Automated Preamble Detection in Dictated Medical Reports

Wael Salloum, Greg Finley, Erik Edwards, Mark Miller, and David Suendermann-Oeft
EMR.AI Inc
90 New Montgomery St #400
San Francisco, CA 94105, USA
david@emr.ai

Abstract

Dictated medical reports very often feature a preamble containing metainformation about the report such as patient and physician names, location and name of the clinic, date of procedure, and so on. In the medical transcription process, the preamble is usually omitted from the final report, as it contains information already available in the electronic medical record. We present a method which is able to automatically identify preambles in medical dictations. The method makes use of state-of-the-art NLP techniques including word embeddings and Bi-LSTMs and achieves preamble detection performance superior to humans.

1 Introduction

For decades, medical dictation and transcription has been used as a convenient and cost-effective way to document patient-physician encounters and procedures and bring reports into a form which can be stored in an electronic medical record (EMR) system, formatted as an out-patient letter, etc (Häyrinen et al., 2008; Johnson et al., 2008; Meystre et al., 2008; Holroyd-Leduc et al., 2011; Kalra et al., 2012; Logan, 2012; Hyppönen et al., 2014; Campanella et al., 2015; Moreno-Conde et al., 2015; Alkureishi et al., 2016; Ford et al., 2016). While dictated speech has traditionally been transcribed by humans (such as clinical assistants or professional transcription personnel), sometimes in multiple stages, it is common nowadays for speech recognition technology to be deployed in the *first stage* to increase transcription speed and cope with the enormous amount of dictated episodes in the clinical context (Hammana et al., 2015; Hodgson and Coiera, 2016; Edwards et al., 2017).

In its purest form, a speech recognizer transforms spoken into written words, as exemplified in Figure 1. Obviously, this raw output will have to undergo multiple transformation steps to format it in a way it can be stored in an EMR or sent out as a letter to the patient, including: formatting numbers, dates, units, etc.; punctuation restoration (Salloum et al., 2017b); and processing physician normals.

Furthermore, dictated reports often contain metadata in a preamble containing information not intended to be copied into the letter, such as patient and physician names, location and name of the clinic, date of procedure, and so on. Rather, the metadata serves the sole purpose of enabling realigning dictations with a particular record or file, in case this alignment is not otherwise possible (usually, metadata in medical transcription systems is automatically retrieved from the EMR system and inserted into the outpatient letter). See Figure 2 for the same text sample as Figure 1 with the preamble highlighted and the above post-processing rules applied.

In a *second stage*, medical transcriptionists take the speech recognizer output and perform a post-editing exercise and quality check before entering the final report into the EMR or sending it off as an outpatient letter. This stage usually involves the removal of metadata, i.e. the preamble, from the dictation's main text body. To facilitate this procedure, this paper explores techniques to automatically mark preambles.

It is worth noting that the accurate detection of preambles in dictated reports is a non-trivial task, even for humans. Clinical dictations may (a) contain metadata at multiple places throughout the report (see Figure 3 for an example), (b) or no such data at all, (c) feature sentences convolving metadata and general narrative, or (d) have grammati-

```
this is doctor mike miller dictating
a maximum medical improvement slash
impairment rating evaluation for
john j o h n doe d o e social one
two three four five six seven eight
nine service i_d one two three four
five six seven eight nine service
date august eight two thousand and
sixteen subjective and treatment to
date the examinee is a thirty-nine
year-old golf course maintenance
worker with the apache harding park
who was injured on eight seven two
thousand sixteen
```

Figure 1: Raw output of a medical speech recognizer.

This is Dr Mike Miller dictating a Maximum Medical Improvement/Impairment Rating Evaluation for John Doe.
SSN: 123-45-6789
Service ID: 123 456 789
Service Date: 08/08/16

Subjective and Treatment:
To date, the examinee is a 39 year-old golf course maintenance worker with the Apache Harding Park who was injured on 08/07/16.

Figure 2: Output of post-processor with preamble highlighted.

cal inaccuracies and lack overall structure caused by the spontaneous nature of dictated speech, including the total absence of punctuations. To systematically quantify the task's complexity, we also determined the human baseline performance of detecting the preamble in clinical dictation.

This paper is structured as follows: After discussing related work in Section 2, we describe the corpus and determine the human baseline in Section 3.3. Section 4 provides details on the techniques we used for the automated detection of preambles, followed by evaluation results and discussion in Section 5. We conclude the paper and provide an outlook on future work in Section 6.

This is Dr Mike Miller.
The patient is a baking associate over at Backwerk.
Today's date is 03/10/2016.
The patient noted he strained his back while he was helping his mother move some household items.

Figure 3: Example of a report intertwining preamble and main body. Physician name and date of the visit are commonly considered preamble, whereas the patient's profession and employer are not. When spontaneously dictating, physicians sometimes remember to mention preamble statements only after they have already started the main body narrative, such as the date of visit in this example.

2 Related Work

To our knowledge, the problem of automated preamble detection in medical transcriptions has not been addressed before. That said, we do build upon classic methods in NLP: specifically, our system is a generalization of sequence tagging, which has seen use in other tasks such as part-of-speech tagging, shallow parsing or chunking, named entity recognition, and semantic role labeling. Traditionally, sequential tagging has been handled using either generative methods, such as hidden Markov models (Kupiec, 1992), or sequence-based discriminative methods, such as conditional random fields (Lafferty et al., 2001; Sha and Pereira, 2003).

More modern approaches have shown performance gains and increased generalizability with neural networks (NNs). Collobert and colleagues (Collobert and Weston, 2008; Collobert et al., 2011) successfully apply NNs to several sequential NLP tasks without the need for separate feature engineering for each task. Their networks feature concatenated windowed word vectors as inputs or, in the case of sentence-level tasks, a convolutional architecture to allow interaction over the entire sentence.

However, this approach still does not cleanly capture nonlocal information. In recent years, recurrent NN architectures, often using gated recurrent (Cho et al., 2014; Tang et al., 2016; Dey and Salem, 2017) or long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997; Hammerton, 2003), have been applied with excellent results to various sequence labeling problems. Many linguistic problems feature dependencies at longer distances, which LSTMs are better able to capture than convolutional or plain recurrent approaches. Bidirection LSTM (Bi-LSTM) networks (Graves and Schmidhuber, 2005; Graves et al., 2005; Wöllmer et al., 2010) also use future context, and recent work has shown advantages of Bi-LSTM networks for sequence labeling and named entity recognition (Huang et al., 2015; Chiu and Nichols, 2015; Wang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016; Plank et al., 2016).

In some approaches, tag labels from NN outputs are combined in a final step, such as conditional random fields, especially when the goal is to apply a single label to a continuous sequence of tags. Our architecture, as described in Section 4, also utilizes a post-tagging step to define a clear preamble endpoint.

3 Corpus and Inter-Annotator Agreement

In this section we report on the corpus used for this study, the methodology for computing inter-annotator agreement, and we analyze the preamble split positions in more detail.

3.1 The Data

A total of 10,517 dictated medical reports were transcribed by a team of professional medical transcriptionists (MTs) organized in a private crowd as described in (Salloum et al., 2017a). The produced transcriptions were raw, i.e., only lowercase alphabetic characters, dash, and underscore were permitted, resulting in output as shown in Figure 1.

In a separate round, we sent these transcribed reports to a private crowd of MTs to acquire a total of five annotation jobs per file. Since we cannot specify all types of information that are expected to be found in preambles ahead of time, we let the MTs, who are well experienced in transcribing medical dictations, determine the exact split position that, in their opinion, separated preamble text from main report. This approach allows us to harvest the wisdom of the crowd and define what they agree on as the ground truth, which we can then learn automatically.

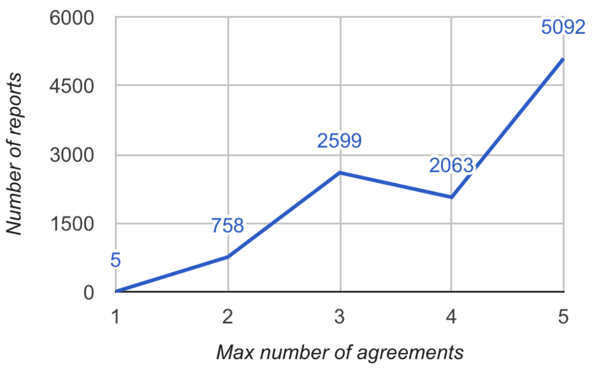

Figure 4: Histogram of the maximum number of exact agreements obtained for the annotated reports

3.2 Inter-Annotator Agreement

In order to establish a corpus with reliable labels which subsequently can be used to measure human accuracy and train and test the automatic preamble detector, we defined a gold-standard annotation to be one where at least three annotators agreed on the exact split between preamble and main body. Figure 4 shows a histogram of the frequency of number of agreements. For example, out of the 10,517 reports, 5,092 have all annotators agree on the split position while only 5 reports have 5 different annotations. By reducing the corpus to only those reports with at least three annotators in agreement about the split position, we ended up with a total of 9,754 reports, or 92.75% of the original body of data. 4.4% of the reports were *not* annotated by all five annotators, constituting the majority of omitted files. The lack of annotations is presumably due to annotators not being sure how to split, or due to oversight. Missing annotations makes it harder for such files to match the three-agreement threshold.

Overall, it became clear that the lack of guidelines on specific types of phenomena featured in the preamble, such as including or excluding a patient's employer, led to disagreements that ultimately caused the exclusion of reports—although note that nearly half of included reports do have at least one dissenting opinion. This analysis is specifically helpful for designing new guidelines for the next round of annotations, which will lead to cleaner data fed to our system.

We split the 9,754 reports randomly into training and test sets. Table 1 shows some statistics about the data split. The test set out-of-vocabulary

Set	# reports	# tokens	# types
Training Set	8,711	3,335,588	30,707
Test Set	1,043	415,491	13,507

Table 1: Corpus statistics.

(OOV) rate against the training set is 10.76% (1,454 types).

In order to quantify the inter-annotator agreement, we compared each annotator against the majority vote, resulting in the following annotator split accuracy scores: 83.22%, 86.09%, 86.09%, 86.58%, 88.20%. The average inter-annotator agreement score, 86.04%, will serve as standard of comparison in this paper.

3.3 Analysis of Preamble Split Positions

As motivated in the introduction, the use of preambles in medical dictations is not very consistent. E.g., a good amount of dictations do not contain a preamble at all, whereas others contain multiple, even others convolve preamble and main text so much that it is very hard to determine the exact split position. In this work, annotators were required to provide a single split tag at the location were they found the boundary to be most appropriate. If annotators did not find any preamble in the dictation, the tag was placed in front of the first token of the dictation.

Figure 5 displays a histogram of the split position in reports. The vast majority of split positions are below 100 tokens into the dictation (compared to the average total token count for the dictations in our corpus of 385; see Table 1 for exact statistics). There are 319 reports (3.3%) with no preamble and, hence, split position 0.

If we define the problem as a sequence tagging problem where every token in a preamble is tagged with I-P (Inside Preamble) and every token in the main report is tagged with I-M (Inside Main), we get the histogram in Figure 6.

4 Approach

Although the training data contains 3.3 M tokens, the evaluation is at the level of reports, of which we have only 8.7 K examples. We determined from preliminary experiments that this limited amount of examples is not enough to train an end-to-end neural network to predict the split position. Therefore, we use a two-step approach to preamble detection:

1. A sequence **tagger** that labels every word in the input sequence with one of two tags: I-P (Inside Preamble) and I-M (Inside Main). This tagger leverages the large number of tokens in our data, as opposed to the small number of example reports, which leads to near perfect tagging accuracy.

2. A report **splitter** that determines heuristically at what position to split the tagged report into preamble and main. This splitter attempts to correct the tagger's mistakes.

4.1 The Tagging Model

Like other recent work, our model is based on LSTM NNs. We experimented with both unidirectional and bidirectional networks. The stack consists of an embedding layer (see Section 4.3 for details), a (Bi-)LSTM layer, and a time-distributed dense layer with softmax activation (illustrated in Figure 7). For the present study, we used Keras with TensorFlow backend (Chollet, 2015; Abadi et al., 2016; Chollet, 2017). We applied a categorical cross-entropy cost function and Adam optimization (Kingma and Ba, 2014).

In addition to word meaning and context, the analysis we did in Section 3.3 motivates that the correct prediction of tags depends on the location of words in the report as well (Figure 5 and Figure 6). Therefore, instead of tagging the input sequence using a sliding window like many taggers do, we have a fixed size input to the network comprising the first 512 tokens of the report. Words after this limit are truncated. We add padding for reports with less than 512 tokens. Informal experiments showed that varying the window length to 256 or 1024 tokens deteriorated preamble detection performance.

Since the data we have is limited in size, we use word vectors pretrained on large amounts of unlabeled text collected from medical reports and medical dictation transcriptions. This transfer learning technique is often used in deep learning approaches to NLP since the vectors learned from massive amounts of unlabeled text can be transfered to another NLP task where labeled data is limited and might not be enough to train the embedding layer.

4.2 The Heuristic Splitter

The training examples of the tagging model always have preamble tags (I-P) preceding main re-

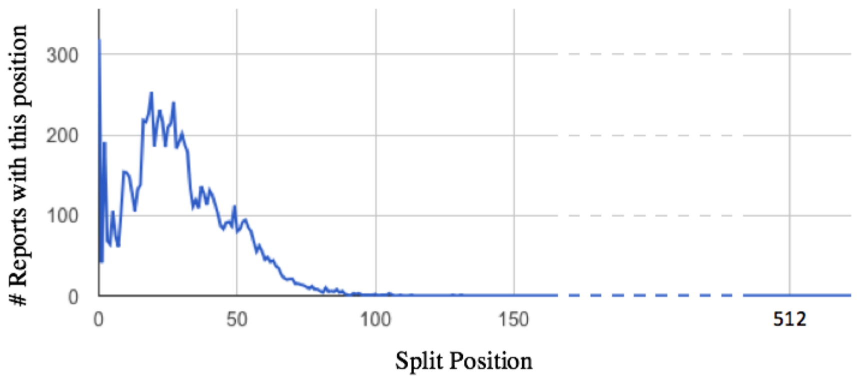

Figure 5: A histogram of the split position in our training set. A point of interest is the split at position 0, which indicates that 319 reports have no preamble text. The longest preamble text ends at position 131, after that the curve stays on 0.

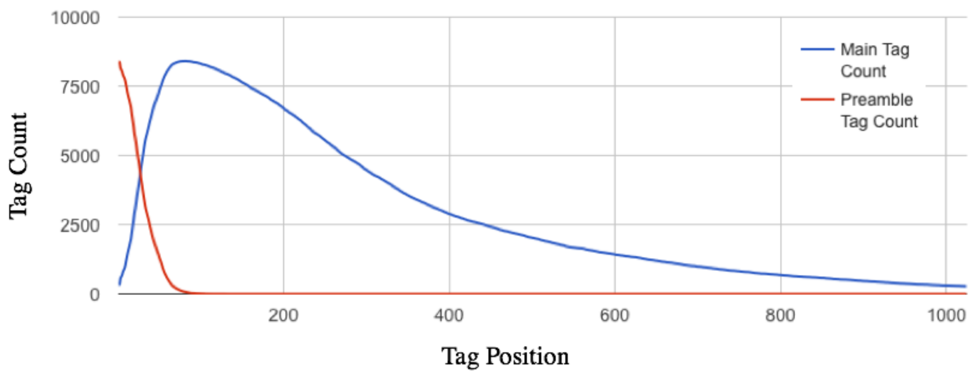

Figure 6: Frequency of the two tags at positions in the first 1000 words of reports. The last preamble tag, I-P, appears at position 131, after that the red curve stays on 0. The main tag, I-M, starts at position 1 with a value 319, and goes up as the report grows longer. The main tag curve then falls down as longer reports are less frequent than shorter ones.

port tags (I-M). Nevertheless, the neural network sometimes produces mixed sequences of I-P and I-M. An example of such output starts with I-P, switches briefly to I-M, then back to I-P, and then to I-M. This situation requires another system to find the exact position in which we need to split preamble from main report. We use simple heuristics to determine the split position as explained in Algorithm 1.

The algorithm looks for concentrations of preamble and main tag sequences. It initializes the split position it is trying to predict, $splitPos$, and a sequence counter, $counter$, to 0. While scanning the tagged sequence, it increases $counter$ if it sees an I-P (Line 6) and decreases it if it sees an I-M (Line 11). $counter > 0$ means that we have seen a long enough I-P tag sequence since the last I-M tag to consider the text so far to be preamble and the previous I-M tags to be errors. However, the next I-M tag will set restart the counter (Line 9) and set $splitPos$ to the previous position (Line 10). Lines 12-13 handle the edge case where the sequence ends while $counter > 0$, which means that the whole report is preamble.

It is important to point out that our splitter is biased, by design, to vote in favor of including more words in main (i.e., shorter preambles). The reason for this bias is that in applications where the main text is more valued than preamble (e.g., to create a formatted note), we take the safe option not to omit content words.

4.3 Pretrained Word Embeddings

Word embeddings were trained offline using the original implementation of the word2vec package (Mikolov et al., 2013b,a). All vectors are 200 dimensions and trained using 15 iterations of the continuous bag-of-words model over a window of 8 words, with no word count minimum.

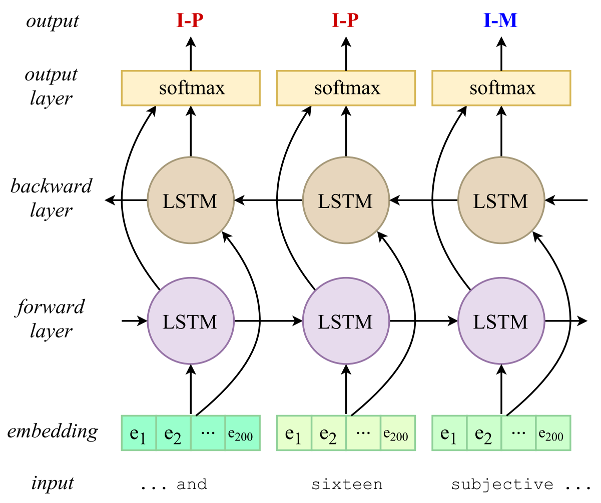

Figure 7: The NN stack using Bi-LSTM. An embedding at each word step is fed into forward and backward LSTM layers, which are fully connected to a softmax-activated output layer. (For the unidirectional LSTM, the backward layer is omitted.)

Algorithm 1 The Heuristic Splitter.

```
 1: $splitPos \leftarrow 0$ // predicted split position.
 2: $counter \leftarrow 0$ // sequence counter.
 3: for $pos := 1 \rightarrow length(tags)$ do
 4:     switch $tags[pos]$ do
            // ... padding is ignored.
 5:         case I-P
 6:             $counter$++
 7:         case I-M
 8:             if $counter > 0$ then
 9:                 $counter \leftarrow 0$ // reset
10:                 $splitPos \leftarrow pos - 1$
11:             $counter$−−
12: if $counter > 0$ then
13:     return $length(predictedTags)$
14: else
15:     return $splitPos$
```

We experimented with three sets of embeddings, each trained on cumulatively more text:

- “SplitEmb” was trained on the same transcriptions as the tagging model (plus those on which only two annotators agreed on the split), with the insertion of a line break at the split between the preamble and main text. This break causes word2vec not to train on co-occurrences of tokens on either side of the split, hypothetically leading to decreased similarity between words typically found inside and outside of preambles. (3.7 M tokens total.)
- “SplitTransEmb” added more transcribed medical dictations which were not part of the preamble-annotated set. (8.3 M tokens.)
- “SplitTransRepEmb” added formatted medical reports processed to look like transcriptions—numerals spelled out, punctuation removed, etc. (60 M tokens.)

5 Evaluation

As a first sanity check, we measured the preamble tagging accuracy on the token level. In other words, we determined how many of the tokens in the test set were correctly tagged as being either part of the preamble or the main body. In this task, our system achieved an accuracy of 99.80%, with only 816 mismatches among the total of 415,491 tokens in the test set.

As motivated in Section 3.2, the ultimate performance measure we are using counts how many perfect splits the preamble detector found, i.e. the split accuracy. Table 2 shows detailed results of the systems introduced in Section 4, comparing all pre-trained word embedding models across two embedding schemes (trainable vs. frozen) and for both Uni- and Bi-LSTM. The best overall system uses Bi-LSTMs and frozen embeddings, performing at 89.84% split accuracy. In comparison, as calculated earlier, the human split accuracy on our corpus was determined to be 86.04% which constitutes a statistically significant difference. The fact that our automated preamble detection system outperforms humans demonstrates the strength of the presented methods in exploiting synergistic effects across a crowd of annotators.

We were also interested in the effectiveness of the heuristic splitter introduced in Section 4.2. We therefore determined results for both Uni-LSTM (75.74%) and Bi-LSTM (87.44%) when leaving out the splitter. Compared to the individual best results for Uni- and Bi-LSTMs in Table 2, this constitutes a difference of 8.25% and 2.4%, demonstrating a clear positive impact of the heuristic splitter.

	Test OOVs (first 512)		LSTM				Bi-LSTM			
			Trainable Emb.		Frozen Emb.		Trainable Emb.		Frozen Emb.	
	# types	*rate*	*# PS*	*%*	*# PS*	*%*	*# PS*	*%*	*# PS*	*%*
Fully-trained Emb.	n/a	n/a	754	72.29%	n/a	n/a	863	82.74%	n/a	n/a
SplitEmb	1133	8.59%	809	77.56%	839	80.44%	911	87.34%	916	87.82%
SplitTransEmb	905	6.86%	809	77.56%	846	81.11%	899	86.19%	**937**	**89.84%**
SplitTransRepEmb	230	1.74%	798	76.51%	876	83.99%	907	86.96%	925	88.69%

Table 2: Evaluation of our LSTM and Bi-LSTM models across all pretrained word embedding models. The first column shows the different pretrained word embedding models we used. The "Test OOVs" column shows the OOV count and rate against each pretrained embedding model. This only includes types in the first 512 words of the report (that are passed to the NN) which contain 13,186 types out of the 13,507. Columns with title "Trainable Emb." report results where backpropagation is allowed to update the pretrained embedding layer after it is loaded, while columns with title "Frozen Emb." does not allow such updates. # PS is the number of Perfect Splits.

6 Conclusion and Future Work

The work presented in this paper shows yet again that careful design and execution of state-of-the-art NLP techniques when applied to traditionally manual tasks (in this case, the detection of preambles in medical dictations) can approach or even surpass human performance. We assume that the presented NLP stack with Bi-LSTMs makes use of the wisdom of the crowd: it exploits the fact that, even though the annotators working on this task were professional MTs, the provided guidelines on how to tell preambles from main body were not very detailed.

In future investigations, we would like to see how more elaborate annotation guidelines can improve human performance and what impact the improved annotations have on the performance of an automated preamble detector. It is specifically interesting to investigate how situations of intertwined preamble and main body, as exemplified in Figure 3, can be resolved by clearer guidelines or, alternatively, by an annotation scheme allowing for more than a single hard split.

We are also interested to further enhance the automatic preamble detector by combining the tagger and splitter into a joint neural network model, or by implementing a transfer learning step which reuses the learned tagger weight in a neural-network-based splitter.

References

M Abadi, A Agarwal, P Barham, E Brevdo, Z Chen, C Citro, GS Corrado, A Davis, J Dean, M Devin, S Ghemawat, I Goodfellow, A Harp, G Irving, M Isard, Y Jia, R Jozefowicz, L Kaiser, M Kudlur, J Levenberg, D Mané, R Monga, S Moore, D Murray, C Olah, M Schuster, J Shlens, B Steiner, I Sutskever, K Talwar, P Tucker, V Vanhoucke, V Vasudevan, F Viégas, O Vinyals, P Warden, M Wattenberg, M Wicke, Y Yu, and X Zheng. 2016. Tensorflow: large-scale machine learning on heterogeneous distributed systems. *arXiv* 1603(04467):1–19.

MA Alkureishi, WW Lee, M Lyons, VG Press, S Imam, A Nkansah-Amankra, D Werner, and VM Arora. 2016. Impact of electronic medical record use on the patient-doctor relationship and communication: a systematic review. *J Gen Intern Med* 31(5):548–560.

P Campanella, E Lovato, C Marone, L Fallacara, A Mancuso, W Ricciardi, and ML Specchia. 2015. The impact of electronic health records on healthcare quality: a systematic review and meta-analysis. *Eur J Public Health* 26(1):60–64.

JPC Chiu and E Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv* 1511(08308):1–14.

K Cho, B van Merriënboer, D Bahdanau, and Y Bengio. 2014. On the properties of neural machine translation: encoder-decoder approaches. *arXiv* 1409(1259):1–9.

F Chollet. 2015. Keras: deep learning library for theano and tensorflow. https://keras.io/.

F Chollet. 2017. *Deep learning with Python*. Manning, Shelter Island, NY.

R Collobert and J Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proc ICML*. ACM, pages 160–167.

R Collobert, J Weston, L Bottou, M Karlen, K Kavukcuoglu, and P Kuksa. 2011. Natural language processing (almost) from scratch. *J Mach Learn Res* 12(8):2493–2537.

R Dey and FM Salem. 2017. Gate-variants of gated recurrent unit (gru) neural networks. *arXiv* 1701(05923):1–5.

E Edwards, W Salloum, GP Finley, J Fone, G Cardiff, M Miller, and D Suendermann-Oeft. 2017. Medical speech recognition: reaching parity with humans. In *SPECOM*. page 10 p.

E Ford, JA Carroll, HE Smith, D Scott, and JA Cassell. 2016. Extracting information from the text of electronic medical records to improve case detection: a systematic review. *J Am Med Inform Assoc* 23(5):1007–1015.

A Graves, S Fernández, and J Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *Proc ICANN*. Springer, pages 799–804.

A Graves and J Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proc IJCNN*. IEEE, volume 4, pages 2047–2052.

I Hammana, L Lepanto, T Poder, C Bellemare, and M-S Ly. 2015. Speech recognition in the radiology department: a systematic review. *HIM J* 44(2):4–10.

J Hammerton. 2003. Named entity recognition with long short-term memory. In *Proc HLT-NAACL*. ACL, volume 4, pages 172–175.

K Häyrinen, K Saranto, and P Nykänen. 2008. Definition, structure, content, use and impacts of electronic health records: a review of the research literature. *Int J Med Inform* 77(5):291–304.

S Hochreiter and J Schmidhuber. 1997. Long short-term memory. *Neural Comput* 9(8):1735–1780.

T Hodgson and EW Coiera. 2016. Risks and benefits of speech recognition for clinical documentation: a systematic review. *J Am Med Inform Assoc* 23(e1):e169–e179.

JM Holroyd-Leduc, D Lorenzetti, SE Straus, L Sykes, and H Quan. 2011. The impact of the electronic medical record on structure, process, and outcomes within primary care: a systematic review of the evidence. *J Am Med Inform Assoc* 18(6):732–737.

Z Huang, W Xu, and K Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv* 1508(01991):1–10.

H Hyppönen, K Saranto, R Vuokko, P Mäkelä-Bengs, P Doupi, M Lindqvist, and M Mäkelä. 2014. Impacts of structuring the electronic health record: a systematic review protocol and results of previous reviews. *Int J Med Inform* 83(3):159–169.

SB Johnson, S Bakken, D Dine, S Hyun, E Mendonca, F Morrison, T Bright, T Van Vleck, J Wrenn, and P Stetson. 2008. An electronic health record based on structured narrative. *J Am Med Inform Assoc* 15(1):54–64.

D Kalra, B Fernando, Z Morrison, and A Sheikh. 2012. A review of the empirical evidence of the value of structuring and coding of clinical information within electronic health records for direct patient care. *Inform Prim Care* 20(3):171–180.

DP Kingma and J Ba. 2014. Adam: a method for stochastic optimization. *arXiv* 1412(6980):1–9.

J Kupiec. 1992. Robust part-of-speech tagging using a hidden markov model. *Comput Speech Lang* 6(3):225–242.

JD Lafferty, A McCallum, and FCN Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc ICML*. ACM, pages 282–289.

G Lample, M Ballesteros, S Subramanian, K Kawakami, and C Dyer. 2016. Neural architectures for named entity recognition. *arXiv* 1603(01360):1–11.

J Logan. 2012. Electronic health information system implementation models a review. *Stud Health Technol Inform* 178:117–123.

X Ma and EH Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv* 1603(01354):1–12.

SM Meystre, GK Savova, KC Kipper-Schuler, and JF Hurdle. 2008. Extracting information from textual documents in the electronic health record: a review of recent research. *Yearb Med Inform* 2008:128–144.

T Mikolov, K Chen, GS Corrado, and J Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv* 1301(3781):1–13.

T Mikolov, W-t Yih, and G Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proc HLT-NAACL*. ACL, volume 13, pages 746–751.

A Moreno-Conde, D Moner, WD da Cruz, MR Santos, JA Maldonado, M Robles, and D Kalra. 2015. Clinical information modeling processes for semantic interoperability of electronic health records: systematic review and inductive analysis. *J Am Med Inform Assoc* 22(4):925–934.

B Plank, A Søgaard, and Y Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv* 1604(05529):1–7.

W Salloum, E Edwards, S Ghaffarzadegan, D Suendermann-Oeft, and M Miller. 2017a. Crowdsourced continuous improvement of medical speech recognition. In *Proc AAAI Wrkshp Crowdsourcing*. AAAI, San Francisco, CA.

Wael Salloum, Greg Finley, Erik Edwards, Mark Miller, and David Suendermann-Oeft. 2017b. Deep learning for punctuation restoration in medical reports. In *Proceedings of the ACL BioNLP Workshop*. Association for Computational Linguistics.

F Sha and F Pereira. 2003. Shallow parsing with conditional random fields. In *Proc HLT-NAACL*. ACL, volume 1, pages 134–141.

Y Tang, Z Wu, H Meng, M Xu, and L Cai. 2016. Analysis on gated recurrent unit based question detection approach. In *Proc Interspeech*. ISCA, San Francisco, CA, pages 735–739.

P Wang, Y Qian, FK Soong, L He, and H Zhao. 2015. A unified tagging solution: bidirectional lstm recurrent neural network with word embedding. *arXiv* 1511(00215):1–10.

M Wöllmer, F Eyben, A Graves, B Schuller, and G Rigoll. 2010. Bidirectional lstm networks for context-sensitive keyword detection in a cognitive virtual agent framework. *Cognit Comput* 2(3):180–190.