Lexically-Triggered Hidden Markov Models for Clinical Document Coding

Svetlana Kiritchenko **Colin Cherry**
Institute for Information Technology
National Research Council Canada
{Svetlana.Kiritchenko,Colin.Cherry}@nrc-cnrc.gc.ca

Abstract

The automatic coding of clinical documents is an important task for today's healthcare providers. Though it can be viewed as multi-label document classification, the coding problem has the interesting property that most code assignments can be supported by a single phrase found in the input document. We propose a Lexically-Triggered Hidden Markov Model (LT-HMM) that leverages these phrases to improve coding accuracy. The LT-HMM works in two stages: first, a lexical match is performed against a term dictionary to collect a set of candidate codes for a document. Next, a discriminative HMM selects the best subset of codes to assign to the document by tagging candidates as present or absent. By confirming codes proposed by a dictionary, the LT-HMM can share features across codes, enabling strong performance even on rare codes. In fact, we are able to recover codes that do not occur in the training set at all. Our approach achieves the best ever performance on the 2007 Medical NLP Challenge test set, with an F-measure of 89.84.

1 Introduction

The clinical domain presents a number of interesting challenges for natural language processing. Conventionally, most clinical documentation, such as doctor's notes, discharge summaries and referrals, are written in a free-text form. This narrative form is flexible, allowing healthcare professionals to express any kind of concept or event, but it is not particularly suited for large-scale analysis, search, or decision support. Converting clinical narratives into a structured form would support essential activities such as administrative reporting, quality control, biosurveillance and biomedical research (Meystre et al., 2008). One way of representing a document is to code the patient's conditions and the performed procedures into a nomenclature of clinical codes. The International Classification of Diseases, 9th and 10th revisions, Clinical Modification (ICD-9-CM, ICD-10-CM) are the official administrative coding schemes for healthcare organizations in several countries, including the US and Canada. Typically, coding is performed by trained coding professionals, but this process can be both costly and error-prone. Automated methods can speed-up the coding process, improve the accuracy and consistency of internal documentation, and even result in higher reimbursement for the healthcare organization (Benson, 2006).

Traditionally, statistical document coding is viewed as multi-class multi-label document classification, where each clinical free-text document is labelled with one or several codes from a pre-defined, possibly very large set of codes (Patrick et al., 2007; Suominen et al., 2008). One classification model is learned for each code, and then all models are applied in turn to a new document to determine which codes should be assigned to the document. The drawback of this approach is poor predictive performance on low-frequency codes, which are ubiquitous in the clinical domain.

This paper presents a novel approach to document coding that simultaneously models code-specific as well as general patterns in the data. This allows

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 742–751,
Portland, Oregon, June 19-24, 2011.

us to predict any code label, even codes for which no training data is available. Our approach, the lexically-triggered HMM (LT-HMM), is based on the fact that a code assignment is often indicated by short lexical triggers in the text. Consequently, a two-stage coding method is proposed. First, the LT-HMM identifies candidate codes by matching terms from a medical terminology dictionary. Then, it confirms or rejects each of the candidates by applying a discriminative sequence model. In this architecture, low-frequency codes can still be matched and confirmed using general characteristics of their trigger's local context, leading to better prediction performance on these codes.

2 Document Coding and Lexical Triggers

Document coding is a special case of multi-class multi-label text classification. Given a fixed set of possible codes, the ultimate goal is to assign a set of codes to documents, based on their content. Furthermore, we observe that for each code assigned to a document, there is generally at least one corresponding *trigger term* in the text that accounts for the code's assignment. For example, if an ICD-9-CM coding professional were to see "allergic bronchitis" somewhere in a clinical narrative, he or she would immediately consider adding code *493.9 (Asthma, unspecified)* to the document's code set. The presence of these trigger terms separates document coding from text classification tasks, such as topic or genre classification, where evidence for a particular label is built up throughout a document. However, this does not make document coding a term recognition task, concerned only with the detection of triggers. Codes are assigned to a document as a whole, and code assignment decisions within a document may interact. It is an interesting combination of sentence and document-level processing.

Formally, we define the document coding task as follows: given a set of documents X and a set of available codes C, assign to each document x_i a subset of codes $C_i \subset C$. We also assume access to a (noisy) mechanism to detect candidate triggers in a document. In particular, we will assume that an (incomplete) dictionary $D(c)$ exists for each code $c \in C$, which lists specific *code terms* associated with c.[1] To continue our running example: $D(\textit{493.9})$ would include the term "allergic bronchitis". Each code can have several corresponding terms while each term indicates the presence of exactly one code. A *candidate code* c is proposed each time a term from $D(c)$ is found in a document.

2.1 From triggers to codes

The presence of a term from $D(c)$ does not automatically imply the assignment of code c to a document. Even with extremely precise dictionaries, there are three main reasons why a candidate code may not appear in a document's code subset.

1. The context of the trigger term might indicate the irrelevancy of the code. In the clinical domain, such irrelevancy can be specified by a negative or speculative statement (e.g., "evaluate for pneumonia") or a family-related context (e.g., "family history of diabetes"). Only definite diagnosis of the patient should be coded.

2. There can be several closely related candidate codes; yet only one, the best fitted code should be assigned to the document. For example, the triggers "left-sided flank pain" (code *789.09*) and "abdominal pain" (code *789.00*) may both appear in the same clinical report, but only the most specific code, *789.09*, should end up in the document code set.

3. The domain can have code dependency rules. For example, the ICD-9-CM coding rules state that no symptom codes should be given to a document if a definite diagnosis is present. That is, if a document is coded with pneumonia, it should not be coded with a fever or cough. On the other hand, if the diagnosis is uncertain, then codes for the symptoms should be assigned.

This suggests a paradigm where a candidate code, suggested by a detected trigger term, is assessed in terms of both its local context (item 1) and the presence of other candidate codes for the document (items 2 and 3).

[1] Note that dictionary-based trigger detection could be replaced by tagging approaches similar to those used in named-entity-recognition or information extraction.

2.2 ICD-9-CM Coding

As a specific application we have chosen the task of assigning ICD-9-CM codes to free-form clinical narratives. We use the dataset collected for the 2007 Medical NLP Challenge organized by the Computational Medicine Center in Cincinnati, Ohio, hereafter refereed to as "CMC Challenge" (Pestian et al., 2007). For this challenge, 1954 radiology reports on outpatient chest x-ray and renal procedures were collected, disambiguated, and anonymized. The reports were annotated with ICD-9-CM codes by three coding companies, and the majority codes were selected as a gold standard. In total, 45 distinct codes were used.

For this task, our use of a dictionary to detect lexical triggers is quite reasonable. The medical domain is rich with manually-created and carefully-maintained knowledge resources. In particular, the ICD-9-CM coding guidelines come with an index file that contains hundreds of thousands of terms mapped to corresponding codes. Another valuable resource is Metathesaurus from the Unified Medical Language System (UMLS) (Lindberg et al., 1993). It has millions of terms related to medical problems, procedures, treatments, organizations, etc. Often, hospitals, clinics, and other healthcare organizations maintain their own vocabularies to introduce consistency in their internal and external documentation and to support reporting, reviewing, and meta-analysis.

This task has some very challenging properties. As mentioned above, the ICD-9-CM coding rules create strong code dependencies: codes are assigned to a document as a set and not individually. Furthermore, the code distribution throughout the CMC training documents has a very heavy tail; that is, there are a few heavily-used codes and a large number of codes that are used only occasionally. An ideal approach will work well with both high-frequency and low-frequency codes.

3 Related work

Automated clinical coding has received much attention in the medical informatics literature. Stanfill et al. reviewed 113 studies on automated coding published in the last 40 years (Stanfill et al., 2010). The authors conclude that there exists a variety of tools covering different purposes, healthcare specialties, and clinical document types; however, these tools are not generalizable and neither are their evaluation results. One major obstacle that hinders the progress in this domain is data privacy issues. To overcome this obstacle, the CMC Challenge was organized in 2007. The purpose of the challenge was to provide a common realistic dataset to stimulate the research in the area and to assess the current level of performance on the task. Forty-four teams participated in the challenge. The top-performing system achieved micro-averaged F1-score of 0.8908, and the mean score was 0.7670.

Several teams, including the winner, built pure symbolic (i.e., hand-crafted rule-based) systems (e.g., (Goldstein et al., 2007)). This approach is feasible for the small code set used in the challenge, but it is questionable in real-life settings where thousands of codes need to be considered. Later, the winning team showed how their hand-crafted rules can be built in a semi-automatic way: the initial set of rules adopted from the official coding guidelines were automatically extended with additional synonyms and code dependency rules generated from the training data (Farkas and Szarvas, 2008).

Statistical systems trained on only text-derived features (such as n-grams) did not show good performance due to a wide variety of medical language and a relatively small training set (Goldstein et al., 2007). This led to the creation of hybrid systems: symbolic and statistical classifiers used together in an ensemble or cascade (Aronson et al., 2007; Crammer et al., 2007) or a symbolic component providing features for a statistical component (Patrick et al., 2007; Suominen et al., 2008). Strong competition systems had good answers for dealing with negative and speculative contexts, taking advantage of the competition's limited set of possible code combinations, and handling of low-frequency codes.

Our proposed approach is a combination system as well. We combine a symbolic component that matches lexical strings of a document against a medical dictionary to determine possible codes (Lussier et al., 2000; Kevers and Medori, 2010) and a statistical component that finalizes the assignment of codes to the document. Our statistical component is similar to that of Crammer et al. (2007), in that we train a single model for all codes with code-

specific and generic features. However, Crammer et al. (2007) did not employ our lexical trigger step or our sequence-modeling formulation. In fact, they considered all possible code subsets, which can be infeasible in real-life settings.

4 Method

To address the task of document coding, our lexically-triggered HMM operates using a two-stage procedure:

1. Lexically match text to the dictionary to get a set of candidate codes;
2. Using features derived from the candidates and the document, select the best code subset.

In the first stage, dictionary terms are detected in the document using exact string matching. All codes corresponding to matches become candidate codes, and no other codes can be proposed for this document.

In the second stage, a single classifier is trained to select the best code subset from the matched candidates. By training a single classifier, we use all of the training data to assign binary labels (present or absent) to candidates. This is the key distinction of our method from the traditional statistical approach where a separate classifier is trained for each code. The LT-HMM allows features learned from a document coded with c_i to transfer at test time to predict code c_j, provided their respective triggers appear in similar contexts. Training one common classifier improves our chances to reliably predict codes that have few training instances, and even codes that do not appear at all in the training data.

4.1 Trigger Detection

We have manually assembled a dictionary of terms for each of the 45 codes used in the CMC challenge.[2] The dictionaries were built by collecting relevant medical terminology from UMLS, the ICD-9-CM coding guidelines, and the CMC training data. The test data was not consulted during dictionary construction. The dictionaries contain 440 terms, with 9.78 terms per code on average. Given these dictionaries, the exact-matching of terms to input documents is straightforward. In our experiments, this process finds on average 1.83 distinct candidate codes per document.

The quality of the dictionary significantly affects the prediction performance of the proposed two-stage approach. Especially important is the coverage of the dictionary. If a trigger term is missing from the dictionary and, as the result, the code is not selected as a candidate code, it will not be recovered in the following stage, resulting in a false negative. Preliminary experiments show that our dictionary recovers 94.42% of the codes in the training set and 93.20% in the test set. These numbers provide an upper bound on recall for the overall approach.

4.2 Sequence Construction

After trigger detection, we view the input document as a sequence of candidate codes, each corresponding to a detected trigger (see Figure 1). By tagging these candidates in sequence, we can label each candidate code as present or absent and use previous tagging decisions to model code interactions. The final code subset is constructed by collecting all candidate codes tagged as present.

Our training data consists of [document, code set] pairs, augmented with the trigger terms detected through dictionary matching. We transform this into a sequence to be tagged using the following steps:

Ordering: The candidate code sequence is presented in reverse chronological order, according to when their corresponding trigger terms appear in the document. That is, the last candidate to be detected by the dictionary will be the first code to appear in our candidate sequence. Reverse order was chosen because clinical documents often close with a final (and informative) diagnosis.

Merging: Each detected trigger corresponds to exactly one code; however, several triggers may be detected for the same code throughout a document. If a code has several triggers, we keep only the last occurrence. When possible, we collect relevant features (such as negation information) of all occurrences and associate them with this last occurrence.

Labelling: Each candidate code is assigned a binary label (present or absent) based on whether it appears in the gold-standard code set. Note that this

[2] Online at https://sites.google.com/site/colinacherry/ICD9CM_ACL11.txt

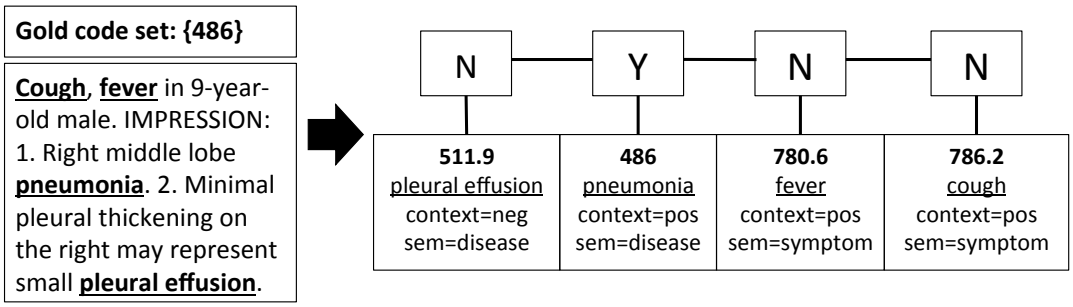

Figure 1: An example document and its corresponding gold-standard tag sequence. The top binary layer is the correct output tag sequence, which confirms or rejects the presence of candidate codes. The bottom layer shows the candidate code sequence derived from the text, with corresponding trigger phrases and some prominent features.

process can not introduce gold-standard codes that were not proposed by the dictionary.

The final output of these steps is depicted in Figure 1. To the left, we have an input text with underlined trigger phrases, as detected by our dictionary. This implies an input sequence (bottom right), which consists of detected codes and their corresponding trigger phrases. The gold-standard code set for the document is used to infer a gold-standard label sequence for these codes (top right). At test time, the goal of the classifier is to correctly predict the correct binary label sequence for new inputs. We discuss the construction of the features used to make this prediction in section 4.3.

4.3 Model

We model this sequence data using a discriminative SVM-HMM (Taskar et al., 2003; Altun et al., 2003). This allows us to use rich, over-lapping features of the input while also modeling interactions between labels. A discriminative HMM has two major categories of features: emission features, which characterize a candidate's tag in terms of the input document x, and transition features, which characterize a tag in terms of the tags that have come before it. We describe these two feature categories and then our training mechanism. All feature engineering discussed below was carried out using 10-fold cross-validation on the training set.

Transition Features

The transition features are modeled as simple indicators over n-grams of present codes, for values of n up to 10, the largest number of codes proposed by our dictionary in the training set.[3] This allows the system to learn sequences of codes that are (and are not) likely to occur in the gold-standard data.

We found it useful to pad our n-grams with "beginning of document" tokens for sequences when fewer than n codes have been labelled as present, but found it harmful to include an end-of-document tag once labelling is complete. We suspect that the small training set for the challenge makes the system prone to over-fit when modeling code-set length.

Emission Features

The vast majority of our training signal comes from emission features, which carefully model both the trigger term's local context and the document as a whole. For each candidate code, three types of features are generated: document features, ConText features, and code-semantics features (Table 1).

Document: Document features include indicators on all individual words, 2-grams, 3-grams, and 4-grams found in the document. These *n-gram features* have the candidate code appended to them, making them similar to features traditionally used in multiclass document categorization.

ConText: We take advantage of the ConText algorithm's output. ConText is publicly available software that determines the presence of negated, hypothetical, historical, and family-related context for a given phrase in a clinical text (Harkema et al., 2009).

[3] We can easily afford such a long history because input sequences are generally short and the tagging is binary, resulting in only a small number of possible histories for a document.

Features	gen.	spec.
Document		
n-gram		x
ConText		
current match		
context	x	x
only in context	x	x
more than once in context	x	x
other matches		
present	x	x
present in context = pos	x	x
code present in context	x	x
Code Semantics		
current match		
sem_type	x	
other matches		
sem_type, context = pos	x	x

Table 1: The emission features used in LT-HMM. Typeset words represent variables replaced with specific values, i.e. context $\in$ {pos,neg}, sem_type $\in$ {symptom,disease}, code is one of 45 challenge codes, n-gram is a document n-gram. Features can come in generic and/or code-specific version.

The algorithm is based on regular expression matching of the context to a precompiled list of context indicators. Regardless of its simplicity, the algorithm has shown very good performance on a variety of clinical document types. We run ConText for each trigger term located in the text and produce two types of features: features related to the candidate code in question and features related to other candidate codes of the document. Negated, hypothetical, and family-related contexts are clustered into a single *negative* context for the term. Absence of the negative context implies the *positive* context.

We used the following ConText derived indicator features: for the current candidate code, if there is at least one trigger term found in a positive (negative) context, if all trigger terms for this code are found in a positive (negative) context, if there are more than one trigger terms for the code found in a positive (negative) context; for other candidate codes of the document, if there is at least one other candidate code, if there is another candidate code with at least one trigger term found in a positive context, if there is a trigger term for candidate code c_i found in a positive (negative) context.

Code Semantics: We include features that indicate if the code itself corresponds to a disease or a symptom. This assignment was determined based on the UMLS semantic type of the code. Like the ConText features, code features come in two types: those regarding the candidate code in question and those regarding other candidate codes from the same document.

Generic versus Specific: Most of our features come in two versions: generic and code-specific. Generic features are concerned with classifying any candidate as present or absent based on characteristics of its trigger or semantics. Code-specific features append the candidate code to the feature. For example, the feature context=pos represents that the current candidate has a trigger term in a positive context, while context=pos:486 adds the information that the code in question is 486. Note that n-grams features are only code-specific, as they are not connected to any specific trigger term.

To an extent, code-specific features allow us to replicate the traditional classification approach, which focuses on one code at a time. Using these features, the classifier is free to build complex submodels for a particular code, provided that this code has enough training examples. Generic versions of the features, on the other hand, make it possible to learn common rules applicable to all codes, including low-frequency ones. In this way, even in the extreme case of having zero training examples for a particular code, the model can still potentially assign the code to new documents, provided it is detected by our dictionary. This is impossible in a traditional document-classification setting.

Training

We train our SVM-HMM with the objective of separating the correct tag sequence from all others by a fixed margin of 1, using a primal stochastic gradient optimization algorithm that follows Shalev-Shwartz et al. (2007). Let S be a set of training points (x, y), where x is the input and y is the corresponding gold-standard tag sequence. Let $\phi(x, y)$ be a function that transforms complete input-output pairs into feature vectors. We also use $\phi(x, y', y)$ as shorthand for the difference in features between

begin
Input: S, λ, n
Initialize: Set w_0 to the 0 vector
for $t = 1, 2 \ldots, n|S|$
Choose $(x, y) \in S$ at random
Set the learning rate: $\eta_t = \frac{1}{\lambda t}$
Search:
$y' = \arg\max_{y''} \left[\delta(y, y'') + w_t \cdot \phi(x, y'')\right]$
Update:
$w_{t+1} = w_t + \eta_t\big(\phi(x, y, y') - \lambda w_t\big)$
Adjust:
$w_{t+1} = w_{t+1} \cdot \min\left[1, \frac{1/\sqrt{\lambda}}{\|w_{t+1}\|}\right]$
end
Output: $w_{n|S|+1}$
end

Figure 2: Training an SVM-HMM

two outputs: $\phi(x, y', y) = \phi(x, y') - \phi(x, y)$. With this notation in place, the SVM-HMM minimizes the regularized hinge-loss:

$$\min_w \frac{\lambda}{2} w^2 + \frac{1}{|S|} \sum_{(x,y) \in S} \ell(w;\ (x, y)) \qquad (1)$$

where

$$\ell(w;\ (x, y)) = \max_{y'} \left[\delta(y, y') + w \cdot \phi(x, y', y)\right] \qquad (2)$$

and where $\delta(y, y') = 0$ when $y = y'$ and 1 otherwise.[4] Intuitively, the objective attempts to find a small weight vector w that separates all incorrect tag sequences y' from the correct tag sequence y by a margin of 1. λ controls the trade-off between regularization and training hinge-loss.

The stochastic gradient descent algorithm used to optimize this objective is shown in Figure 2. It bears many similarities to perceptron HMM training (Collins, 2002), with theoretically-motivated alterations, such as selecting training points at random[5] and the explicit inclusion of a learning rate η and a regularization term λ. The search step can be carried out with a two-best version of the Viterbi algorithm; if the one-best answer y'_1 matches the gold-standard y, that is $\delta(y, y'_1) = 0$, then y'_2 is checked to see if its loss is higher.

We tune two hyper-parameters using 10-fold cross-validation: the regularization parameter λ and a number of passes n through the training data. Using F1 as measured by 10-fold cross-validation on the training set, we found values of $\lambda = 0.1$ with $n = 5$ to prove optimal. Training time is less than one minute on modern hardware.

[4] We did not experiment with structured versions of δ that account for the number of incorrect tags in the label sequence y', as a fixed margin was already working very well. We intend to explore structured costs in future work.

[5] Like many implementations, we make n passes through S, shuffling S before each pass, rather than sampling from S with replacement $n|S|$ times.

	training	**test**
# of documents	978	976
# of distinct codes	45	45
# of distinct code subsets	94	94
# of codes with < 10 ex.	24	24
avg # of codes per document	1.25	1.23

Table 2: The training and test set characteristics.

5 Experiments

5.1 Data

For testing purposes, we use the CMC Challenge dataset. The data consists of 978 training and 976 test medical records labelled with one or more ICD-9-CM codes from a set of 45 codes. The data statistics are presented in Table 2. The training and test sets have similar, very imbalanced distributions of codes. In particular, all codes in the test set have at least one training example. Moreover, for any code subset assigned to a test document there is at least one training document labelled with the same code subset. Notably, more than half of the codes have less than 10 instances in both training and test sets. Following the challenge's protocol, we use micro-averaged F1-measure for evaluation.

5.2 Baseline

As the first baseline for comparison, we built a one-classifier-per-code statistical system. A document's code subset is implied by the set of classifiers that assign it a positive label. The classifiers use a feature set designed to mimic our LT-HMM as closely as possible, including n-grams, dictionary matches, ConText output, and symptom/disease se-

mantic types. Each classifier is trained as an SVM with a linear kernel.

Unlike our approach, this baseline cannot share features across codes, and it does not allow coding decisions for a document to inform one another. It also cannot propose codes that have not been seen in the training data as it has no model for these codes. However, one should note that it is a very strong baseline. Like our proposed system, it is built with many features derived from dictionary matches and their contexts, and thus it shares many of our system's strengths. In fact, this baseline system outperforms all published statistical approaches tested on the CMC data.

Our second baseline is a symbolic system, designed to evaluate the quality of our rule-based components when used alone. It is based on the same hand-crafted dictionary, filtered according to the ConText algorithm and four code dependency rules from (Farkas and Szarvas, 2008). These rules address the problem of overcoding: some symptom codes should be omitted when a specific disease code is present.[6]

This symbolic system has access to the same hand-crafted resources as our LT-HMM and, therefore, has a good chance of predicting low-frequency and unseen codes. However, it lacks the flexibility of our statistical solution to accept or reject code candidates based on the whole document text, which prevents it from compensating for dictionary or ConText errors. Similarly, the structure of the code dependency rules may not provide the same flexibility as our features that look at other detected triggers and previous code assignments.

5.3 Coding Accuracy

We evaluate the proposed approach on both the training set (using 10-fold cross-validation) and the test set (Table 3). The experiments demonstrate the superiority of the proposed LT-HMM approach over the one-per-code statistical scheme as well as our symbolic baseline. Furthermore, the new approach shows the best results ever achieved on the dataset, beating the top-performing system in the challenge, a symbolic method.

[6]Note that we do not match the performance of the Farkas and Szarvas system, likely due to our use of a different (and simpler) dictionary.

	Cross-fold	**Test**
Symbolic baseline	N/A	85.96
Statistical baseline	87.39	88.26
LT-HMM	89.39	89.84
CMC Best	N/A	89.08

Table 3: Micro-averaged F1-scores for statistical and symbolic baselines, the proposed LT-HMM approach, and the best CMC hand-crafted rule-based system.

System	Prec.	Rec.	F1
Full	90.91	88.80	89.84
-ConText	88.54	85.89	87.19
-Document	89.89	88.55	89.21
-Code Semantics	90.10	88.38	89.23
-Append code-specific	88.96	88.30	88.63
-Transition	90.79	88.38	89.57
-ConText & Transition	86.91	85.39	86.14

Table 4: Results on the CMC test data with each major component removed.

5.4 Ablation

Our system employs a number of emission feature templates. We measure the impact of each by removing the template, re-training, and testing on the challenge test data, as shown in Table 4. By far the most important component of our system is the output of the ConText algorithm.

We also tested a version of the system that does not create a parallel code-specific feature set by appending the candidate code to emission features. This system tags code-candidates without any code-specific components, but it still does very well, outperforming the baselines.

Removing the sequence-based transition features from our system has only a small impact on accuracy. This is because several of our emission features look at features of other candidate codes. This provides a strong approximation to the actual tagging decisions for these candidates. If we remove the ConText features, the HMM's transition features become more important (compare line 2 of Table 4 to line 7).

5.5 Low-frequency codes

As one can see from Table 2, more than half of the available codes appear fewer than 10 times in the

System	Prec.	Rec.	F1
Symbolic baseline	42.53	56.06	48.37
Statistical baseline	73.33	33.33	45.83
LT-HMM	70.00	53.03	60.34

Table 5: Results on the CMC test set, looking only at the codes with fewer than 10 examples in the training set.

System	Prec.	Rec.	F1
Symbolic baseline	60.00	80.00	68.57
All training data	72.92	74.47	73.68
One code held out	79.31	48.94	60.53

Table 6: Results on the CMC test set when all instances of a low-frequency code are held-out during training.

training documents. This does not provide much training data for a one-classifier-per-code approach, which has been a major motivating factor in the design of our LT-HMM. In Table 5, we compare our system to the baselines on the CMC test set, considering only these low-frequency codes. We show a 15-point gain in F1 over the statistical baseline on these hard cases, brought on by an substantial increase in recall. Similarly, we improve over the symbolic baseline, due to a much higher precision. In this way, the LT-HMM captures the strengths of both approaches.

Our system also has the ability to predict codes that have not been seen during training, by labelling a dictionary match for a code as present according to its local context. We simulate this setting by dropping training data. For each low-frequency code c, we hold out all training documents that include c in their gold-standard code set. We then train our system on the reduced training set and measure its ability to detect c on the unseen test data. 11 of the 24 low-frequency codes have no dictionary matches in our test data; we omit them from our analysis as we are unable to predict them. The micro-averaged results for the remaining 13 low-frequency codes are shown in Table 6, with the results from the symbolic baseline and from our system trained on the complete training data provided for comparison.

We were able to recover 49% of the test-time occurrences of codes withheld from training, while maintaining our full system's precision. Considering that traditional statistical strategies would lead to recall dropping uniformly to 0, this is a vast improvement. However, the symbolic baseline recalls 80% of occurrences in aggregate, indicating that we are not yet making optimal use of the dictionary for cases when a code is missing from the training data. By holding out only *correct* occurrences of a code c, our system becomes biased against it: all trigger terms for c that are found in the training data *must* be labelled absent. Nonetheless, out of the 13 codes with dictionary matches, there were 9 codes that we were able to recall at a rate of 50% or more, and 5 codes that achieved 100% recall.

6 Conclusion

We have presented the lexically-triggered HMM, a novel and effective approach for clinical document coding. The LT-HMM takes advantage of lexical triggers for clinical codes by operating in two stages: first, a lexical match is performed against a trigger term dictionary to collect a set of candidates codes for a document; next, a discriminative HMM selects the best subset of codes to assign to the document. Using both generic and code-specific features, the LT-HMM outperforms a traditional one-per-code statistical classification method, with substantial improvements on low-frequency codes. Also, it achieves the best ever performance on a common testbed, beating the top-performer of the 2007 CMC Challenge, a hand-crafted rule-based system. Finally, we have demonstrated that the LT-HMM can correctly predict codes never seen in the training set, a vital characteristic missing from previous statistical methods.

In the future, we would like to augment our dictionary-based matching component with entity-recognition technology. It would be interesting to model triggers as latent variables in the document coding process, in a manner similar to how latent subjective sentences have been used in document-level sentiment analysis (Yessenalina et al., 2010). This would allow us to employ a learned matching component that is trained to compliment our classification component.

Acknowledgements

Many thanks to Berry de Bruijn, Joel Martin, and the ACL-HLT reviewers for their helpful comments.

References

Y. Altun, I. Tsochantaridis, and T. Hofmann. 2003. Hidden Markov support vector machines. In *ICML*.

A. R. Aronson, O. Bodenreider, D. Demner-Fushman, K. W. Fung, V. K. Lee, J. G. Mork, A. Nvol, L. Peters, and W. J. Rogers. 2007. From indexing the biomedical literature to coding clinical text: Experience with MTI and machine learning approaches. In *BioNLP*, pages 105–112.

S. Benson. 2006. Computer assisted coding software improves documentation, coding, compliance and revenue. *Perspectives in Health Information Management*, CAC Proceedings, Fall.

M. Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *EMNLP*.

K. Crammer, M. Dredze, K. Ganchev, P. P. Talukdar, and S. Carroll. 2007. Automatic code assignment to medical text. In *BioNLP*, pages 129–136.

R. Farkas and G. Szarvas. 2008. Automatic construction of rule-based ICD-9-CM coding systems. *BMC Bioinformatics*, 9(Suppl 3):S10.

I. Goldstein, A. Arzumtsyan, and Uzuner. 2007. Three approaches to automatic assignment of ICD-9-CM codes to radiology reports. In *AMIA*, pages 279–283.

H. Harkema, J. N. Dowling, T. Thornblade, and W. W. Chapman. 2009. Context: An algorithm for determining negation, experiencer, and temporal status from clinical reports. *Journal of Biomedical Informatics*, 42(5):839–851, October.

L. Kevers and J. Medori. 2010. Symbolic classification methods for patient discharge summaries encoding into ICD. In *Proceedings of the 7th International Conference on NLP (IceTAL)*, pages 197–208, Reykjavik, Iceland, August.

D. A. Lindberg, B. L. Humphreys, and A. T. McCray. 1993. The Unified Medical Language System. *Methods of Information in Medicine*, 32(4):281–291.

Y. A. Lussier, L. Shagina, and C. Friedman. 2000. Automating ICD-9-CM encoding using medical language processing: A feasibility study. In *AMIA*, page 1072.

S. M. Meystre, G. K. Savova, K. C. Kipper-Schuler, and J. F. Hurdle. 2008. Extracting information from textual documents in the electronic health record: a review of recent research. *Methods of Information in Medicine*, 47(Suppl 1):128–144.

J. Patrick, Y. Zhang, and Y. Wang. 2007. Developing feature types for classifying clinical notes. In *BioNLP*, pages 191–192.

J. P. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch. 2007. A shared task involving multi-label classification of clinical free text. In *BioNLP*, pages 97–104.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. 2007. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *ICML*, Corvallis, OR.

M. H. Stanfill, M. Williams, S. H. Fenton, R. A. Jenders, and W. R. Hersh. 2010. A systematic literature review of automated clinical coding and classification systems. *JAMIA*, 17:646–651.

H. Suominen, F. Ginter, S. Pyysalo, A. Airola, T. Pahikkala, S. Salanter, and T. Salakoski. 2008. Machine learning to automate the assignment of diagnosis codes to free-text radiology reports: a method description. In *Proceedings of the ICML Workshop on Machine Learning for Health-Care Applications*, Helsinki, Finland.

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *Neural Information Processing Systems Conference (NIPS03)*, Vancouver, Canada, December.

A. Yessenalina, Y. Yue, and C. Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *EMNLP*.