Shallow Training is cheap but is it good enough? Experiments with Medical Fact Coding

Ramesh Nallapati, Radu Florian
IBM T. J. Watson Research Center
1101 Kitchawan Road,
Yorktown Heights, NY 10598, USA
{nallapati,radu}@us.ibm.com

Abstract

A typical NLP system for medical fact coding uses multiple layers of supervision involving fact-attributes, relations and coding. Training such a system involves expensive and laborious annotation process involving all layers of the pipeline.

In this work, we investigate the feasibility of a shallow medical coding model that trains only on fact annotations, while disregarding fact-attributes and relations, potentially saving considerable annotation time and costs. Our results show that the shallow system, despite using less supervision, is only 1.4% F1 points behind the multi-layered system on Disorders, and contrary to expectation, is able to improve over the latter by about 2.4% F1 points on Procedure facts. Further, our experiments also show that training the shallow system using only sentence-level fact labels with no span information has no negative effect on performance, indicating further cost savings through weak supervision.

1 Introduction

Medical fact coding is the joint task of recognizing the occurrences of medical facts from electronic patient medical records expressed in natural language, and linking each occurrence of a fact to a specific code in a medical taxonomy such as SNOMED[1].

A representative sentence from a medical record along with its annotated facts is shown in Figure 1. In the parlance of traditional natural language processing, this task is roughly equivalent to the tasks of named-entity recognition (Nadeau and Sekine, 2007) and entity-linking[2] rolled into one.

Several open evaluations such as *ShARe-CLEF* (Pradhan et al., 2013) and *Semeval* (Pradhan et al., 2014) have been run recently to address the twin problems of fact recognition (recognizing occurrences of medical facts in text) and fact-coding (linking each occurrence of a fact to a pre-assigned code). These evaluations report performance numbers on both the tasks separately.

Often times, facts that occur in a medical text may not correspond to any pre-assigned codes, and are referred to as *CUI-less* facts in the *Semeval* evaluation. In the aforementioned evaluations, the systems are expected to output and are evaluated against CUI-less facts as well. However, in typical end-user applications such as medical billing, one does not care about the occurrences of unrecognized, non-billable facts. This work is targeted at such end applications where discovering only the occurrences of fact-codes recognized by a medical taxonomy is desirable. Consequently, CUI-less facts are ignored in our evaluation framework.[3]

In this work, we will focus only on the fact types of Disorders and Procedures, and use SNOMED as our medical taxonomy. We also use *Linkbase*[4] as our knowledge-base for descriptions of the fact codes.

2 Multi-layered Models for Fact Coding

Some of the unique characteristics of medical fact coding compared to the traditional entity recognition are as follows:

1. Unlike traditional entities, medical facts can be non-contiguous.
2. Unlike traditional entities, medical facts can be overlapping.

[1] http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

[2] http://www.nist.gov/tac/2013/KBP/EntityLinking/index.html

[3] In the official data of the Semeval task, it is reported that at least a quarter of the annotated facts are CUI-less (Pradhan et al., 2014). Hence ignoring these facts essentially renders a comparison of our evaluation numbers with the official Semeval numbers meaningless.

[4] http://www.nuance.com/for-healthcare/resources/clinical-language-understanding/ontology/index.htm

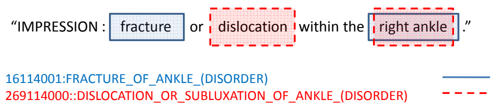

"IMPRESSION : fracture or dislocation within the right ankle ."
16114001:FRACTURE_OF_ANKLE_(DISORDER)
269114000::DISLOCATION_OR_SUBLUXATION_OF_ANKLE_(DISORDER)

Figure 1: An example sentence containing two non-contiguous and mutually overlapping facts: Fact 1 is composed of the words 'fracture', 'right', and 'ankle' while Fact 2 comprises the words 'dislocation', 'right' and 'ankle'. Note that both facts are non-contiguous, since there is a break between 'fracture' and 'right ankle' as well as between 'dislocation' and 'right ankle'. Likewise, both facts are overlapping with each other since they share the tokens 'right' and 'ankle'.

The example sentence in Figure 1 satisfies these two unique criteria. Since entities may not occur contiguously, a BIO (Begin-Inside-Outside) style sequence tagger is no longer directly applicable (Bodnari et al., 2013). Therefore, some researchers have used BIOT (Begin-Inside-Outside-BeTween) style coding to model the non-contiguous nature of the entities (Cogley et al., 2013), while others have attempted the approach of breaking down the entities into attributes that satisfy the contiguousness requirement of the BIO style taggers, and then reconstructing the original non-contiguous entities by tying the mentions of attributes together using relations (Gung, 2013). The former approach of BIOT tagging addresses the problem of non-contiguous entities but does not address the problem of overlapping entities, while the latter can address both the problems. Hence, in this work, we will use the latter approach as our *multi-layered* baseline system.

An example output produced by various stages of the multi-layered system for the example sentence of Figure 1 is shown in Figure 2. In this example, a Disorder fact is broken down into *attributes* such as *Disorder-Core*, *Body-site* and *Laterality*, whose occurrences are always contiguous. The mentions of these attributes are identified by a BIO-style sequence tagger such as the CRF (Lafferty et al., 2001). Next, a relations classifier is run on all pairs of attribute mentions in a given sentence. Finally, all attribute-mentions connected by relations are aggregated to produce fact-mentions, which are then lexically compared to a database of fact-descriptions to output the code of each mentioned fact, if one exists in the taxonomy.

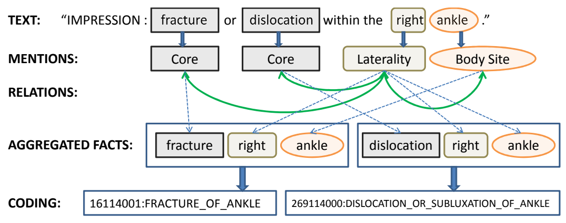

TEXT:
"IMPRESSION : fracture or dislocation within the right ankle ."
MENTIONS:
Core
Core
Laterality
Body Site
RELATIONS:
AGGREGATED FACTS:
fracture
right
ankle
dislocation
right
ankle
CODING:
16114001:FRACTURE_OF_ANKLE
269114000:DISLOCATION_OR_SUBLUXATION_OF_ANKLE

Figure 2: Output of various stages of the multi-layered pipeline on the example sentence in Figure 1. The mentions stage produces mentions of contiguous attributes, while the relations stage ties them together to produce larger, potentially non-contiguous entities. The final coding stage compares the fact-text to a database of fact-codes and their descriptions, and outputs the predicted medical codes.

Although the above mentioned strategy of a multi-layered system is very effective, annotating the training data required for all stages of the pipeline can be quite laborious and expensive. The motivating question for us in this work is whether we can eliminate some of the stages such as attribute-mentions and relations and still deliver comparable fact-coding performance. We call our approach *shallow coding* as it aims to reduce the number of layers of supervised data needed to train the model.

3 Shallow Coding Model

The multi-layered model uses a bottom-up approach starting from attribute-mentions and incrementally building all the way to fact-codes. In contrast, the shallow coding model uses a top-down approach wherein the entire text of the sentence is used as a query to retrieve matching facts directly. Note that this model does not use any sequence tagger to identify relevant spans of the text before matching them with the fact-descriptions. Since using the entire sentence for matching results in retrieval of many spurious facts, they are further analyzed in subsequent stages to output the final set of predicted facts.

This approach detects occurrences of only the facts with pre-assigned codes in the taxonomy since the retrieved candidate facts are those that already exist in the taxonomy. In contrast, the multi-layered model can also detect facts that have no pre-assigned codes since the fact-recognition step is independent of the taxonomy. Since our final objective in this work is to generate recognizable fact codes, the shallow coding model is an appro-

Fact Code (Fact-Type)	**Description**	**Source**
49436004 (Disorder)	atrial fibrillation	Linkbase
	af	DocID-131
	afib	DocID-236
195080001 (Disorder)	atrial fibrillation and flutter	Linkbase
	atrial flutter	DocID-567

Table 1: A sample of the database of fact codes and their descriptions collected from a union of Linkbase and training data annotations.

priate candidate for the task. It is however not an appropriate model for the *ShARe-CLEF* and *Semeval* evaluations that also care about unrecognizable (CUI-less) facts. Hence we are unable to evaluate this model using the official evaluations. We however, compare the shallow model with our own implementation of the multi-layered approach as a baseline.[5]

The rest of the section discusses in detail, the various stages of the shallow coding system in the order of their execution.

3.1 Information Retrieval (IR) Stage

An inverted-index of codes and their corresponding concept-descriptions, as provided in the Linkbase knowledge-base is first created. The index is also augmented with fact annotations from training data, treating each fact-mention as an additional description for the corresponding fact-code. Such augmentation with training annotations is necessary since the language used in SNOMED descriptions differs significantly from that used in medical reports.[6] To prevent overfitting at training time, we use a leave-one-out strategy where for each sentence in the training set, the retrieval results exclude fact-annotations from the document that the sentence belongs to. A few example descriptions augmented with training data annotations are shown in Table 1.

During the retrieval process for a given sentence, the sentence is first filtered for all punctuation and stop-words, and an initial search is performed using a sliding-window of length 3 words and the retrieved descriptions over all the window searches are pooled together by their fact-codes. The reason for using a sliding-window search is that it minimizes spurious long-distance matches with the sentence. Any facts that span longer than the sliding window size may be ranked lower in the initial search, but are boosted in the re-ranking stage as described below.

The pooled descriptions are then pruned by their retrieval scores to a maximum of 10 descriptions per code. We then re-rank the retrieved facts by the maximum of the inclusion scores of their retrieved descriptions computed with respect to the entire sentence:

$$\text{incl-score}(f, s) = \max_{d \in f}\left(\frac{\sum_{w \in (d \cap s)} \text{TF}(w, f)\text{IDF}(w)}{\sum_{w \in d} \text{TF}(w, f)\text{IDF}(w)}\right), \quad (1)$$

where f is a fact-code, s is a sentence, d is a description pooled into f, and w is a word-token in the description obtained after removing stop-words and stemming the remaining words. The inverse-document-frequency (IDF) weights are computed from the index of descriptions and not from the training documents, and term-frequency TF(w, f) is computed as the proportion of all descriptions in the fact f that contain the specific word w. The inclusion score is simply the IDF-weighted fraction of the description tokens contained in the sentence.

Further, to ensure that a single instance of the sliding window query does not dominate the search results, we also introduce a redundancy based penalty term into the inclusion score in Eqn. 1 where each word w in the numerator is discounted by $\log(1 + c(w))$, where $c(w)$ is the count of the number of times the word w is seen in the retrieved descriptions in the original ranking thus far.

The number of top ranking facts we return per sentence is a variable based on the sentence-length:

$$\text{n}(s) = \max(25, \min(3 \times \text{len}(s), 50)), \quad (2)$$

where n(s) is the number of facts returned for the sentence s, and len(s) is the number of processed tokens in s.

Note that we use unstemmed tokens in the initial search, but stemmed tokens for re-ranking, as this has been empirically found to improve performance by a small amount. In all our experiments, the initial search is performed us-

[5]The multi-layered approach should in fact be considered an upper-bound since it has access to more layers of labeled data.

[6]For example, one of the descriptions for Disorder code 49436004 is '*Atrial fibrillation*'. However, in medical reports, doctors typically use the short form '*afib*' to represent the same fact. Such variations can only be captured if we include training annotations as additional descriptions in the index.

Component	**Recall**
Lucene Search only	89.10
+ Inclusion-score-reranking	95.68
+ Redundancy penalty	96.12

Table 2: Contribution of various components towards the performance of the IR system. The numbers reported are on our Integris development set on Disorders facts.

ing default-ranking function as implemented in *Lucene*.[7]

Table 2 lists the contribution of Lucene search, re-ranking using inclusion score and using redundancy based penalty on our development set. The results indicates that while re-ranking is very critical towards achieving high recall, using a redundancy-based penalty to encourage diversity of results also is incrementally useful.

3.2 Alignment

All the descriptions retrieved from the previous step are then independently aligned word-to-word with the sentence text. For each description, we compute the alignment that has the minimum span but maximal possible matching, using a dynamic programming implementation that has a quadratic complexity in sentence length. We allow non-contiguous alignments in keeping with the fact a medical fact may consist of non-contiguous words. If multiple alignments satisfy this criterion, we return all such alignments. Note that the matches are computed using stemmed tokens, and order of matching is disregarded in computing the alignment. An example alignment where a single fact matches twice in a sentence via multiple descriptions is displayed in Figure 3.

For each description d aligned with the sentence s, an alignment score is computed as follows:

$$\begin{aligned}\text{Align-score}(a(d,s)) = \text{incl-score}(d,s) &\times\\ \text{tightness-ratio}(a(d,s)) &\times\\ (\sum_{w\in d\cap s}(\log(1.0+\text{IDF}(w)))),&\end{aligned} \tag{3}$$

where $a(d,s)$ is the alignment of the description with the sentence, incl-score(d,s) is computed as shown in Eqn. 1, and tightness-ratio$(a(d,s))$ is computed as follows:

$$\text{tightness-ratio}(a(d,s)) = \frac{\sum_{w\in d\cap s}(1)}{\text{span-len}(a(d,s))}, \tag{4}$$

[7]http://lucene.apache.org/

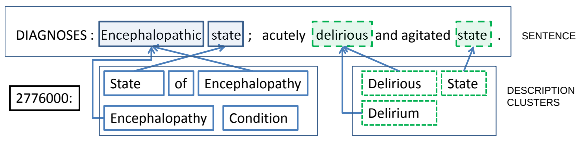


Figure 3: The Disorder code 2776000 occurs twice in the same sentence expressed as 'encephalopathic state' and 'delirious state'. Note that word order is ignored in computing the alignment. The words 'encephalopathy' and 'encephalopathic' match with each other due to matching of their stems. Also worth noting is the observation that the word 'state' in the description 'state of encephalopathy' could have been aligned with the last word in the sentence, but it does not happen since the alignment algorithm prefers maximal matches that have minimal span. The descriptions 'state of encepahlopathy' and 'encephalopathy condition' overlap over the word 'encephalopathy' and therefore form a cluster of descriptions. Likewise, the descriptions 'delirious state' and 'delirium' overlap over the word 'delirious' in the sentence, and form another cluster with the same fact code. These two clusters represent two distinct occurrences of the fact-code 2776000 in the sentence.

where span-len$(a(d,s))$ is the difference between the sentence-positions of right-most word in the alignment and the left-most word. Tightness ratio is higher for contiguous alignments than otherwise. As Eqn. 3 indicates, alignments that have a high inclusion score, tight alignment and a number of 'important' aligned words (as measured by their IDF scores) get high alignment-scores.

Since each fact can have multiple descriptions each of which may align in one or more ways with the sentence, we cluster the alignments of each fact based on their alignment positions in the sentence. In other words, each alignment-cluster c for a given fact contains all the descriptions that have at least one aligned position in common with another description in the cluster. Each such alignment-cluster constitutes an example, that goes to classifier-stages for further analysis. The alignment of a cluster with respect to a sentence is given by the alignment of the description in the cluster that has the best alignment score as given by Eqn. 3.

$$a(c,s) = \arg\max_{a(d,s)\ \forall\ d\in c} \text{Align-score}(a(d,s)) \tag{5}$$

3.3 Match Classifier

As mentioned above, each alignment cluster is treated as an example that is analyzed by the Match-classifier. At training time, the clusters are first mapped to positively annotated facts, such that each cluster is aligned with a positive fact in a greedy manner on a one-to-one basis. All the clusters mapped to positively annotated facts are considered positive examples, and the rest, negative.

Further, for training the Match-classifier, we only use those negative examples whose alignments do not overlap with those of any positive examples. This is done so that the Match-classifier accurately captures the semantics of similarity between the sentence and retrieved facts. The negative examples that overlap with the positive ones may have been annotated as negative for one of the following two reasons: (i) the retrieved fact is not related to the sentence, and (ii) the retrieved fact is related but is overruled because some other retrieved fact applies more accurately to the sentence. The Match-classifier is designed to deal with only case (i) above, hence we ignore the negative facts that overlap with any of the positive facts for training purposes. These facts will be handled separately by the Overlap-classifier in the next stage.

At test time, all the examples are run through the Match-classifier and classified as positive or negative for a given sentence. If the alignment of a given positively classified example does not overlap with that of any other example, it is directly output as positive for the given sentence. Else, it is sent to the subsequent stages for further analysis.

The following is the full list of features used in the Match-classifier.

Similarity features:
Unigrams: number of words and proportion of words in the description that are matched in the sentence, as well as the IDF-weighted versions of these two features.
Bigrams: number and proportion of bigrams in the description matched, as well as IDF-weighted versions of these features, where the IDF of a bigram is computed as the average of the IDFs of the pair of words in it.
Unordered bigrams: same as above, but ignoring the ordering of the bigrams.
Character-trigram features: each word in the description is mapped to a word in the sentence that has the highest number of character-level trigrams in common, and its similarity to the mapped word is measured in terms of the proportion of its character-trigrams matched. As features, we use the number and proportion of words in the description mapped, weighted by the character-trigram similarity scores.
Edit-distance based features: similar to character-trigram features, we map each word in the description to a word in the sentence using minimum edit-distance as the criterion. Next, we compute number and proportion of words matched using (1-edit-distance)/(word-length) as the similarity weight.
Synonym features: each word in the description is replaced with one of its synonyms from a dictionary[8], and computed unigram features with the replaced words, as above. The maximum value of the features over all synonyms is used as the final feature value.

For each of the above features, we compute its maximum value over all descriptions in the cluster and it as the final feature value.

Lexical features:
Matched and unmatched words: the matched words and their bigrams in the best alignment of the cluster, conjoined with the code, as well as the unmatched words within the span of the alignment conjoined with the code.
POS features: the parts-of-speech categories of matched words and their bigrams in the best alignment of the cluster, conjoined with the code, as well the POS categories of unmatched words within the span of the alignment, conjoined with the code.
Context words: Two words to the left and two words to right of the alignment, conjoined with the code of the description, used both as unigrams and bigrams.

Other features:
Alignment-based features: the tightness ratio (see Eqn. 4 above) of the best alignment for the cluster, average distance between the words in the align-

[8]The synonyms are generated in an unsupervised fashion based on descriptions that co-occur in a fact but differ by a single word, e.g.: ‘lung cancer’, and ‘pulmonary cancer’ are used to describe the same fact, hence ‘lung’ and ‘pulmonary’ are considered synonymous.

ment, and the number of unmatched words in the span of the alignment.

Prior features: the number and fraction of times the best aligned description in the cluster has been annotated with the given code in the training set.

Header features: the section-header name of the current sentence (E.g.: Diagnosis, History of illnesses, Discharge Summary, etc.) conjoined with the code of the matching description.

3.4 Overlap Classifier

All the examples classified as positive by the Match classifier that overlap with at least one other positively classified example are input to the Overlap classifier, that further analyzes these examples. The Overlap classifier uses all the features used in the Match-classifier as well as additional features based on the type of overlap between the two examples, and hierarchy relationship in SNOMED taxonomy between the two overlapping facts. We compute these features for each example with respect to all other examples that overlap with it. For a given example, even if the same feature fires with multiple overlapping examples, we do not add up the counts since we consider it as a binary feature.

Overlap features: For each example, a binary feature is computed to characterize whether its alignment (a) is subsumed by the alignment of the other example, (b) subsumes the alignment of the other example, (c) exactly equals the alignment of the other example or (d) overlaps without any of the three properties above. Other variants of this feature also include the feature conjoined with the overlapping words, and conjoined with the fact codes of the two examples.

Hierarchy features: For each example, we define a binary feature to characterize whether an example's fact-code is (a) a descendant, (b) an ancestor, (c) a sibling or (d) a co-parent of the other overlapping example's code in the taxonomy. Variants of this feature also include the feature conjoined with words in the overlap, and the fact codes of the two examples.

We only use positive examples that overlap with at least one other example, and negative examples that overlap with at least one positive example for training the classifier. This kind of sub-sampling of the training data allows the Overlap classifier to learn the semantics of how certain facts overrule other facts although both facts may be equally related to the sentence in question.[9]

Component	F1
Match Classifier only	78.65
+ Overlap Classifier	81.73
+ Rejection Rules	83.07

Table 3: Contribution of the two classifiers and the rejection rules towards the performance of the Shallow coding system. All numbers are reported on Disorder facts on the Integris development set.

At test time, each example is classified in an I.I.D. manner[10] , and the positively classified examples are then input to the final stage as described below.

3.5 Rejection Rules

In the final stage of the Shallow coding model, we apply two rules to potentially reject inconsistently classified examples from the previous stage. The two rules are listed below:

Rejection of subsumed examples: If the alignment of a positively classified example A strictly subsumes that of another positively classified example B, then example B is rejected and labeled as negative, since example A, with its longer alignment, is usually the more reliable and more specific fact. E.g.: Fact#195080001 with alignment 'atrial fibrillation and flutter' overrules Fact#49436004 that aligns with only 'atrial fibrillation', as its alignment is subsumed by the former's.

Rejection of ancestors: If the alignment of a positively classified example A overlaps with that of another positively classified example B and A is an ancestor of B, then the example A is rejected, since B, being the descendant is a more specific fact than A.

Note that the above rules are applied to pairs of positively classified and overlapping examples A and B, where A's confidence score as given by the Overlap classifier is higher than that of B.

[9]For example, if the medical text contains the phrase 'atrial fibrillation and flutter', it would match against both the facts shown in Table 1. However, Fact#195080001, being the more specific match is the correct fact and therefore overrules Fact#49436004.

[10]It is easy to see that the interactions of the overlapping examples may be modeled by a joint model such as the CRF. We have tried using CRFs in our experiments. Since the structure of the CRF can be arbitrary depending on the overlapping structure in a sentence, exact inference is hard. Hence, we used pseudo-likelihood for training the CRF and Gibbs sampling for testing, but it has not produced better results than the I.I.D. classifier using the features listed above. Hence we do not report the CRF's performance.

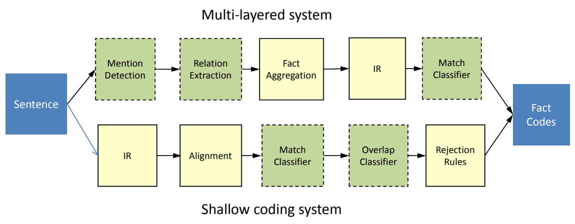


Figure 4: Comparison of various stages in the multi-layered pipeline vs the Shallow coding pipeline: the boxes with broken borders in either pipeline represent the stages that require labeled data. In the Shallow coding pipeline, Match-classifier and Overlap-classifier are the only stages that need training data, and they both use different slices of the same fact-span data for training. In contrast, the deep pipeline needs separate training data for mentions, relations and coding.

Dataset (FactType)	**Subset**	**nDoc**	**nSent**	**nFact**
Integris (Disorders)	train	409	14,218	10,906
	test	384	28,408	7,807
Mult-inst (Disorders)	train	12,370	484,822	204,124
	test	1,530	99,564	27,307
Proc-notes (Procedures)	train	1,624	71,151	17,996
	test	201	8,915	2,996

Table 4: Statistics of the datasets and corresponding fact-types used in our experiments. Integris is used purely as a development dataset on which we developed and tuned our models. We trained and evaluated Disorders on the multi-institution train and test datasets respectively. Similarly, we trained and evaluated our models on Procedures on the Proc-notes train and test splits. In the table, *nDoc* stands for number of documents, *nSent* for number of sentences, and *nFact* for number of facts.

Table 3 reports the incremental contribution of each classifier component to the overall performance of the shallow coding system. The numbers show that each component makes a significant contribution towards the overall performance.

Figure 4 compares the various stages involved in the multi-layered pipeline to the shallow coding system. The number of stages that need annotated data for training are indicated by boxes with broken edges in the figure, and is much less for the shallow system. In fact, both the stages that need training data in the shallow system, namely the Match classifier and Overlap classifier use different slices of the same training data, as described earlier.

Dataset	**Model**	**Prec.**	**Rec.**	**F1**
Integris	Mult-layer	82.76	84.51	**83.63**
	Shallow	85.27	80.98	83.07
Multi-Inst	Mult-layer	86.36	87.72	**87.03**
	Shallow	86.68	84.54	85.60
Proc-notes	Mult-layer	38.31	55.27	45.25
	Shallow	44.79	50.90	**47.65**

Table 5: Performance comparison: the shallow coding system is only about 0.6% F1 points below the multi-layered one on Disorders on the development set. On the unseen data of Multi-institution using the same fact-types, it is about 1.4% F1 behind the multi-layered model. On Proc-notes data involving Procedure facts, the shallow system is able to outperform the multi-layered architecture by 2.4% F1 points.

4 Experiments and Results

For tuning and developing our model, we used medical reports from an institution called *Integris*, which are partitioned into training and test sets. We tuned our model only on Disorder facts and evaluated them on both Disorders and Procedures. For evaluating the model on Disorders, we used another dataset from multiple institutions with its own train and test partitions which we call the *Multi-inst* dataset. For evaluating procedures, we used a dataset consisting of Procedure Notes documents with its own train and test partitions. The statistics of the datasets are summarized in Table 4.

The results of our experiments are summarized in Table 5. The shallow coding model is only about 1.4% F1 points behind the traditional multi-layered supervised model on Disorder facts, making it attractive for situations where cost savings are critical. On the more complex medical fact-types of Procedures, the shallow coding system outperforms the multi-layered system by 2.4 % F1 points. The fact that Procedure facts are harder is evident from the performance numbers of either system on Procedures compared with those on Disorders. A a few example Procedure facts, along with their attribute level annotations are displayed in Figure 5.

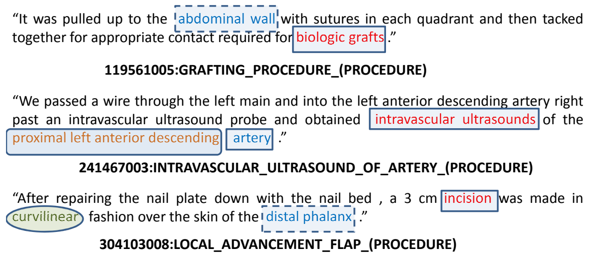
On complex fact-types involving long distance relations between the attributes, errors accumulate over the layers of the multi-layered system resulting in poorer performance.[11] In such a scenario, the shallow model may be more attractive.

[11]We are unable to show detailed comparison of the errors of the two models as our datasets are proprietary.

4.1 Weakly supervised training

Further, our experiments on both Disorders and Procedures showed that the performance of the shallow system practically remains unchanged even if it is provided with only sentence-level fact labels at training time, omitting their actual spans. The exact span of each fact in a training sentence is not needed since the model's alignment stage computes this information reasonably accurately, as long as it knows that the fact exists in the sentence. There was however, a caveat in our experiments: we retained the fact descriptions in the retrieval index that were created from the fact-spans in training sentences (see Section 3.1). Without these augmented descriptions, the performance of the system degrades considerably. Although this fact-span information was used only in the IR stage, it essentially means that the system did ultimately have access to fact-spans, and therefore is not a strict weakly-supervised model. Despite this important caveat, we believe that there is promise in a *weakly-supervised system* for medical fact coding, where facts are annotated only at sentence level without the exact span information, which may yield additional annotation cost savings. Note that such a weakly supervised model will not be applicable in the context of a sequence tagger that annotates mentions of facts or attributes first (such as the multi-layered model described in this paper or the ones described in (Bodnari et al., 2013) and (Gung, 2013)), since these models demand availability of annotated mention spans at training time.

Weakly supervised training has been successful in other information extraction tasks such as relation extraction (Surdeanu et al., 2012; Weston et al., 2013), but has not been used in the context of entity recognition, to the best of our knowledge. This may have been due to the fact that in traditional entity recognition, entities tend to be contiguous and non-overlapping, and therefore annotating entity spans may cause no significant overhead over annotating only sentences with entity-labels. Since these two properties do not hold true in medical fact recognition, weak supervision may be more attractive here. We hope this work paves the way for more future work in this direction.


"It was pulled up to the abdominal wall with sutures in each quadrant and then tacked together for appropriate contact required for biologic grafts."

119561005:GRAFTING_PROCEDURE_(PROCEDURE)

"We passed a wire through the left main and into the left anterior descending artery right past an intravascular ultrasound probe and obtained intravascular ultrasounds of the proximal left anterior descending artery ."

241467003:INTRAVASCULAR_ULTRASOUND_OF_ARTERY_(PROCEDURE)

"After repairing the nail plate down with the nail bed , a 3 cm incision was made in curvilinear fashion over the skin of the distal phalanx ."

304103008:LOCAL_ADVANCEMENT_FLAP_(PROCEDURE)

Figure 5: Examples of Procedure facts along with their attributes: the rectangle with sharp edges are *Procedure-Cores*, ones with broken edges are *Body-sites*, rectangles with rounded edges are *Lateralities*, and the ovals are *Approaches*. Note that the attributes for Procedures are more complicated and exhibit long-distance relations among themselves.

5 Conclusions and Future Work

In this work, we propose a new shallow coding model that learns to annotate medical facts that are overlapping and non-contiguous without using any attribute level annotations and relations annotations. Our work shows that this approach, while not being too far behind on Disorders, actually outperforms a more sophisticated and more deeply supervised model on Procedures.

As part of future work, we plan to investigate the feasibility of a weakly-supervised system that trains on only sentence-level fact labels. We believe that optimal performance may be achieved by a hybrid system that uses a small number of annotated training facts for generating an augmented retrieval index, and a large number of sentences with fact-labels but without span information, for training the classifiers. This would further reduce the annotation costs substantially.

We implemented a basic system combination of the shallow coding and the multi-layered models where the predictions of the multi-layered system are re-ranked based on the prediction of the shallow model for facts that are aligned between the two systems. However, such combination did not result in any significant improvement. As part of future work, we plan to build a meta-classifier that learns to effectively combine the outputs of the two systems using more sophisticated features, hopefully further improving over either system.

Acknowledgments

We are extremely thankful to *Nuance, Inc.* for supporting this work, and for providing all the resources including annotated medical reports for training and evaluation, and for providing access to the knowledge-bases needed in this work.

References

A. Bodnari, L. Deleger, T. Lavergne, A. Neveol, and P. Zweigenbaum. 2013. A supervised named-entity extraction system for medical text. In *Online Working Notes of the CLEF 2013 Evaluation Labs and Workshop*.

J. Cogley, N. Stokes, and J. Carthy. 2013. Medical disorder recognition with structural support vector machines. In *Online Working Notes of the CLEF 2013 Evaluation Labs and Workshop*.

J. Gung. 2013. Using Relations for Identification and Normalization of Disorders: Team CLEAR in the ShARe CLEF 2013 eHealth Evaluation Lab. In *Online Working Notes of the CLEF 2013 Evaluation Labs and Workshop*.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

D. Nadeau and S. Sekine. 2007. A survey of named entity recognition and classification. In *Linguisticae Investigationes*, number 30, pages 3–26.

S. Pradhan, N. Elhadad, B. R. South, D. Martinez, L. Christensen, A. Vogel, H. Suominen, W. W. Chapman, and G. Savova. 2013. Task 1: ShARe/CLEF eHealth Evaluation Lab. In *Online Working Notes of the CLEF 2013 Evaluation Labs and Workshop*.

S. Pradhan, N. Elhadad, W. Chapman, S. Manandhar, and G. Savova. 2014. SemEval-2014 Task 7: Analysis of Clinical Text. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 54–62.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 455–465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. *CoRR*, abs/1307.7973.