Master's Thesis

De-Identification of free text

Sameed Hayat

Examiner: Prof. Dr. Hannah Bast
Advisers: Francesco Alda

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Chair of Algorithms and Data Structures

July 08th, 2019

Writing Period

26. 11. 2018 – 08. 07. 2019

Examiner

Prof. Dr. Hannah Bast

Second Examiner

Dr. Fang Wei-Kleiner

Advisers

Francesco Alda

Declaration

I hereby declare, that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.
I hereby also declare, that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

Place, Date	Signature

Abstract

Sharing data in the form of text is important for a wide range of activities but it also raises a concern about privacy when sharing data that could be sensitive. Text-based patient medical records are a vital resource in medical research. Processing of a large amount of medical data can be done in order to derive meaningful statistical information from such data as well as for other research purposes. The use of such data usually imposes problems with privacy of patients' personally identifiable information. Automated text de-identification is a solution for removing all the sensitive information from documents. However, this is a challenging task due to the unstructured form of textual data and the ambiguity of natural language. In this thesis, we developed two models that use semi-supervised approaches. These approaches use mix of labeled and unlabeled data to remove personal information from the data. First approach uses contextualized character-level word embedding trained on large corpus of unlabeled data. This model outperforms the state-of-the-art systems on i2b2 2014 de-identification challenge dataset without using hand-curated feature. This model yields an F1-score of 97.99%, with a recall 97.52% and a precision of 98.46%. Whereas, the second approach uses Cross-View Training (CVT) algorithm, which aims at improving the performance of the model by incorporating unlabeled data. Our results show that this approach can aid in increasing the performance of the model on the de-identification task. This model outperforms the state-of-the-art systems on i2b2 2014 de-identification challenge dataset without using hand-curated feature. Our best performing model yields an F1-score of 97.99%, with a recall 97.52% and a precision of 98.46%. We also showed with our experiments that as the amount of data decreases the performance of semi-supervised approaches grow larger over purely supervised learning approaches.

Contents

List of Figures

List of Tables

1 Introduction

Whenever companies and research institutions collect, store or use some information regarding a person, they are legally required to protect this individual's privacy. For instance, personally identifiable information from text-based documents, e.g. emails or reports, must be removed before such documents can be shared, even within the same organization. This process is known as de-identification. De-identification aims at preventing a person's identity from being disclosed or linked with the information that is processed or shared.

After the inception of electronic health records (EHRs), the next natural step in advancement of medical research is the utilization of this large amount of medical data to derive meaningful statistical information as well as for other research purposes.

In addition to honouring patient-doctor confidentiality, which helps gain patients' trust, enabling them to share information with more confidence, data protection laws like, Europe's General Data Protection Regulation (GDPR, 2016) and the Privacy Rule of the American Health Insurance Portability Accountability Act (HIPAA, 1996) impose restrictions on the use of such data by unauthorized personnel.

Unlike GDPR, however, HIPAA allows the use of such data without patient's consent for research purposes if all protected health information (PHI) is removed and replaced with surrogate information, so that the data cannot be used to trace back the original subject without sacrificing the information contained within it.

The major problem however, is that any commercially or otherwise available tools for processing data and linguistics prove to be futile because such data not only contains jargons specific to the field, which are not only part of standard vocabulary, but also, in a lot of cases, cannot be distinguished from the personally identifiable information because a lot of the medical jargon contains nomenclature named after their discoverers. Additionally, such specific information is written

in a hurried manner for the eyes of the experts. Such writings are often sentence fragments prone to having spelling mistakes. Following can be common sources of ambiguity causing a simpler algorithm to fail:

- Overlap of words that can be names of people and medical terminologies.
- The names may be very uncommon or misspelled.
- There is no standard data formatting scheme (bulleted data, paragraphs, tabular form etc).

The simplest solution for such a task would be to hire a professional who can identify such information and replace it with suitable alternatives. But given the amount of such datasets, the financial and temporal costs can rise quickly. Besides, the data protection laws limit the access of this data to a restricted group of people. In this thesis, we try to find out answers of following questions:

> *Could semi-supervised machine learning methods help in improving the performance of model on the task of de-identification in medical domain? Also, how these approaches perform as compared to supervised approaches, when dealt with a problem of unavailability of large amount of annotated data.*

1.1 Contributions

In this thesis, we worked on two semi-supervised approaches for de-identification of patient notes. Our key contributions are:

- A model that adds character-level contextual embeddings to vanilla sequence labelling model. This model is able to perform better than current state-of-the-art system with an F1-score of 97.99%
- A model that uses semi-supervised algorithm based on the principle of self-training, where effective use of labelled and unlabelled data improves overall model representation power. This model performs better than current state-of-the-art system that doesn't use an hand-curated features, in terms of recall, with a score of 97.69%
- Experiments on variable training set size, which proves that for the sequence labelling task, semi-supervised algorithms outperform supervised algorithm significantly as the dataset size decreases.

2 Related Work

This chapter provides a brief overview of the work that has already been performed in the field of de-identification of patient notes using different approaches.

2.1 Rule-Based Methods

Commonly used methods are rule-based methods that leverage look-up tables and regular expressions and heuristics in order to remove personally identifiable information. One example is the paper [Meystre et al., 2010]. The algorithm defined in this paper, uses perl to perform lexical matching with look-up tables, regular expression and simple heuristics to identify and remove PHI. For PHIs that include numeric patterns, such as dates, telephone/fax number and street address are matched using the regular expression as well as using heuristics that involves matching with the contextual keywords. For example street address may contain the keyword "road". For PHIs that doesn't contain numeric information such as names and locations, look-up tables are used to match the text as well as some heuristics are used. For example, in case of name, look-up tables and heuristics are used to identify the potential PHI. Look-up step involves matching the token with dictionary of names, whereas the heuristic step includes checking of contextual keywords such as "Mr.", "Dr.", "name is", etc. Other examples of rule-based systems are [Berman, 2003]. These methods are easy to develop and interpret and don't require a labeled dataset. These methods work reasonably well, especially if one has some prior knowledge on the domain the document belongs to. However, a significant drawback of these techniques is that they are based on manually written rules, which cannot deal with unexpected or rare cases, and come with high maintenance costs. These type of system are not flexible to language changes and don't take context into account, which could be really important for the de-identification task. This severely limits their applicability, especially to new (or previously unseen) domains.

2.2 Machine Learning based Methods

Problem faced using rule-based systems could be resolved using the machine learning systems for the de-identification of free-text. To alleviate some downsides of the rule-based systems, there have been many attempts to use supervised machine learning algorithms to de-identify text. These methods are more generalizable than the rule-based methods and can automatically learn complex patterns For de-identification we can make use of statistical named-entity recognition (NER) systems, which aim at identifying and classifying named entities in a document into predefined categories, e.g. names, credit card numbers, social security numbers, etc.

2.2.1 Feature-engineered supervised systems

These methods share two downsides: they require a decent-sized labeled dataset and lot of feature engineering. As with rules, quality features are challenging and time-consuming to develop. [Uzuner et al., 2008] developed a system that uses support vector machines and local context, whereas [Aberdeen et al., 2010] uses conditional random field to de-identify the medical discharge summaries.

2.2.2 Hybrid Systems

The object of the hybrid methods is to combine ruled-based system with machine learning system. Hybrid systems could achieve impressive results, as they aim to combine the advantages of rule-based and feature-engineered supervised systems. [Yang and Garibaldi, 2015] proposed a system that combines rule-based methods, dictionary look-ups and conditional random field (CRF). [Liu et al., 2015] proposed a hybrid system that combines rule-based approaches with machine learning method. Character-level features were also added along with word-level features to avoid boundary errors caused by token-level CRF. Hybrid systems are strongly dependent on feature engineering, a process that could be time consuming. Features extracted might be very specific to certain domain and might not work if domain is changed.

2.2.3 Neural network systems

Recent approaches to natural language processing (NLP) based on neural networks do not require handcrafted rules or features. They use neural network based system that learns internal representations on the basis of vast amounts of unlabeled training data. [Collobert et al., 2011] was the first work to demonstrate the usefulness of pre-trained word embeddings. They suggested an architecture of the neural network that forms the basis of many present methods. Their proposed approach achieved state-of-the-art performance for variety of NLP task including NER. [Huang et al., 2015] proposed using long short-term memory (LSTM, a variant of recurrent neural network) instead of convolutional neural network (CNN) for generating the word representation. Including LSTM improved the performance on the NER task. [Ma and Hovy, 2016] used CNN for creating the character-level representation. This type of architecture turned out be extremely effective, as it helped with spelling mistakes and out-of-vocabulary words. [Lample et al., 2016] proposed a model where they used LSTMs for creating the character-level representation. [Dernoncourt et al., 2017] have successfully applied neural network to the task of de-identification. This paper proposes two models: artificial neural network (ANN) model and CRF model. The main component of the ANN model are Bidirectional LSTM (Bi-LSTM) models, this model uses architecture similar to [Lample et al., 2016]. For CRF model, features are extracted for each token. They used a combination of n-gram, morphological, orthographic, and gazetteer features. The best performing model combines the outputs of the CRF model and ANN model in a way that token that is either identified by ANN or CRF model is considered a PHI. [Liu et al., 2017] proposed hybrid system for the de-identification task on the training set. This system is based on Bi-LSTM, Bi-LSTM with features, conditional random field (CRF) and a rule-based subsystem. [Khin et al., 2018] uses architecture similar to [Lample et al., 2016] but combines character-level LSTM embeddings, word embeddings, embeddings from Language Models (ELMo) embeddings, part-of-speech one-hot-encoded vector and the casing embedded vector to produce a vector that could be used as an input to neural network model.

3 Background

Recent approaches involved training of model on unlabeled data for the task of language modelling then these context specific representations are used as input to task specific models

3.1 De-Identification Task

3.1.1 The 2014 i2b2/UTHealth Dataset

The 2014 i2b2/UTHealth natural language processing shared task featured a track focused on the de-identification of longitudinal medical records. This track contains 1,304 longitudinal medical records describing 296 patients. This dataset is de-identified using the guidelines provided by HIPAA. HIPAA is a U.S. law providing data privacy and safety provisions to protect medical information. HIPAA relates to patient-identifying data as PHI and defines 18 PHI categories that could be linked directly to patient or anyone related to patient. Medical records in this track were de-identified using the following two steps. First, all the PHIs in the medical record are manually annotated and then these PHIs were replaced with pseudonyms. 18 HIPAA categories are divided into 6 primary categories and 25 sub categories, which are as follows.

3.1.2 MIMIC-III

MIMIC-III ('Medical Information Mart for Intensive Care') is a large, single-center database comprising information relating to patients admitted to critical care units at a large tertiary care hospital [Johnson et al., 2016, Goldberger et al., 2000, Saeed et al., 2011]. In this thesis, we are only using the discharge summaries instead of the whole dataset, which comprises of 59652 notes. Table 2 shows the distribution of PHIs.

PHI category	Sub-category	HIPAA	i2b2	MIMIC
Name	Names of patients and family members	✓	✓	✓
	Provider name		✓	✓
Profession	Profession		✓	
Age	Ages ≥ 90	✓	✓	✓
	Ages < 90		✓	
Location	Hospital	✓	✓	✓
	Organization	✓	✓	✓
	Street	✓	✓	✓
	City	✓	✓	✓
	State		✓	✓
	Country		✓	✓
	Employers	✓	✓	✓
	Hospital name		✓	✓
	Ward name			✓
Date	Date	✓	✓	✓
	Year		✓	✓
	Holidays		✓	✓
	Day of the week		✓	
Contact	Phone	✓	✓	✓
	Fax	✓	✓	✓
	Email	✓	✓	✓
	URL & IP Address			
IDs	Social Security Number	✓	✓	✓
	Medical Record Number	✓	✓	✓
	Account Number	✓	✓	✓
	Certificate or license numbers	✓	✓	✓
	Vehicle or device ID	✓	✓	✓
	Biometric ID			

Table 1: PHI types as defined by HIPAA, i2b2, and MIMIC. PHI categories are defined in the i2b2 dataset, redrawn after [Dernoncourt et al., 2017]

Category	Notes	Tokens	PHI Instances
Discharge summary	59652	80986971	2632527

Table 2: Distribution of PHIs in MIMIC-III

3.1.3 1 Billion Word Language Model Benchmark

1 Billion Word Language Model Benchmark [Chelba et al., 2013] dataset is standard dataset for the task of language modelling consisting of 1 billion words. This dataset is based on WMT 2011 News Crawl data, which is based on text crawled from online news.

3.2 Evaluation

The main metrics used for evaluation of models in our de-identification task are precision, recall and F1.

$$Precision = \frac{Number\ of\ tokens\ correctly\ identified\ as\ PHI\ tokens}{Number\ of\ tokens\ identified\ as\ PHI\ tokens} \quad (1)$$

$$Recall = \frac{Number\ of\ tokens\ correctly\ identified\ as\ PHI\ tokens}{Total\ number\ of\ PHI\ tokens} \quad (2)$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (3)$$

F1 score is the harmonic mean between precision and recall and is used as evaluation metric for many architectures.

We use the official evaluation script [Kotfic, 2014] to evaluate our models.

3.3 Word Representation

For using text data in machine learning, the data has to be transformed into a real-valued vector representation. Feature engineering is one of the way to convert words into vectors. One way of converting text into vectors is by using one-hot vectors in which all entries are 0 except the one corresponding to the respective word, which is 1. This one-hot vector will have a size equal to the size of vocabulary. Consider the following sentence: "He likes football" and "He likes soccer". If we

Word	One-hot encoding
He	[1, 0, 0, 0]
likes	[0, 1, 0, 0]
football	[0, 0, 1, 0]
soccer	[0, 0, 0, 1]

create a set of vocabulary it would look like this $V = \{He, likes, football, soccer\}$. If we visualize these representations in 4 dimensional space, all of the vectors would be equidistant from each other. So in our example "football" and "soccer" are as different as "He" and "likes", which is wrong. One-hot representation has the following disadvantage: they capture no semantic relationship between words, data sparsity is an issue as number of dimensions increase.

3.4 Word embedding

In NLP, mostly used word representation strategy is word embeddings, ([Mikolov et al., 2013], [Pennington et al., 2014]). These embeddings are dense vectors of fixed size, which are learned through the shallow neural networks. These embeddings capture the semantic information about the word, so semantically similar words lie close to each other in the lower dimension latent space. Problem with these approaches are that the word embeddings generated by these approaches are static and always generate the same embedding for a specific word regardless of the context in which word occurs. For example the word 'bank' could refer to financial institution or 'bank' of the river depending upon the context but the word 'bank' would have one embedding.

3.4.1 Word2vec

Word2Vec are word embeddings learned through shallow neural networks. In Word2Vec ([Mikolov et al., 2013] approach, a simple neural network with a single hidden layer to learn word vectors. Model is trained on unlabeled data. After the training is completed output layer is removed. Hidden layer weight learned through model training is used as word embeddings. Two basic neural network models for training of Word2Vec model are:

- Continuous Bag of Word (CBOW)
- Skip-gram (SG)

In continuous bag of word (CBOW) model, given a context word model trys to predict the target word. Whereas, in skip-gram (SG), model predicts context words given a target word.

3.4.2 GloVe: Global Vectors for Word Representation

For generating the glove embeddings word-context matrix is created, which encapsulates the co-occurrence information. The co-occurrence information is obtained by count how frequently a word occurs in some context. This matrix is then factorized to obtain a low dimensional representation. For this thesis, we are using pre-trained glove embeddings trained on Wikipedia 2014 and Gigaword corpus consisting of 6B tokens.

3.5 Contextual Embeddings

Language is very complex. Same word could get different meaning depending on the context it is used in. Contextual embeddings aim to capture word semantics in different contexts. Contextual models ideally capture complex characteristics of word use and how they differ across language contexts. This property of contextual embeddings make them extremely effective for the de-identification task.

3.5.1 ELMo: Deep contextualized word representations

ELMo is a latest technique to generate contextual embeddings. ELMo produces embedding as a function of a whole sentence instead of a single word. ELMo uses multiple stacks of Bi-LSTM to train a language model. Then for the input sequence of words hidden state of each layer is extracted. Final contextual embeddings are obtained using weighted sum of those hidden states for each word.

3.5.2 Flair: Contextual String Embeddings for Sequence Labelling

Like ELMo, language modelling is used to extract embeddings. Difference between ELMo and Flair is that Flair is a character-based, whereas ELMo is word-level

language model. In Flair, characters are used to predict the next characters in the model

3.6 Tagging Scheme

In sequence tagging task, our aim is to assign correct tag to each token. Many entities consist of multiple words like San Francisco, due to which there is a need of tagging scheme. Tagging scheme could remove ambiguities regarding the start and end of an entity. They can also improve the performance of the model by providing effective representation. IOB proposed in [Ramshaw and Marcus, 1999] is a common tagging scheme for tagging tokens in named entity recognition task. [Ratinov and Roth, 2009] and [Krishnan and Ganapathy, 2005] showed that more sophisticated tagging techniques like BIOES/BILOU tend to perform better that IOB tagging scheme in NER task.

IOB IOB stands for Beginning, Inside and Outside of a sequence. There are two IOB tagging schemes. In IOB, I- prefix indicates that the tag is inside the entity, and B- prefix indicates that the tag is at the beginning of the entity, if it is followed by the tag of the same type without O tag in between. O indicates that the token belongs to no entity. In IOB2, tagging is the same as IOB, except the beginning of every entity is indicated by B- prefix.

Smith	is	going	to	San	Francisco
I-PER	O	O	O	B-LOC	I-LOC

Table 3: IOB tagging scheme

Smith	is	going	to	San	Francisco
B-PER	O	O	O	B-LOC	I-LOC

Table 4: IOB2 tagging scheme

IOBES In this tagging scheme, S- prefix is used to represent an entity containing single token. If an entity is more that one token, then it starts with S- prefix and ends with E- prefix.

Smith	is	going	to	San	Francisco	to	meet	George	A.	Miller
S-PER	O	O	O	B-LOC	I-LOC	O	O	B-PER	I-PER	I-PER

Table 5: IOBES tagging scheme

3.7 Feedforward Models

Feedforward models are a composition of interconnected layers in which the output of each layer is passed to the next following layer. These models are called feedforward because the information is only propagated forward and never backward. The input is passed to the input layer which is then connected to hidden layers (single or multiple) and finally the output is provided through the output layer.

Feedforward networks have fully connected layers and activation layers. Fully connected layers multiply their input with weights matrix and add a bias vector.

$$y = W^T x + b \tag{4}$$

Size of output of each layer depends on the shape of weight matrix. The parameters of a layer are the weight matrix and the bias vector.

Feedforward models also have an activation layer which applies non-linearity ϕ

$$y = \phi(x) \tag{5}$$

The rectified linear unit (ReLU) [Glorot et al., 2011] is the most popular activation function for activation layers. RelU discards any value which is less than zero.

$$ReLU(x) = max(0, x) \tag{6}$$

Feedforward neural networks are needed because models, like perceptron, are unable to handle data which is not linearly separable. Feedforward networks have fully connected hidden layers between input and output layers which are able to handle complex non-linearly separable relations in data.

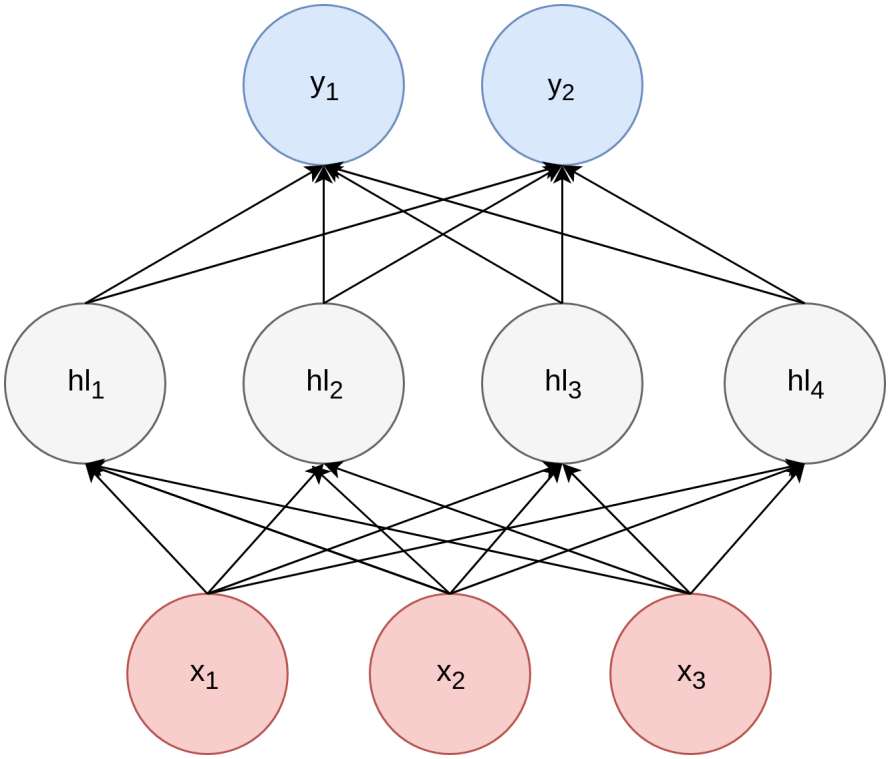

Figure 1: A feedforward neural network with one hidden fully connected layer

3.8 Recurrent Models

Recurrent Neural Networks (RNNs) comprise a type of neural network architectures that are used to model, and predict using sequential data. Vanilla architectures of neural nets with fully connected, and convolutional layers are faced with the limitation that they can only take in fixed-length inputs, a property very disadvantageous for processing sequences of variable length. They also do not store context or information about past data in the sequence to reason about the current prediction. The reason being the assumption that all input data are unrelated and independent of each other. These two limitations are overcome by the RNN by looping its past internal state back into itself. This 'memory' of the past allows it to reason about current inputs using past context in the sequence of inputs.

A simple RNN cell consists of 3 components. An input layer, a hidden state, and

an output layer. The input layer is the current input data which enters the hidden state along with the previous hidden state to get the updated hidden state. The hidden state operation is described by the following function:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{7}$$

where: x_t is the current input, h_t and h_{t-1} the current and previous hidden states, W_{hh} the weight matrix for the previous hidden state, W_{xh} the weight matrix for the input.

The output cell (or layer) receives the hidden state, and gives output predictions as described by the following function:

$$y_t = Softmax\ (W_{ho}h_t) \tag{8}$$

where y represents the output at time t.

Both of these operations constitute an RNN 'step' at time t.

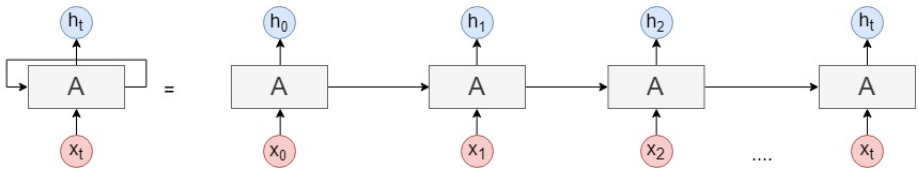

Figure 2: Recurrent Neural Network

3.8.1 Long Short-Term Memory

While recurrent neural nets are amazingly effective at modelling sequences, they can only use context from the very recent inputs. The limitation comes from the 'Vanishing gradient problem' which many-layered neural networks suffer from. RNNs, in their iterative operation, also behave like multi-layered networks in which the gradients at earlier layers tend to shrink to zero, slowing down the training of parameters to a complete halt. This shrinking of the partial derivatives to zero is called the vanishing gradient problem and affects deep networks. It means that RNNs can not make use of memory beyond the recent past. This limitation is resolved by LSTMs.

The key idea in LSTMs is that the hidden-state is replaced by the cell-state. Information can be added or removed from the cell state using gates. The previous cell state enters the new cell state as it is, i.e without being multiplied by weighting parameters, thus avoiding the vanishing gradient problem. As a result, the LSTM can make use of long term memory, using dependencies from way back in time, and allowing it to contextualise and understand the whole sequence better.

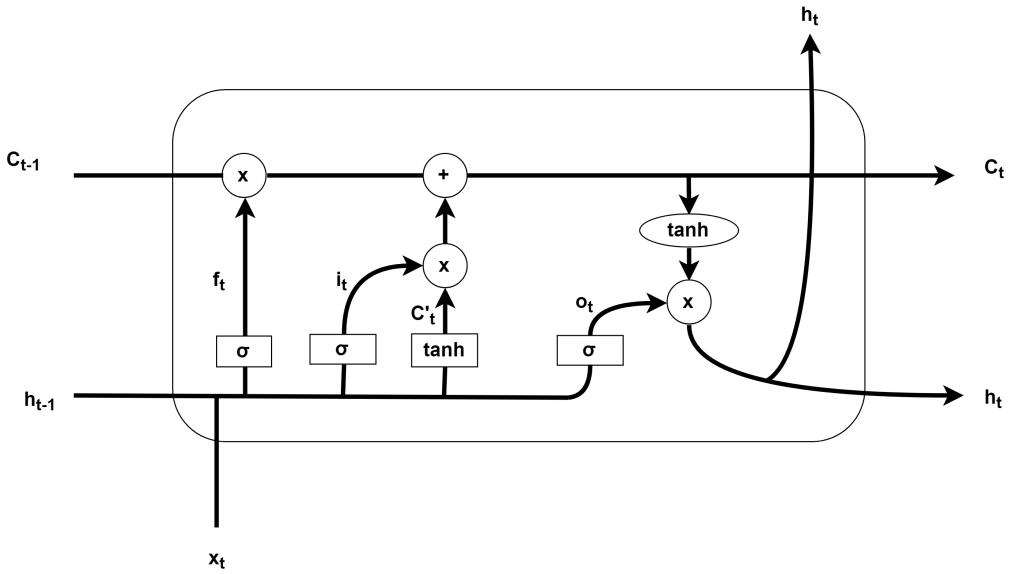

Figure 3: Long short-term memory (LSTM)

The LSTM unit contains a cell state and 3 gates to regulate the flow of information. The first is a forget gate f_t, consisting of a sigmoid layer, which decides which information to remove from the cell state. The second is the input-gate layer i_t, which decides which information to add to the cell state. It's output of one's and zero's masks the input vector of the candidate values C_t leaving only those multiplied by one's to be added to the cell state. The third gate o_t decides which information to output. It creates another masking vector to be multiplied by the cell state vector passed through a tanh layer. The remaining contents are then sent to the cell output. The current cell state, and output are also fed back to the LSTM cell for the next time step.

3.9 Convolutional Models

Convolutional Neural Networks [Krizhevsky et al., 2012] are a special type of Neural Networks that make the assumption of convolutional kernels on the data,

which is a reasonable assumption when working with images. Convolution operations apply a filter to the image. CNNs consist of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers.

Convolutional Layers

The convolutional layer is the core building block of a CNN. It's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects a specific pattern at some spatial position in the input. Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer.

Pooling Layers

The function of pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, to make it trainable with reasonable resources and to avoid overfitting. Pooling layer operates on each feature map independently.

3.10 Neural Network for Sequence Labeling

Most of the neural models used for sequence labelling have three main components, which are as follows.

1. Character representation layer
2. Word representation layer
3. Tag decoding layer

4 shows the neural architecture used in most of the sequence labeling task. In character representation, convolutional neural networks (CNNs) or bi-directional LSTM (Bi-LSTM) is used to encode character-level information of a word into its character-level representation. Then in word representation layer, character-level and word-level representations are combined and fed into convolutional neural networks (CNN) or bi-directional LSTM (Bi-LSTM) to model context information of each word. Finally, in tag decoding layer conditional random field (CRF) or softmax is used to jointly decode labels for the whole sentence.

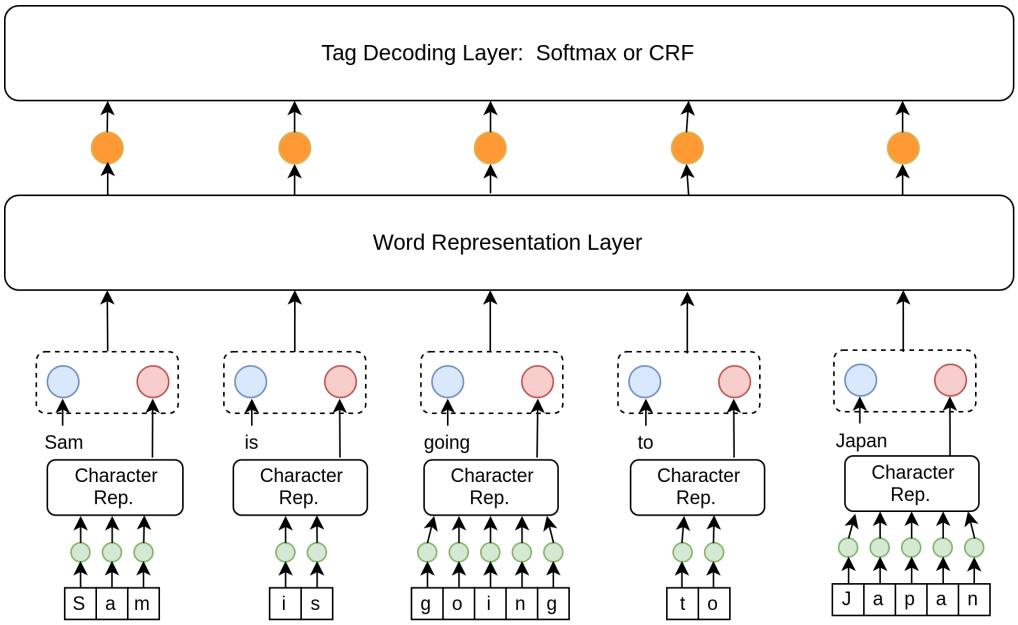

Figure 4: Neural Architecture for Sequence Labeling

3.11 Character representation layer

Word could be represented as a sequence of characters. Character representation could be extremely useful, as it could help with typographical error. Character information such as capitalization, prefix and suffix could be encoded using feature based look-up tables. These look-up tables were used in [Collobert et al., 2011] and [Huang et al., 2015]. Neural network could be used to encode the character-level information, without having the need of hand-curated features. [Santos and Zadrozny, 2014] introduced the notion of creating neural character embedding to boost the performance on part-of-speech tagging task. All of the

state-of-the-art methods in named entity recognition as well as de-identification task use character-level representation layer.

CNN Character Representation Layer [Santos and Guimaraes, 2015] used CNN architecture to create character representation, which improved the performance on named entity recognition task. [Chiu and Nichols, 2016] and [Ma and Hovy, 2016] used CNN architecture to extract character representation. In their architecture CNN was followed by followed by dropout and max pooling layer.

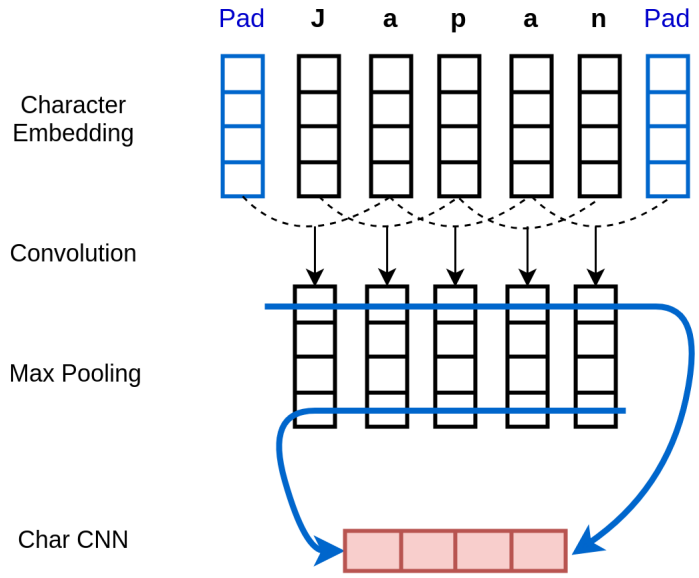

Figure 5: Character-level Representation Layer CNN

LSTM Character Representation Layer Bidirectional LSTM could be used to generate character representation. Representations produced using forward LSTM (F_{LSTM}) and backward LSTM (B_{LSTM}) are concatenated to get the final representation. [Lample et al., 2016] used this representation model to generate representation for individual words for the task of named entity recognition. [Liu et al., 2018] used character representation layer by treating sequence as characters and applying bidirectional LSTM on top of it to obtain character-level representation.

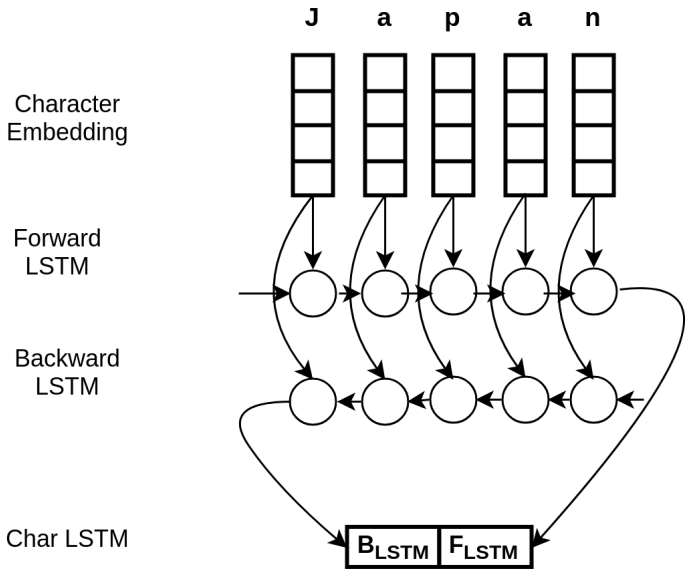

Figure 6: Character-level Representation Layer LSTM

3.12 Word representation layer

To encode word-level representation convolutional neural network (CNN) or bidirectional LSTM (Bi-LSTM) could be used. If character representation layer is used then character representations are concatentated with word embeddings and pass through convolutional neural network (CNN) or bidirectional LSTM (Bi-LSTM) to produce word representations.

CNN Word Representation Layer CNN could be used as a word representation layer. In CNN, a filter of fixed size is passed through the input to extract features from the word input, which is mostly followed by RELU function. Usually batch normalization and dropout is used in case of CNNs. Advantages of CNN include parallelism and stable gradient. [Strubell et al., 2017] used modified form of convolution known as dilated convolution as a word representation layer. Dilated convolution tend to have higher receptive field, which means number of words visible to each filter at a time. This turns out to be very useful in sequence labelling

LSTM Word Representation Layer Bi-LSTM is mostly used as word representation layer in most of the sequence labelling tasks. Usually, character representa-

tions are concatentated with word embeddings and fed into forward and backward LSTM. The representation extracted through forward and backward LSTM is concatenated to obtain the final representation.

Tag decoding Layer The tag decoding layer takes as an input word representation and assign tag to each token in the sequence. Softmax or CRF layer could be used for this task. However, CRF layer seems to be more effective as in the task of sequence labelling there is a strong dependency between labels. CRF is used in almost all of the best performing models for the task of named entity recognition.

4 Approach

4.1 Cross View Training (CVT layer)

Deep learning models work well when trained on large amount of data. In most of the real world problem, task specific labeled data is not available. Mostly used unsupervised techniques include models that learn the word representation. One of the disadvantages is that while learning the representation, labeled data is not utilized and these representations are learned without utilizing the labeled data. Due to this, there is a need of semi-supervised learning algorithms that effectively use the data.

CVT [Clark et al., 2018] is an effective training mechanism to make use of labeled and unlabeled data for training the model. This model utilizes two key concepts of self training and multi-view learning.

4.1.1 Background

Self-training is a process that makes use of both labeled and unlabeled data for training model. In self-training, model is trained as normal on labeled data, whereas in case of unlabeled data, this model performs a role of a teacher, while making the predictions and a student that is trained on those predictions. Self-training seems a bit circular. For example in the following sentence "He is an engineer" where we do not have human provided annotation what we can do is to perform inference from our model to produce prediction such as engineer is a profession and use this pseudo-label to train our model. Here we are training the model that engineer is a profession, this is what our model already knew because our label profession came from the model itself.

Co-training: [Blum and Mitchell, 1998] came up with a solution to this problem called co-training. In this approach instead of one model teaching itself, we are going to have two models teach each other. Each model is going to see different view of the input sentence.

In 7 if our example is ‘He worked in design‘. Model 1 is only going to make prediction looking at ‘He worked in‘ and model 2 is going to make prediction only seeing ‘design‘. These different views are useful because they provide a way for the models to teach each other. In our case model 1 seeing ‘He worked in‘ can probably guess that profession is coming next, whereas model 2 might struggle on "design" cause its a generic work and could be used in any other context than profession. In this case model 1 should be able to teach model 2 something on the unlabeled example.

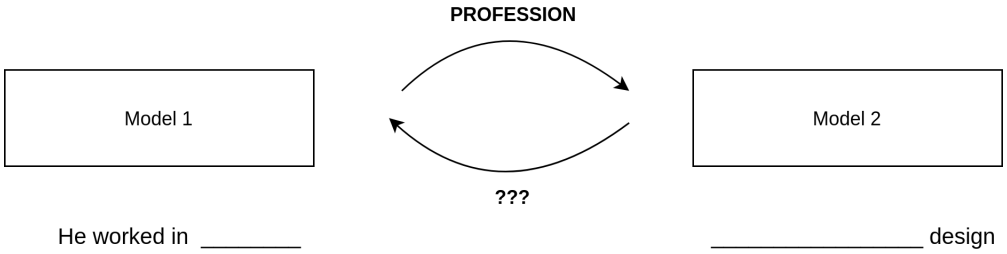

Figure 7: Co-training: Example 1

In 8, ‘He is an engineer‘ model 2 might be able to teach model 1 something. Here model 1 might get confused over kind of generic phrase but ‘engineer‘ is quite a common profession. Here we avoided the issue of model teaching itself only things it knew in the first place.

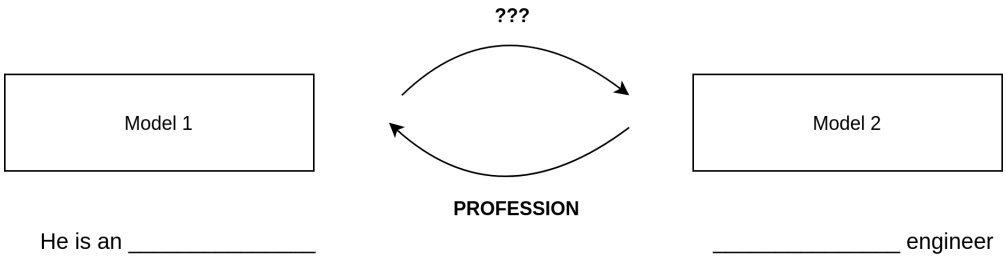

Figure 8: Co-training: Example 2

4.1.2 Approach

In CVT, as the name suggests, like in co-training we are going to be taking advantage of different view of the input to improve how self-training works. Two models are sharing their knowledge by passing the predictions around to each other on unlabeled examples. In neural models it is easy to share knowledge by sharing the parameters, this will improve the knowledge sharing between two models because now if model 1 learns something and that knowledge gets reflected in weights model 2 could automatically have access to that information 9.

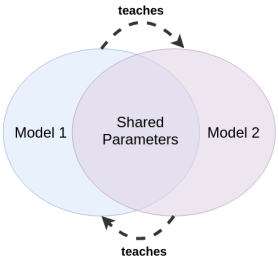

Figure 9: Neural Model can share parameters

In multi-view learning, model is trained to produce consistent predictions looking at the different subset of input. CVT model consists of primary module and auxiliary modules.

Labeled data: For the case of labeled data, primary module is trained like in the case of named entity recognition task. For example input sequence "He is an engineer" is passed through Bi-LSTM encoder and on top of it primary prediction module which is a softmax layer is applied to get prediction for the sequence. This prediction is compared against the ground truth available in the labeled data to calculate the loss.

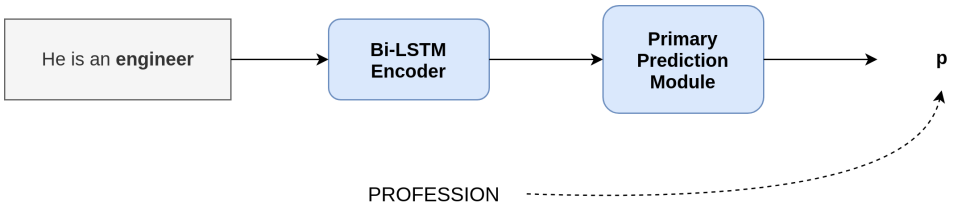

Figure 10: CVT: Labeled data

Unlabeled data: For the case of unlabeled data, primary module acts as a teacher, where predictions are made using all of the input. Auxiliary modules then makes prediction, using the limited view of the input. Primary module acts as a teacher for all the auxiliary modules because it has unrestricted view of the input and auxiliary modules act as the students, who learn from the primary module. This improves the contextual representation produced by the model.

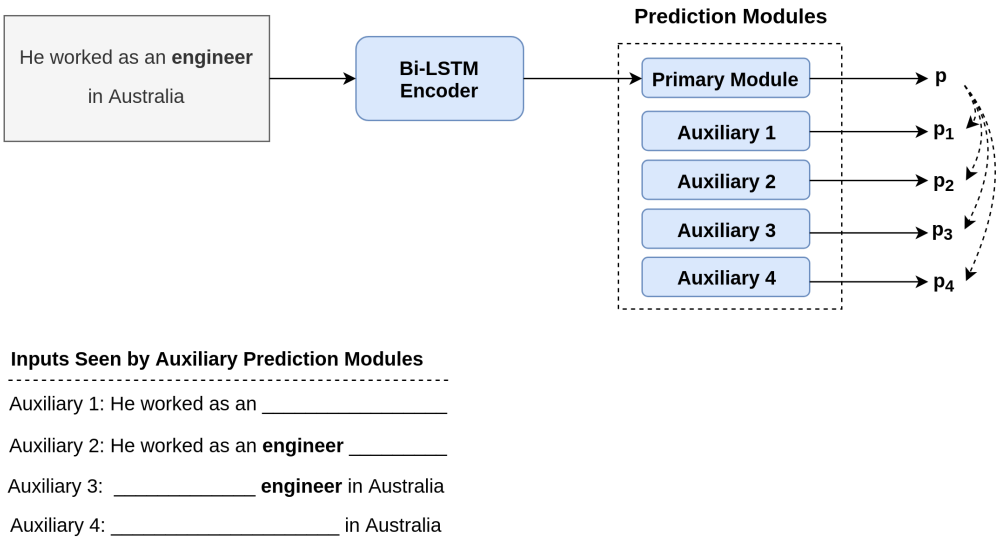

Figure 11: CVT: Unlabeled data

Primary Prediction Module Model will make predictions for each word in the sentence taking in the representation produced by the Bi-LSTM encoder going in different directions and pass this representation through primary module which is a softmax layer to predict its named entity type.

Auxiliary Module 1 (Future Predictor) First auxiliary predictor will also be softmax layer that makes a prediction about the current word but now instead of using both LSTMs representations to make this prediction from the forward or left to right going LSTM, we can take advantage of the structure of the model to create this predictor to now only sees the part of the input sentence like in co-training. Like in example, model only sees "He worked as an" while predicting the label for the word "engineer" and not the right context of the current word including the word itself.

Auxiliary Module 2 (Forward Predictor) This auxiliary predictor is also softmax layer that makes prediction about the current word but now it uses forward LSTM. Like in example, model only sees "He worked as an engineer" while predicting the label for the word "engineer" and not the right context of the current word.

Auxiliary Module 3 (Backward Predictor): Similarly, we can do the same with the backward LSTM to get backward predictor that makes prediction about the current word by only seeing the right context of word. Like in example, model only sees "engineer in Australia" while predicting the label for the word "engineer" and not the right context of the current word.

Auxiliary Module 4 (Past Predictor) Same is the case with backward predictor where model guesses the label for the word without taking into account the right context plus the word itself.

Now the primary predictor that sees the whole input is going to be teaching these auxiliary predictors. Auxiliary predictor can learn from the primary predictor because the primary predictor sees more of the input, so it would produce more accurate predictions and auxiliary predictors can benefit from these improved labels. So we have this nice cycle where primary model teaches the auxiliary models and the auxiliary models improve the primary model through this shared representation.

Model used for the CVT is the model defined in [Ma and Hovy, 2016]. Instead of one Bi-LSTM sentence encoder, two Bi-LSTM sentence encoders are used. The output of first Bi-LSTM sentence encoder is fed to the second Bi-LSTM sentence encoder. Model takes as an input $x_i = \left[x_i^1, x_i^2, \ldots, x_i^T\right]$ a sequence. The output of character representation layer which is basically a CNN is concatenated with embedding of word to get a representation $e = \left[e^1, e^2, \ldots, e^T\right]$. This representation is fed into first Bi-LSTM layer to get the forward $\left[\overrightarrow{a}_1^1, \overrightarrow{a}_1^2, \ldots \overrightarrow{a}_1^T\right]$ and backward $\left[\overleftarrow{a}_1^1, \overleftarrow{a}_1^2, \ldots \overleftarrow{a}_1^T\right]$ representation. These representations are concatenated $h_1 = \left[\overrightarrow{a}_1^1 \oplus \overleftarrow{a}_1^1, ..., \overrightarrow{a}_1^T \oplus \overleftarrow{a}_1^T\right]$. The second Bi-LSTM takes as an input this concatenated information a_1 and produces the output a_2 in the same way as the first layer.

Given the labeled dataset represented as $\mathcal{D}_l = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$. Unlabeled dataset represented as $\mathcal{D}_{ul} = \{x_1, x_2, \ldots, x_M\}$. If θ defines the parameters of our model, then we could define the output distribution over classes

generated by our model given the input x_1 as $p_\theta\left(y|x_i\right)$. The derivations and equations below are cited and summarized from [Clark et al., 2018].

The primary prediction module uses a one-layer neural network. The output of the first and the second Bi-LSTM layer is concatenated and fed into one-layer neural network to produce a probability distribution over classes for the corresponding t^{th} label.

$$\begin{aligned} p\left(y^t|x_i\right) &= \mathrm{NN}\left(a_1^t \oplus a_2^t\right) \\ &= \mathrm{softmax}\left(\mathrm{ReLU}\left(W\left(a_1^t \oplus a_2^t\right)\right) + b\right) \end{aligned}$$

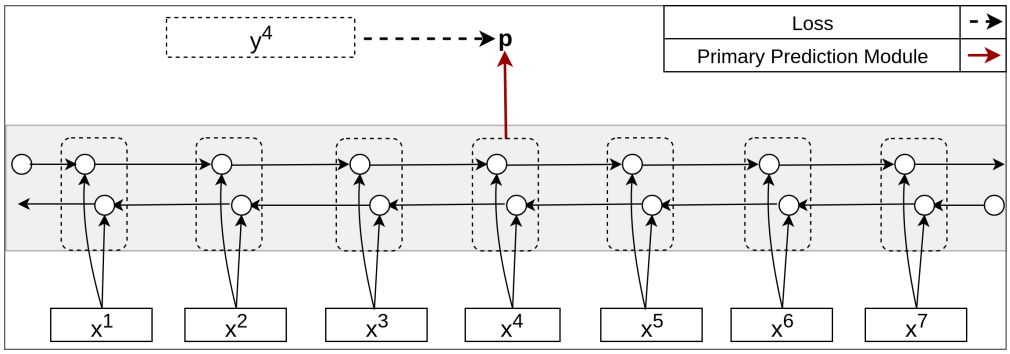

Figure 12: CVT: Labeled data detailed

(9)

The auxiliary prediction modules take as an input the representation generated from the forward and backward LSTM of the first Bi-LSTM encoder. The forward, backward, future and past predictors are defined as follows.

$$p_\theta^{\mathrm{fwd}}\left(y^t|x_i\right) = \mathrm{NN}^{\mathrm{fwd}}\left(\overrightarrow{a}_1^t\left(x_i\right)\right) \tag{10}$$

$$p_\theta^{\mathrm{bwd}}\left(y^t|x_i\right) = \mathrm{NN}^{\mathrm{bwd}}\left(\overleftarrow{a}_1^t\left(x_i\right)\right) \tag{11}$$

$$p_\theta^{\mathrm{future}}\left(y^t|x_i\right) = \mathrm{NN}^{\mathrm{future}}\left(\overrightarrow{a}_1^{t-1}\left(x_i\right)\right) \tag{12}$$

$$p_\theta^{\mathrm{past}}\left(y^t|x_i\right) = \mathrm{NN}^{\mathrm{past}}\left(\overleftarrow{a}_1^{t+1}\left(x_i\right)\right) \tag{13}$$

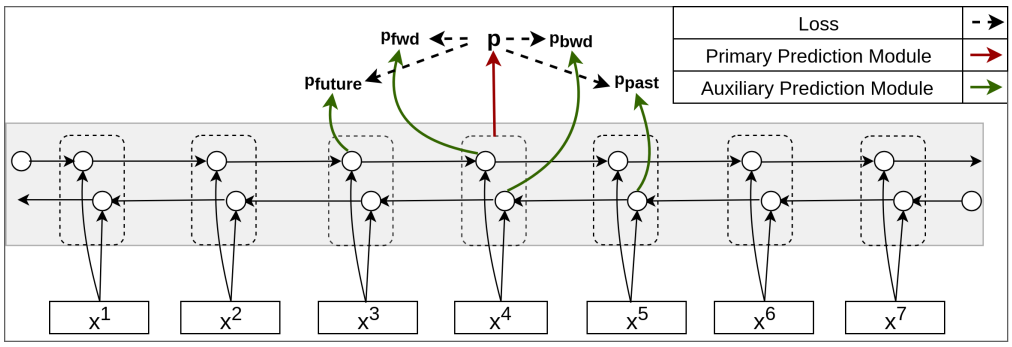

Figure 13: CVT: Unlabeled data detailed

Training a CVT model requires us to alternate between labeled and unlabeled examples. For the case of labeled data, cross entropy loss is used.

$$\mathcal{L}_{\text{sup}}(\theta) = \frac{1}{|\mathcal{D}_l|} \sum_{x_i, y_i \in \mathcal{D}_l} CE\left(y_i, p_\theta\left(y|x_i\right)\right) \tag{14}$$

For the case of unlabeled data, fixed number of auxiliary modules are used. Each auxiliary module uses the intermediate representation $a^j\left(x_i\right)$, where it sees the limited part of the input and produces the distribution over classes $p_\theta^j\left(y|x_i\right)$. The predictions produced by the primary module $p_\theta\left(y|x_i\right)$ are matched with the prediction of the auxiliary modules. The model is learning by miminizing the following equation.

$$\mathcal{L}_{\text{CVT}}(\theta) = \frac{1}{|\mathcal{D}_{ul}|} \sum_{x_i \in \mathcal{D}_{ul}} \sum_{j=1}^{k} D\left(p_\theta\left(y|x_i\right), p_\theta^j\left(y|x_i\right)\right) \tag{15}$$

Here D defines the distance between probability distributions, which is KL divergence in this case. Total loss $\mathcal{L} = \mathcal{L}_{\text{ sup}} + \mathcal{L}_{\text{ CVT}}$ is minimized by alternating between the mini-batch of labeled and unlabeled examples.

4.2 Flair Embedding

Flair embeddings as proposed in [Akbik et al., 2018] are a contextual embedding obtained by character level language modelling Bi-LSTM is used for the task of character-level language modelling. Each sentence is treated as sequence of characters. At each time-step within a sequence, model is trained to predicts the next character in the sequence. Each hidden state corresponds to a character

within a sequence. Character-level language model tends to predict a distribution over the sequence of characters, which encapsulates properties related to natural language. The joint distribution over the entire sequence could be calculated by taking the product of the distribution of characters given the previous characters.

$$P\left(\mathbf{x}_{0:T}\right) = \prod_{t=0}^{T} P\left(\mathbf{x}_t | \mathbf{x}_{0:t-1}\right) \tag{16}$$

In LSTM model as the hidden state at time-step t contains history of all the characters before the time-step t, therefore the conditional probability could be approximated as a function of the network output h_t.

$$P\left(\mathbf{x}_t | \mathbf{x}_{0:t-1}\right) \approx \prod_{t=0}^{T} P\left(\mathbf{x}_t | \mathbf{h}_t; \theta\right) \tag{17}$$

In LSTM, h_t could be calculated for each time-step using the additional quantity c_t as follows.

$$\mathbf{h}_t\left(\mathbf{x}_{0:t-1}\right) = f_{\mathbf{h}}\left(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}; \theta\right) \tag{18}$$

$$\mathbf{c}_t\left(\mathbf{x}_{0:t-1}\right) = f_{\mathbf{c}}\left(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}; \theta\right) \tag{19}$$

The fully connected softmax layer is added on top of h_t to get likelihood of every character.

$$\begin{aligned} P\left(\mathbf{x}_t | \mathbf{h}_t; \mathbf{V}\right) &= \operatorname{softmax}\left(\mathbf{V}\mathbf{h}_t + \mathbf{b}\right) \\ &= \frac{\exp\left(\mathbf{V}\mathbf{h}_t + \mathbf{b}\right)}{\left\|\exp\left(\mathbf{V}\mathbf{h}_t + \mathbf{b}\right)\right\|_1} \end{aligned} \tag{20}$$

Where V and b, weights and biases, are part of the model parameters θ.

Also backward recurrent neural network is used to extract contextualized embeddings, which works the same way as forward, but in the opposite direction.

$$P^b\left(\mathbf{x}_t | \mathbf{x}_{t+1:T}\right) \approx \prod_{t=0}^{T} P^b\left(\mathbf{x}_t | \mathbf{h}_t^b, \theta\right) \tag{21}$$

$$\mathbf{h}_t^b = f_{\mathbf{h}}^b\left(\mathbf{x}_{t+1}, \mathbf{h}_{t+1}^b, \mathbf{c}_{t+1}^b; \theta\right) \tag{22}$$

$$\mathbf{c}_t^b = f_{\mathbf{c}}^b\left(\mathbf{x}_{t+1}, \mathbf{h}_{t+1}^b, \mathbf{c}_{t+1}^b; \theta\right) \tag{23}$$

To extract the contextualized word embedding, both forward and backward recurrent neural network is utilized. In the case of forward network, hidden state after the last character in the word is extracted, which encapsulates the semantic and syntactic information from the start of the sequence until the word. Similarly, in the case of backward network, hidden state before the first character in the word is extracted, which entails the semantic and syntactic information of the sequence from the word to the end of sequence. Both these embeddings are combined to generate the final embedding of the word, which contains the information about the word itself as well as the surrounding context. Where we want to learn to find semantic relationship in the words

For all the words in the sequence w_0, w_1, ... w_n, embedding for each word could be defined as follows.

$$\mathbf{w}_i^{CharLM} := \begin{bmatrix} \mathbf{h}^f_{t_{i+1}-1} \\ \mathbf{h}^b_{t_i-1} \end{bmatrix} \tag{24}$$

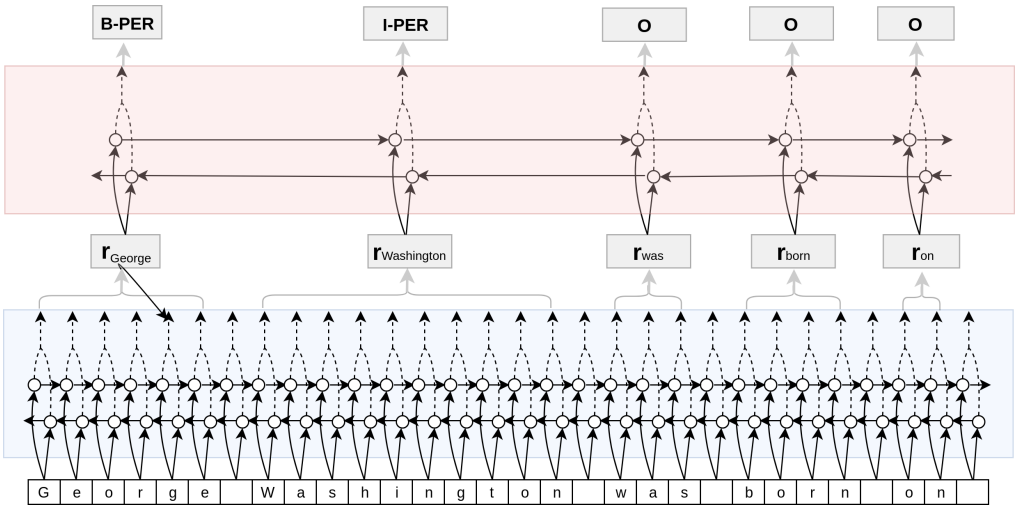

Figure 14: Flair diagram

4.2.1 Models

For this thesis, we are using two different models with different configurations, which are as follows.

- CVT(1b)
- CVT(MIMIC)
- Flair(1b)

- Flair(MIMIC)

Two of these models use discharge summaries from MIMIC-III dataset for unsupervised training, so we will first discuss the transformation of discharge summary from MIMIC-III dataset to the format that looks more like i2b2 dataset.

MIMIC-III conversion

Discharge summaries from the MIMIC-III dataset looks like the following example "[**Patient Name**] visited [**Hospital**]", where PHIs are identified and replaced with certain pattern. Whereas, the i2b2 dataset has the following format "John Doe visited SAH", where PHIs are identified and replaced with a surrogate. Therefore, there is a need of converting MIMIC-III dataset to the format that looks more like i2b2 format, which could be useful for unsupervised learning. Concretely converting "[**Patient Name**] visited [**Hospital**]" to "John Doe visited SAH". For this conversion various gazetteer(hospital names, first names, etc.) and regular expressions were used to replace the PHI with a surrogate.

CVT(1b)

We trained the CVT model using a mix of labeled and unlabeled data. For labeled data we using i2b2 2014 dataset, whereas for unlabeled data, we used 1 Billion Word Language Model Benchmark data. For the sequence labelling task architecture similar to [Ma and Hovy, 2016] is used, see further details in section 3.10. Only difference is that instead of CRF for tag decoding, we have 5 fully connected layers for primary and auxiliary modules. Figure 15 defines the component diagram. The hyperparameters configuration for the model are defined in table 6

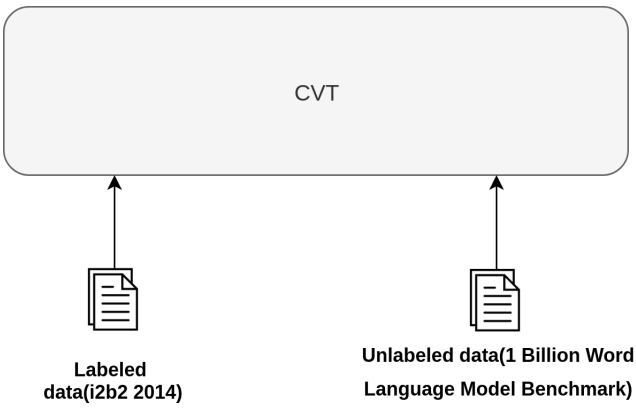

Figure 15: CVT(1b) Model

CVT(MIMIC)

We trained the CVT model using a mix of labeled and unlabeled data. For labeled data we using i2b2 2014 dataset, whereas for unlabeled data, we used discharge summaries from MIMIC-III dataset after its conversion to i2b2 format. Figure defines the model used 16. Architecture used is similar to the one used in CVT(1b) 4.2.1.

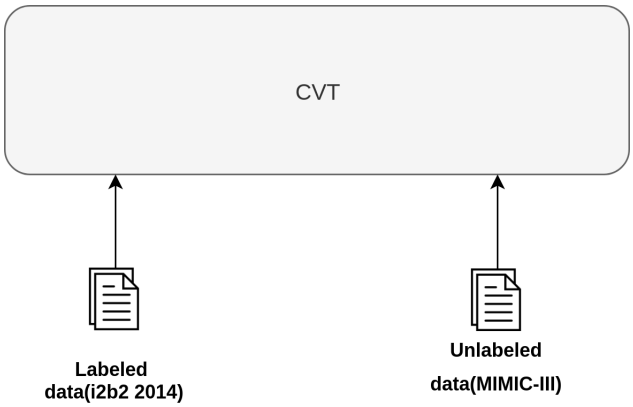

Figure 16:

Category	Hyperparameter	Details	Values
Inputs	Token embedding	Glove 6b	300
	Mini batch size		32
Architecture	Char embedding size		50
	Char CNN	filter width	[2, 3, 4]
	Char CNN	filter number	100
	Bi-LSTM	First Layer	1024
	Bi-LSTM	Second Layer	512
Training	Variational dropout		0.5
	dropout labeled		0.5
	dropout unlabeled		0.8
	Optimizer		SGD
	Learning rate		0.1
	Momemtum		0.9
	Weight decay		1e-5

Table 6: Hyperparameters for the CVT models

Flair(1b)

This approach could be divided into two steps. First, we trained our own Flair model on corpora that have about 1 billion words. Input sentence is fed as a sequence of characters into this pre-trained character language model. For each word contextual embedding is obtained. This embedding is then fed into BiLSTM-CRF model to extract the labels for each word. Figure 17 shows the component diagram.

Flair(MIMIC)

This approach uses discharge summaries from MIMIC-III dataset to train the Flair model. First, discharge summaries from MIMIC-III dataset are converted to i2b2 format. This transformed data is then used to train the Flair model for the task of character language model. Then, input sentence is fed as a sequence of characters into this pre-trained character language model. For each word contextual embedding is obtained. This embedding is then fed into BiLSTM-CRF model to extract the labels for each word. BiLSTM-CRF model for sequence labelling is explained in the section 3.10. Hyperparameters used to train character

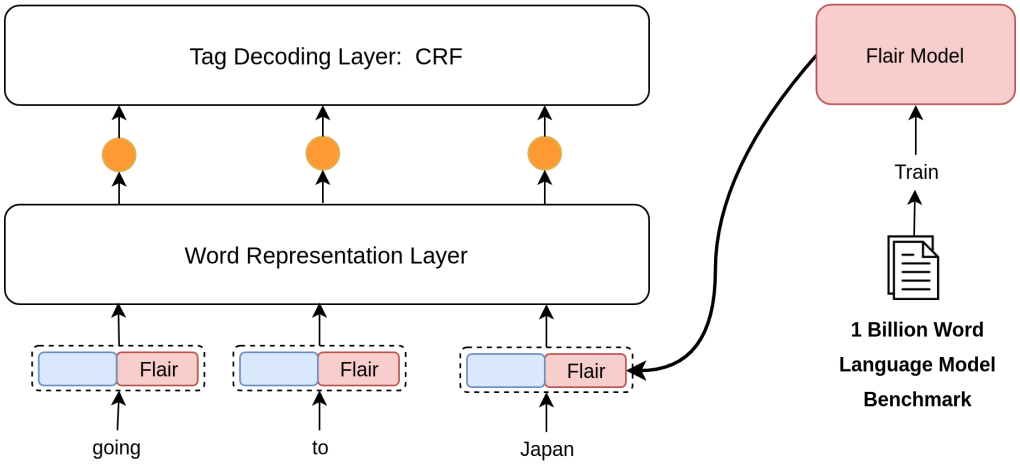

Figure 17: Flair(1b) Model

level language model to generate Flair embeddings are given in 7. Hyperparameters used by our model are defined in table 8. Figure 18 shows the component diagram.

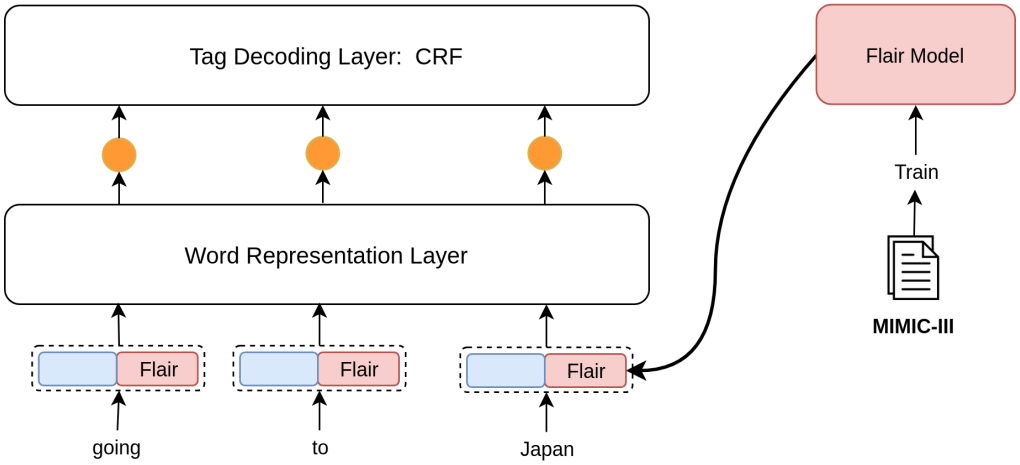

Figure 18: Flair(MIMIC) model

Hyperparameter	Details	Values
Bi-LSTM	One Layer	1024
Sequence length		10
Mini batch size		100
Max epoch		10

Table 7: Hyperparameters for the Flair language model

Category	Hyperparameter	Details	Values
Inputs	Token embedding	Glove 6b	300
	Flair embedding	1 billion or MIMIC-III	1024
	Mini batch size		32
Architecture	Bi-LSTM	One Layer	300
Training	Variational dropout		0.5
	Word dropout		0.05
	Optimizer		SGD
	Learning rate		0.1
	Momemtum		0.9
	Weight decay		1e-5

Table 8: Hyperparameters for the overall Flair models

5 Experiments

In this section, we provide an empirical evaluation of the models that we defined in the thesis. In summary, we have two models and different variations of those models. First set of experiments include unsupervised training on default dataset, which in our case is 1 billion word benchmark data [Chelba et al., 2013] based on news data. Whereas, the second set of experiments used the task specific data, which in our case is patient notes extracted from MIMIC-III dataset [Johnson et al., 2016] for unsupervised learning task to enhance the performance of the model.

5.1 De-Identification Baseline

We compare our results to current state-of-the-art model proposed in [Dernoncourt et al., 2017] as well as model proposed in [Ma and Hovy, 2016] and [Khin et al., 2018]. In these experiments, we want to produce results that could achieve performance similar to state-of-the-art without the use of hand curated features. Also, as semi-supervised techniques work well with small amount of data, we provide a comparison of models using training dataset of different sizes.

5.1.1 Baselines

[Ma and Hovy, 2016] is among the best performing models that uses CNNs to encode character-level information of a word into its character-level representation. Then the character-level and word-level representations are combined and fed into Bi-LSTM to model context information of each word. On top of this, sequential CRF is used to jointly decode labels for the whole sentence.
[Dernoncourt et al., 2017] is current state-of-the-art model for de-identification of the patient notes on the 2014 i2b2 de-identification Track 1 data set [Stubbs et al., 2015]. This paper proposes two models: ANN model and CRF model. The main component of the ANN model are Bi-LSTM models, this model uses

architecture similar to [Lample et al., 2016], where architecture is similar to [Ma and Hovy, 2016] but [Lample et al., 2016] uses Bi-LSTMs instead of CNN to produce character-enhanced representations of each unique token. For CRF model, features are extracted for each token. They used a combination of n-gram, morphological, orthographic, and gazetteer features. The best performing model combines the outputs of the CRF model and ANN model in a way that token that is either identified by ANN or CRF model is considered a PHI.
[Khin et al., 2018] uses architecture similar to [Lample et al., 2016] but combines character-level LSTM embeddings, word embeddings, ELMo embeddings, part-of-speech one-hot-encoded vector and the casing embedded vector to produce word-level representations. This vector is then fed into Bi-LSTM to model context information of each word. Then, sequential CRF is used to jointly decode labels for the whole sentence.

5.2 Results

The table 9 shows the HIPAA binary token-based de-identification scores on the i2b2 2014. For each model, we ran the experiments 3 times and averaged the results. Flair(1b) is the best performing model in terms of F1-Score. It outperforms all the other models with a score of 97.99%. However, for recall, CVT(MIMIC) has the highest score of 97.69%, followed by Flair(MIMIC) + BiLSTM-CRF with a score of 97.61% among the models that don't use hand-curated feature. [Dernoncourt et al., 2017] model that includes ANN + CRF performs best in terms of recall but it uses hand-curated features. Table 10 and figure 19 shows F1-score by category for each model, whereas 11 depicts recall score by category for each model. Our primary focus in de-identification task is on recall metric because having high recall means there is minimal leakage of personal information. Our experiments show that using in-domain unlabeled data improves the model, which allows it to achieve better results.

5.3 Error Analysis

CVT(MIMIC) outperforms all the other models in PROFESSION, AGE and DATE categories. Flair(1b) outperforms all the other models in NAME, CONTACT and ID categories. In the category of Location, Flair(MIMIC) shows a

Model	Precision	Recall	F1-Score
ANN + CRF [Dernoncourt et al., 2017]	97.92	**97.83**	97.87
Elmo + BiLSTM-CRF [Khin et al., 2018]	98.30	97.37	97.83
BiLSTM-CRF [Ma and Hovy, 2016]	98.03	97.20	97.61
CVT + 1b	97.96	97.27	97.62
CVT + MIMIC	98.22	97.69	97.95
Flair(1b) + BiLSTM-CRF	**98.46**	97.52	**97.99**
Flair(MIMIC) + BiLSTM-CRF	98.28	97.61	97.94

Table 9: F1-Score (%) on HIPAA-PHI categories on 2014 i2b2/UTHealth shared task Track 1. Best performing according to each metric is highlighted.

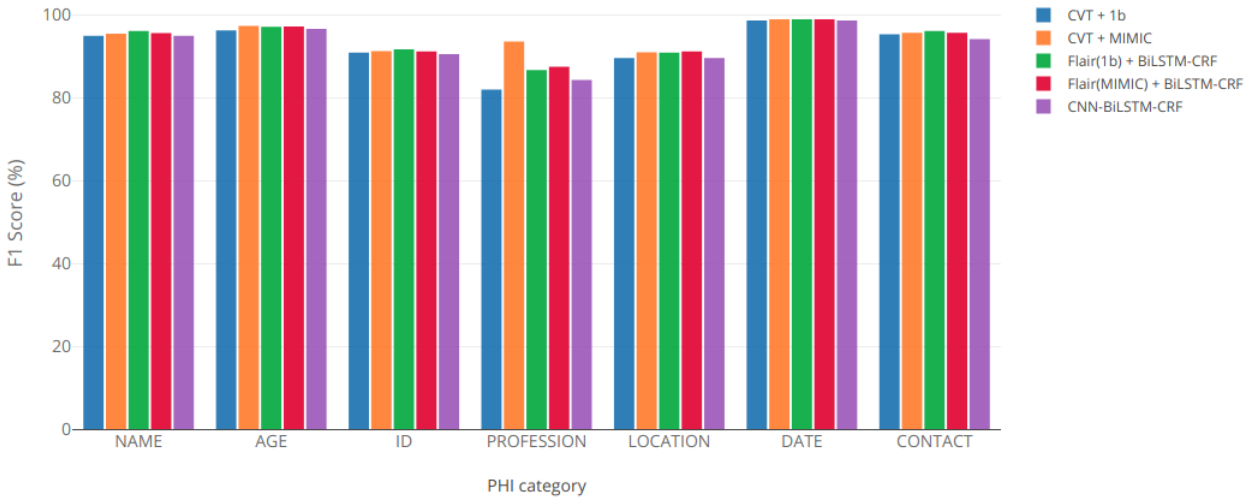

Figure 19: i2b2-PHI by Category

significant improvement. CVT(MIMIC) is able to perform best on the PROFESSION category. PROFESSION category is the hardest category to distinguish according to [Dernoncourt et al., 2017] and [Khin et al., 2018]. Good performance on this category is due to the fact that MIMIC dataset contains lots of profession examples and CVT approach is able to find the pattern in the profession category by using the auxiliary modules, which only see the limited input, while predicting the label. For example, in i2b2 dataset there is a sentence "and worked as a high school principle" in which all of the models except CVT(MIMIC) were unable to identify "high school principle" as a profession, but due to a lot examples like the following sentence "works as a high school teacher in the culinary arts" and use of auxiliary module, which only sees the limited input, model was able to identify the label.

i2b2-PHI	CVT (1b)	CVT (MIMIC)	Flair (1b)	Flair (MIMIC)	Bi-LSTM-CRF
Name	94.95	95.5	**96.15**	95.65	94.98
Age	96.28	**97.39**	97.16	97.24	96.66
Profession	82.0	**93.58**	86.7	87.48	84.28
Location	89.62	91.0	90.91	**91.19**	89.6
Date	98.72	**98.99**	98.96	**98.99**	98.65
Contact	95.33	95.69	**96.14**	95.7	94.21
ID	90.9	91.32	**91.74**	91.23	90.59

Table 10: F1-Score (%) on i2b2-PHI categories on 2014 i2b2/UTHealth shared task Track 1. Best performance according to each category is highlighted.

i2b2-PHI	CVT (1b)	CVT (MIMIC)	Flair (1b)	Flair (MIMIC)	Bi-LSTM-CRF
Name	94.39	94.94	**95.68**	95.12	94.36
Age	94.94	**96.71**	95.32	95.95	95.19
Profession	77.06	**92.35**	81.47	81.17	77.35
Location	85.84	88.6,	89.17	**89.3**	87.17
Date	98.8	**98.98**	98.8	98.88	98.67
Contact	94.99	95.47	95.23	**95.7**	95.23
ID	90.94	91.12	**91.3**	90.5	90.23

Table 11: Recall (%) on i2b2-PHI categories on 2014 i2b2/UTHealth shared task Track 1. Best performance according to each category is highlighted.

5.3.1 Flair(1b) VS Flair(MIMIC)

In Location category, Flair(MIMIC) was able to perform better than Flair(1b). This is due to the fact that lots of abbreviations were used as a hospital names, so the contextual embeddings trained on MIMIC dataset performs better as lots of hospital names have abbreviations in that dataset. For example, abbreviations like SAH (Saint Anthony Hospital) and ALS (Athens-Limestone Hospital). Also specific medical terms like "Hunt and Hess" and "Fisher scale" were wrongly categorized in NAME category by the Flair(1b) model. Flair(1b) was able to perform better on name, contact and ID category. Flair(1b) was able to identify common names, which were missed by Flair(MIMIC).

5.3.2 CVT(1b) VS CVT(MIMIC)

In profession category, CVT(MIMIC) was able to perform way better than all the other approaches. This is due to the fact that lots of profession are there in the MIMIC dataset and CVT approach is able to find the pattern in the profession category by using the auxiliary modules, which only see the limited input which predicting the label.

5.3.3 Training Models on Small Datasets

We evaluated CVT(MIMIC) against vanilla BiLSTM-CRF model [Ma and Hovy, 2016] for different dataset size. Our experiment shows that as the amount of labeled data decrease, improvement due to CVT approach is significant as compared to purely supervised model. CVT(MIMIC) reaches the same F1-score as supervised model by only using 50% of the data. We can see the comparison in Figure 20.

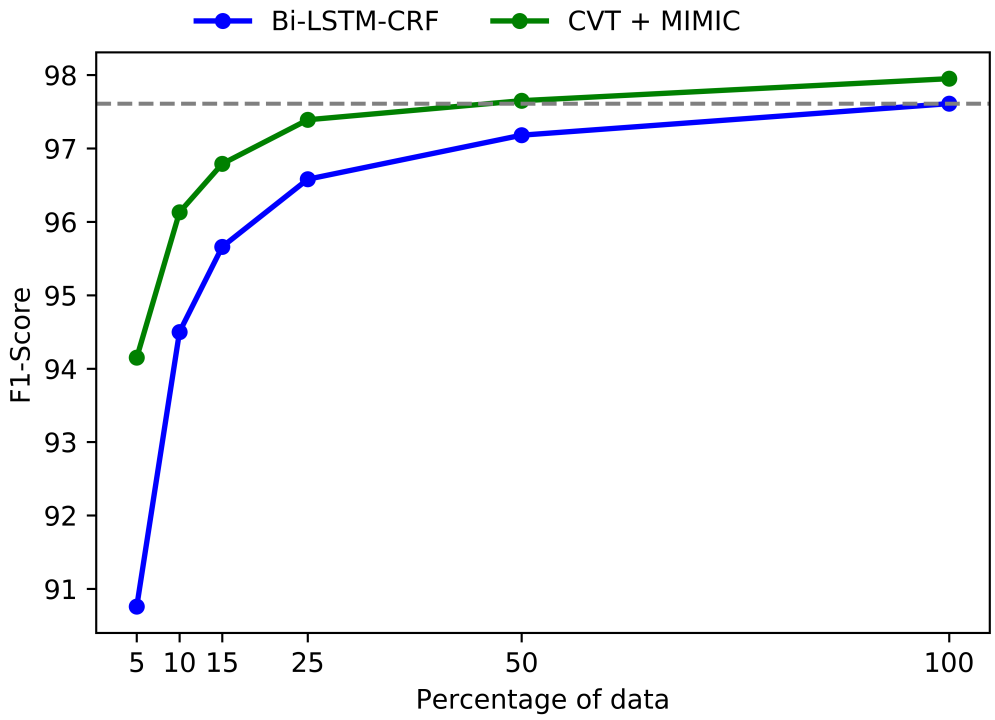

Figure 20: Training Models on Small Datasets

6 Conclusion and Future Work

In this thesis, we presented two approaches that use semi-supervised learning models for the task of de-identification of patient notes. Our results show that semi-supervised approaches are effective for the task of sequence modelling in medical domain. First approach uses pre-trained character contextual embeddings, which improves the performance of the model in contrast to purely supervised approach. This could be due to the fact that medical notes contain a lot of spellings mistakes and out-of-vocabulary word for which character-level could be effective. Also, training a language model on in-domain data to obtain embeddings helps in boosting the performance. Second approach is based on the principle of self-training, where model uses a mix of labeled and unlabeled data. This approach is only effective when we have in-domain data and fails to work if we use out-of-domain dataset.

As character-level contextual embedding perform better in terms of precision and CVT perform better in terms of recall, combination of character-level contextual embedding along with CVT approach could further help the model in achieving the optimal results, this could be further investigated. Obtaining more unlabeled data of patient notes could further improve the performance of the model. Optimization of hyperparameter can be investigated to check the impact on performance of the model.

7 Acknowledgments

I would like to thank Professor Dr. Hannah Bast for providing me the opportunity to work on such an interesting topic for my thesis. I express my sincere gratitude to Dr. Fang Wei-Kleiner, my second examiner. I also want to thank Daniel Dahlmeier for providing me an opportunity to work on a cutting edge research topic, Francesco Alda for his valuable time to supervise me, Markus Näther and Frank Dal-Ri for assisting me with technical support. My deepest gratitude to my family and friends for their continuous encouragement, and support, allowing me to concentrate on my thesis.

Bibliography

[Aberdeen et al., 2010] Aberdeen, J., Bayer, S., Yeniterzi, R., Wellner, B., Clark, C., Hanauer, D., Malin, B., and Hirschman, L. (2010). The mitre identification scrubber toolkit: design, training, and assessment. *International journal of medical informatics*, 79(12):849–859.

[Akbik et al., 2018] Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

[Berman, 2003] Berman, J. J. (2003). Concept-match medical data scrubbing: how pathology text can be used in research. *Archives of pathology & laboratory medicine*, 127(6):680–686.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

[Chelba et al., 2013] Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

[Chiu and Nichols, 2016] Chiu, J. P. and Nichols, E. (2016). Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

[Clark et al., 2018] Clark, K., Luong, M.-T., Manning, C. D., and Le, Q. V. (2018). Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

[Dernoncourt et al., 2017] Dernoncourt, F., Lee, J. Y., Uzuner, O., and Szolovits, P. (2017). De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*, 24(3):596–606.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

[Goldberger et al., 2000] Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220.

[Huang et al., 2015] Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

[Johnson et al., 2016] Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. (2016). Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.

[Khin et al., 2018] Khin, K., Burckhardt, P., and Padman, R. (2018). A deep learning architecture for de-identification of patient notes: Implementation and evaluation. *arXiv preprint arXiv:1810.01570*.

[Kotfic, 2014] Kotfic (2014). i2b2$_e$*valuation*$_s$*cripts*..

Krishnan, V. and Ganapathy, V. (2005). Named entity recognition. 2005. *Dostupno na: http://cs229. stanford. edu/proj2005/KrishnanGanapathy-NamedEntityRecognition. pdf (13.4. 2012)*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. pages 1097–1105.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Liu, L., Shang, J., Ren, X., Xu, F. F., Gui, H., Peng, J., and Han, J. (2018). Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Liu, Z., Chen, Y., Tang, B., Wang, X., Chen, Q., Li, H., Wang, J., Deng, Q., and Zhu, S. (2015). Automatic de-identification of electronic medical records using token-level and character-level conditional random fields. *Journal of biomedical informatics*, 58:S47–S52.

Liu, Z., Tang, B., Wang, X., and Chen, Q. (2017). De-identification of clinical notes via recurrent neural network and conditional random field. *Journal of biomedical informatics*, 75:S34–S42.

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Meystre, S. M., Friedlin, F. J., South, B. R., Shen, S., and Samore, M. H. (2010). Automatic de-identification of textual documents in the electronic health record: a review of recent research. *BMC medical research methodology*, 10(1):70.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ramshaw, L. A. and Marcus, M. P. (1999). Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning*, pages 147–155. Association for Computational Linguistics.

Saeed, M., Villarroel, M., Reisner, A. T., Clifford, G., Lehman, L.-W., Moody, G., Heldt, T., Kyaw, T. H., Moody, B., and Mark, R. G. (2011). Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 39(5):952.

Santos, C. D. and Zadrozny, B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Santos, C. N. d. and Guimaraes, V. (2015). Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Strubell, E., Verga, P., Belanger, D., and McCallum, A. (2017). Fast and accurate entity recognition with iterated dilated convolutions. *arXiv preprint arXiv:1702.02098*.

Stubbs, A., Kotfila, C., and Uzuner, Ö. (2015). Automated systems for the de-identification of longitudinal clinical narratives: Overview of 2014 i2b2/uthealth shared task track 1. *Journal of biomedical informatics*, 58:S11–S19.

Uzuner, Ö., Sibanda, T. C., Luo, Y., and Szolovits, P. (2008). A de-identifier for medical discharge summaries. *Artificial intelligence in medicine*, 42(1):13–35.

Yang, H. and Garibaldi, J. M. (2015). Automatic detection of protected health information from clinic narratives. *Journal of biomedical informatics*, 58:S30–S38.