IOSR Journal of Engineering (IOSR JEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 05-08

Causality Patterns and Machine Learning For The Extraction of Problem-Action Relations in Discharge Summaries

S.Swathi[1], Mr.V.Madhavan[2].
PG Student Mailam Engineering College
Associate Professor CSE Mailam Engineering College

***Abstract*:** *Clinical narrative text includes information related to a patient's medical history such as chronological progression of medical problems and clinical treatments. A chronological view of a patient's history makes clinical audits easier and improves quality of care. In this paper, we propose a clinical Problem-Action relation extraction method, based on clinical semantic units and event causality patterns, to present a chronological view of a patient's problem and a doctor's action. Based on our observation that a clinical text describes a patient's medical problems and a doctor's treatments in chronological order, a clinical semantic unit is defined as a problem and/or an action relation. Since a clinical event is a basic unit of the problem and action relation, events are extracted from narrative texts, based on the external knowledge resources context features of the conditional random fields. A clinical semantic unit is extracted from each sentence based on time expressions and context structures of events. Then, a clinical semantic unit is classified into a problem and/or action relation based on the event causality patterns of the support vector machines. Experimental results on Korean discharge summaries show 78.8% performance in the F1-measure. This result shows that the proposed method is effectively classifies clinical Problem-Action relations.*

I. Introduction

Cloud computing is a computing paradigm, where a large pool of systems are connected in private or public networks, to provide dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly.

The idea of cloud computing is based on a very fundamental principal of reusability of IT capabilities'. The difference that cloud computing brings compared to traditional concepts of "grid computing", "distributed computing", "utility computing", or "autonomic computing" is to broaden horizons across organizational boundaries.

Cloud Computing models

- Software as a Service (SaaS)
- platform as a service (paas)
- Infrastructure as a service(Iaas)

II. Model Architectures

Many different types of models were proposed for estimating continuous representations of words, including the well-known Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). In this paper, we focus on distributed representations of words learned by neural networks, as it was previously shown that they perform significantly better than LSA for preserving linear regularities among words [20, 31]; LDA moreover becomes computationally very expensive on large data sets. Similar to [18], to compare different model architectures we define first the computational complexity of a model as the number of parameters that need to be accessed to fully train the model. Next, we will try to maximize the accuracy, while minimizing the computational complexity.

For all the following models, the training complexity is proportional to

$$O = E * T * Q$$

(1)where E is number of the training epochs, T is the number of the words in the training set and Q is defined further for each model architecture. Common choice is E = 3□50 and T up to one billion. All models are trained using stochastic gradient descent and backpropagation [26].

FEEDFORWARD NEURAL NET LANGUAGE MODEL (NNLM):

The probabilistic feedforward neural network language model has been proposed in [1]. It consists of input, projection, hidden and output layers. At the input layer, N previous words are encoded using 1-of-V coding, where V is size of the vocabulary. The input layer is then projected to a projection layer P that has dimensionality N * D, using a shared projection matrix. As only N inputs are active at any given time, composition of the projection layer is a relatively cheap operation. The NNLM architecture becomes complex for computation between the projection and the hidden layer, as values in the projection layer are dense. For a common choice of N = 10, the size of the projection layer (P) might be 500 to 2000, while the hidden layer size H is typically 500 to 1000 units. Moreover, the hidden layer is used to compute probability distribution over all the words in the vocabulary, resulting in an output layer with dimensionality V . Thus, the computational complexity per each training example is

$$\mathbf{Q = N \times D + N \times D \times H + H \times V; (2)}$$

where the dominating term is H× V . However, several practical solutions were proposed for avoiding it; either using hierarchical versions of the softmax [25, 23, 18], or avoiding normalized models completely by using models that are not normalized during training [4, 9]. With binary tree representations of the vocabulary, the number of output units that need to be evaluated can go down to around log2(V). Thus, most of the complexity is caused by the term N × D ×H. In our models, we use hierarchical softmax where the vocabulary is represented as a Huffman binary tree.

This follows previous observations that the frequency of words works well for obtaining classes in neural net language models [16]. Huffman trees assign short binary codes to frequent words, and this further reduces the number of output units that need to be evaluated: while balanced binary tree would require log2(V) outputs to be evaluated, the Huffman tree based hierarchical softmax requires only about log2(Unigram perplexity(V)). For example when the vocabulary size is one million words, this results in about two times speedup in evaluation. While this is not crucial speedup for neural network LMs as the computational bottleneck is in the N×D×H term, we will later propose architectures that do not have hidden layers and thus depend heavily on the efficiency of the softmax normalization.

RECURRENT NEURAL NET LANGUAGE MODEL (RNNLM):

Recurrent neural network based language model has been proposed to overcome certain limitations of the feedforward NNLM, such as the need to specify the context length (the order of the model N), and because theoretically RNNs can efficiently represent more complex patterns than the shallow neural networks [15, 2]. The RNN model does not have a projection layer; only input, hidden and output layer. What is special for this type of model is the recurrent matrix that connects hidden layer to itself, using time-delayed connections.

This allows the recurrent model to form some kind of short term memory, as information from the past can be represented by the hidden layer state that gets updated based on the current input and the state of the hidden layer in the previous time step.The complexity per training example of the RNN model is

$$Q = H \times H + H \times V;$$

(3)where the word representations D have the same dimensionality as the hidden layer H.Again, the term H ×V can be efficiently reduced to H × log2(V) by using hierarchical softmax. Most of the complexity then comes from H × H.

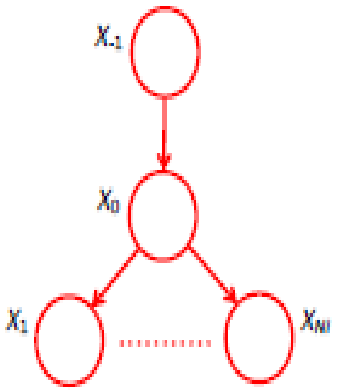
PARALLEL TRAINING OF NEURAL NETWORKS

To train models on huge data sets, we have implemented several models on top of a large-scale distributed framework called DistBelief [6], including the feedforward NNLM and the new models proposed in this paper. The framework allows us to run multiple replicas of the same model in parallel, and each replica synchronizes its gradient updates through a centralized server that keeps all the parameters. For this parallel training, we use mini-batch asynchronous gradient descent with an adaptive learning rate procedure called Adagrad [7]. Under this framework, it is common to use one hundred or more model replicas, each using many CPU cores at different machines in a data center.

III. New Log-Linear Models

In this section, we propose two new model architectures for learning distributed representations of words that try to minimize computational complexity. The main observation from the previous section was that most of the complexity is caused by the non-linear hidden layer in the model. While this is what makes neural networks so attractive, we decided to explore simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.

The new architectures directly follow those proposed in our earlier work [13, 14], where it was found that neural network language model can be successfully trained in two steps: first, continuous word vectors are learned using simple model, and then the N-gram NNLM is trained on top of these distributed representations of words. While there has been later substantial amount of work that focuses on learning word vectors, we consider the approach proposed in [13] to be the simplest one. Note that related models have been proposed also much earlier [26, 8].



CONTINUOUS BAG-OF-WORDS MODEL

The first proposed architecture is similar to the feed forward NNLM, where the non-linear hidden layer is removed and the projection layer is shared for all words (not just the projection matrix); thus, all words get projected into the same position (their vectors are averaged). We call this architecture a bag-of-words model as the order of words in the history does not influence the projection.

Furthermore, we also use words from the future; we have obtained the best performance on the task introduced in the next section by building a log-linear classifier with four future and four history words at the input, where the training criterion is to correctly classify the current (middle) word. Training complexity is then Q = N × D + D × log2(V): (4)We denote this model further as CBOW, as unlike standard bag-of-words model, it uses continuous distributed representation of the context. The model architecture is shown at Figure 1. Note that the weight matrix between the input and the projection layer is shared for all word positions in the same way as in the NNLM.

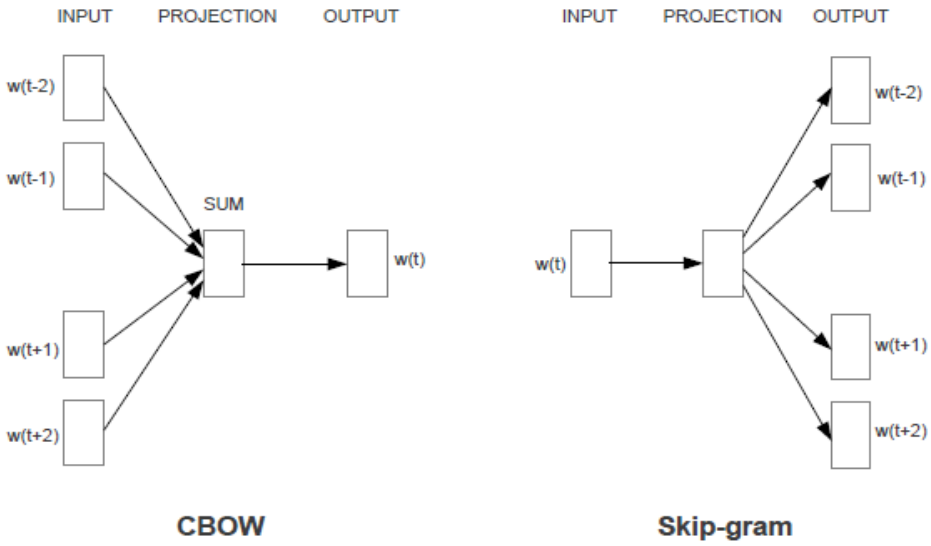
CONTINUOUS SKIP-GRAM MODEL

The second architecture is similar to CBOW, but instead of predicting the current word based on the context, it tries to maximize classification of a word based on another word in the same sentence. More precisely, we use each current word as an input to a log-linear classifier with continuous projection layer, and predict words within a certain range before and after the current word.

We found that increasing the range improves quality of the resulting word vectors, but it also increases the computational complexity. Since the more distant words are usually less related to the current word than those close to it, we give less weight to the distant words by sampling less from those words in our training examples. The training complexity of this architecture is proportional to

$$Q = C \times (D + D \times \log2(V)); \qquad (5)$$

Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. R words from the future of the current word as correct labels. This will require us to do R × 2 word classifications, with the current word as input, and each of the R + R words as output. In the following experiments, we use C = 10.



PROPOSED SYSTEM

The system receives as input the narrative text in addition to the narrative domain, i.e. the sets of allowable slot values for the variables representing the action and its arguments, i.e. characters/avatars, items/objects, tools and movement directions, in accordance with the elements defined in the graphical framework.

The proposed framework starts by extracting a set of cues from the text by using state-of-the-art algorithms for natural language processing. we introduced a framework to map text from written stories to a specific low-level KR. This new framework is able to reason with uncertainty, to integrate training from annotated data and constraints encoding information on mutually exclusive values, beyond evidence from external sources

The paper is organized as follows. Section II presents the state of the art; Section III contextualizes the instantiation problem resulting from the translation. To ease the understanding of the proposed framework, Section IV introduces the basic ideas and a high-level diagram of the proposed approach with its different constituent parts.

Once the context and details of the proposed framework have been explained, Section V provides the motivation for our approach. The adopted statistical model and features are described in Sections VI and VII respectively, while the statistical reasoning and the learning method are described in Sections VIII and IX respectively. The NLP pipeline is evaluated in Section X. Finally, Section XI presents the conclusions

IV. Conclusion

In this work, we introduced a framework to map text from written stories to a specific low-level KR. This new framework is able to reason with uncertainty, to integrate training from annotated data and constraints encoding information on mutually exclusive values, beyond evidence from external sources, such as information from the language model . Similar to other methods for structured prediction, the mapping aims at predicting the most likely structure by searching in the large search space derived from the exponential explosion of instance combinations, i.e. MAP inference. Therefore, an algorithm based on GA, able to exploit some properties of the Bayesian network, see (15) and (6), was developed for the statistical inference, requiring less CPU time than state-of-the-art tools, while providing parallel scalability to deal with larger domains. Moreover, the new constrained learning algorithm for Bayesian networks yielded performance gains in predicting the most likely structure given new sentences (unseen during the training).