this, we used our model trained on the 5000 abstracts to predict tags for the 135 abstracts from the previous dataset. The resulting Precision, Recall, and F1 scores are 0.482, 0.382 and 0.426. Then, we used the model trained on the previous dataset to predict tags for the development set from the 5000 abstracts. The resulting Precision, Recall, and F1 scores are 0.104, 0.016, and 0.027. Thus, the model trained on the 5000 abstracts are able to somewhat generalize to the previous dataset, achieving the F1 score of 42.6%. But the model trained on the previous dataset cannot generalize to the 5000 abstracts, achieving the F1 score of 2.7%. We concluded that the previous dataset is more specific in nature, as it only deals with sentences that contain numbers, while our dataset is more general, as it concerns the tagging of whole abstracts.

The performance of our best model is compared to the

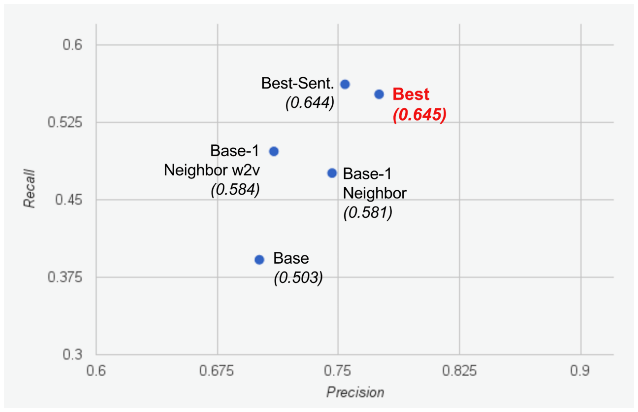

Figure 2. Scatterplot of Precision vs. Recall for the CRF models. Base is the model without any neighbors and only one-hot vector representation. Base-1 neighbor is with a neighbor, base-1 neighbor w2v is with word2vec embeddings for word representations. Best is the CRF model with all features and tuned regularization parameters, and Best-sentence is the same, but with sentences as input instead of abstracts.

basic CRF with only 1-hot vectors, the CRF with word embeddings, the CRF with all features and tuned regularization parameters, and the sentence-level CRF. The results of these models can be visualized in Figure 2.

4.3. CNN

The first experiment, running the CNN on the entire abstract, did not yield any results. Due to memory constraints as well as the size of the CNN, it did not converge during training. Our attempt at training the CNN with word embeddings in the input also did not yield results due to memory constraints.

Our experiments involved training the CNN to output the tags of tokens, given the input of the word and its k neighbors preceding and following. We experimented with changing the number of neighbors, as well as regularization parameters. We found the optimal number of neighbors to be around three or four, with more and less both adversely affecting performance. Three neighbors yielded an f1 score of 0.511610 and four neighbors yielded an f1 score of 0.511664. On the other hand, two neighbors had an f1 of 0.507797, and six had an f1 of 0.489445, as seen in Table 7.

Table 7. Precision, Recall and F1 scores on Test set for CNN model with different numbers of neighbors. k is the number of neighbors.

k	**Prec.**	**Rec.**	**F1**
2	0.601348	0.439435	0.507797
3	0.616955	0.436993	0.511610
4	0.632914	0.429402	0.511664
6	0.587397	0.419493	0.489445

There also was a large discrepancy between the CNNs' performance on the training set versus dev and test sets. With three neighbors in each direction and no regularization, the CNN had an f1 score of 0.886806 on the training set, 0.51717 on the dev set and 0.511610 on the test set, as seen in Table 8.

Table 8. Precision, Recall and F1 scores on for train, dev and test sets of CNN model with 3 neighbors, no regularization.

	Prec.	**Rec.**	**F1**
Train	0.914644	0.860612	0.886806
Dev	0.622706	0.442222	0.517170
Test	0.616955	0.436993	0.511610

We hypothesize that there are two potential causes of this discrepancy; the CNN is over-fitting to the training data or there is a problem of unseen words. To address the second problem, we would use word embeddings. However, due to memory constraints we were unable to test the effect of word embeddings. Our only option was to test whether potential overfitting could be improved with L2 regularization. As seen in Table 9, a small amount of L2 regularization helped, but larger values, such as 0.025 and 0.5, hurt performance.

Table 9. Precision, Recall and F1 scores on Test set for CNN with different L2 regularization parameters. k is the number of neighbors.

k	L2 Reg	**Prec.**	**Rec.**	**F1**
3	0.0	0.616955	0.436993	0.511610
3	0.001	0.622342	0.441732	0.516709
4	0.0	0.632914	0.429402	0.511664
4	0.01	0.602382	0.458238	0.520515
4	0.025	0.647059	0.398036	0.492879
4	0.05	0.606457	0.430403	0.503483

5. Conclusion

The CRF model with all features performed best, followed by the CNN. The simple MaxEnt model performed worse, unable to model more complex relationships. The results of the models on train and test sets can be visualized in Figure 3. The CNN model performed similarly to the CRF model with no additional features, however the CNN model performs far better on the training set. This leads us to believe that if we could compensate for the over-fitting, or use word embeddings to solve a possible unseen word problem, the CNN's performance could match or exceed the performance of the CRF. Additionally, we were able to use grid search to find the optimal regularization parameters for the CRF model. The CNN takes significantly longer to train, however, making it more difficult to optimize regularization parameters.

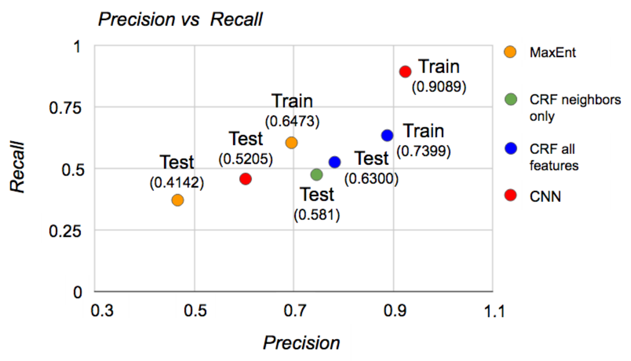

Figure 3. Scatterplot of Precision vs. Recall for the different models. Orange represents the trigram MaxEnt, green represents CRF with only one-hot vectors and neighbors, blue represents CRF with all features, and red represents CNN with 4 neighbors and L2 reg of 0.01.

The results of our experiments with the CNN and CRF lead us to believe current state-of-the-art performances of the CRF could be exceeded with deep learning. Given more computational power, training the CNN with word embeddings could yield better performance. Lastly, future experimentation should include LSTMs for sequence prediction, especially with additional data.

6. Member Contributions

6.1. Tiffany Wong

I implemented (1) the CNN Models (2) gold annotations using Nat's preprocessed, aggregated annnotations. Some of my implementation involved peer programming with Hansa and other times we took reins on specific aspects of the model (e.g. I worked more with data/input/ouput while Hansa did more with loss function/evalution.

6.2. Hansa Srinivasan

My work was primarily in the data prepossessing, implementing the simple MaxEnt models, implementing the CNN models, and writing the evaluation functions. Overall, I helped in brainstorming and designing the prepossessing system, as well as deciding on and planning CRFs, MaxEnt and CNNs for modeling. For the preprocessing, I peer-programmed with Tiffany the program which takes in the tokenized abstracts and the indexes of the participant phrases in order to produce arrays of tokens and matching arrays of their tags (Either 'P' or 'None'). This involved writing a program that took the indexes of participant phrases as character indexes from the original abstract and translating it to the corresponding tokens. I also wrote the evaluation functions to calculate precision, recall and f1 for all the models. I then quickly wrote up a simple unigram MaxEnt model and trigram MaxEnt model that tagged individual words as either part of participant phrases or not. Last, along with Tiffany, I helped adapt Denny Britz's sentence classification tensorflow CNN implementation for participant phrase extraction. We ran multiple variations on the CNN as described in the paper.

6.3. Jing Wen

I am most involved in the setup stage and CRF model implement of the project. At the early stage of the project, I reviewed relevant work, studied their approaches to similar problem, and searched for answers to questions, such as whether we should train our model at sentence level or abstract level, what a CRF classifier is, and what features have been proven useful. I also participate in brainstorming annotation scheme and data preprocessing. Then, I summarized features that would be useful, set up procedures to implement and tune CRF models, and adapt Byron Wallace's implementation of feature generation and CRF model fitting. Nat made experimentations with different features more flexible. Lastly, together with Nat, I experimented with CRF models and perform diagnosis analysis. I was responsible for unseen word analysis and sentence level analysis described in Section 3.6, and results are summarized in Section 4.2. To evaluate our best CRF model, Nat and I tested our model with a well-annotated dataset and compared the results of work done on the same data.

6.4. Nat Sothanaphan

My contributions lie mainly in three areas. First, I dealt substantively with the generations of gold annotations for data preprocessing, as described in Section 3. This included brainstorming and coding up the annotation scheme. Second, I, together with Jing Wen, implemented and experimented with the CRF model, as described in Section 3.6. In particular, I worked on features generations and finding the best model. Third, I performed the following analyses as described in Section 3.6: the interval analysis, the noun phrase analysis, and the comparison with previous work, the last one jointly with Jing Wen. The results from my investigations are summarized in Section 4.2

References

[1] M. F. M. Chowdhury and A. Lavelli. Disease mention recognition with specific features. *Proc. 2010 Workshop on Biomedical Natural Language Processing*, pages 83–90, 2005.

[2] B. C. W. Elisa Ferracane, Iain Marshall and K. Erk. Leveraging coreference to identify arms in medical abstracts: An experimental study.

[3] S. S. K. K. G. Lample, M. Ballesteros and C. Dyer. Neural architectures for named entity recognition. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[4] P. T. S. C. Hyung Paek, Yacov Kogan and M. Krauthammer. Shallow semantic parsing of randomized controlled trial reports. *AMIA Annual Symp Proc. 2006*, pages 604–608, 2006.

[5] Y. Kim. Convolutional neural networks for sentence classification. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[6] Y. L. X. L. L. Yao, H. Liu and M. W. Anwar. Biomedical named entity recognition based on deep neural network. *International Jounral of Hybrid Information Technology*, 2015.

[7] R. Leaman and G. Gonzalez. Banner: An executable survey of advances in biomedical named entity recognition. *Pacific Symposium on Biocomputing*, pages 13:652–663, 2008.

[8] G. D. M. Francis-Landau and D. Klein. Capturing semantic similarity for entity linking with convolutional neural networks. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[9] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction andweb-enhanced lexicons. *Seventh Conference on Natural Lan- guage Learning (CoNLL)*, 2003.

[10] E. G. N. Kalchbrenner and P. Blunsom. A convolutional neural network for modelling sentences. *In proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014.

[11] S. B. J. H. Rodney L. Summerscales, Shlomo Argamon and A. Schwartz. Automatic summarization of results from clinical trials.

[12] S. B. J. H. Rodney L. Summerscales, Shlomo Argamon and A. Schwartz. Identifying treatments, groups, and outcomes in medical abstracts. *The Sixth Midwest Computational Linguistics Colloquium(MCLC 2009)*, 2009.

[13] H. M. T. S. Sampo Pyysalo, Filip Ginter and S. Ananiadou. Distributional semantics resources for biomedical text processing. *LBM*, 2013.

[14] H. M. T. S. S. A. Sampo Pyysalo, Filip Ginter. Distributional semantics resources for biomedical text processing.

[15] F. Sha and F. Pereira. Shallow parsing with conditional random fields. *Conference on Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 213–220, 2003.

[16] C. Sutton and A. McCallum. Collective segmentation and labeling of distant entities in information extraction. *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*, 2004.

[17] C. Sutton and A. McCallum. An introduction to conditional random fields. 2010.

[18] B. E. Systems. Knowledge translation in the ed: How to get evidence used. 2016.

[19] Y. Tsuruoka and J. Tsujii. Bidirectional inference with the easiest-first strategy for tagging sequence data. *Proceedings of HLT/EMNLP 2005*, pages 467–474, 2005.