A Method for Improving Unsupervised Intent Detection using Bi-LSTM CNN Cross Attention Mechanism

HAMID REZA, SADEGHI *

Amirkabir University of Technology, Department of Mathematics and Computer Science, Tehran, Iran

SAEED, SHIRY GHIDARY+

Staffordshire University School of Digital, Technologies and Arts, College Road, Stoke on Trent, UK

Benhur, Bakhtiari Bastaki†

Staffordshire University School of Digital, Technologies and Arts, College Road, Stoke on Trent, UK

Spoken Language Understanding (SLU) can be considered the most important sub-system in a goal-oriented dialogue system. SLU consists of User Intent Detection (UID) and Slot Filling (SF) modules. The accuracy of these modules is highly dependent on the collected data. On the other hand, labeling operation is a tedious task due to the large number of labels required. In this paper, intent labeling for two datasets is performed using an unsupervised learning method. In traditional methods of extracting features from text, the feature space that is obtained is very large, therefore we implemented a novel architecture of auto-encoder neural networks that is based on the attention mechanism to extract small and efficient feature space. This architecture which is called **Bi-LSTM CNN Cross Attention Mechanism (BCCAM**), crosswise applies the attention mechanism from Convolutional Neural Network (CNN) layer to Bi-LSTM layer and vice versa. Then, after finding a bottleneck on this auto-encoder network, the desired features are extracted from it. Once the features are extracted, then we cluster each sentence corresponding to its feature space using different clustering algorithms, including K-means, DEC, Agglomerative, OPTICS and Gaussian mixture model. In order to evaluate the performance of the model, two datasets are used, including ATIS and SNIPS. After executing various algorithms over the extracted feature space, the best obtained accuracy and NMI for ATIS dataset are 86.5 and 91.6, respectively, and for SNIPS dataset are 49.9 and 43.0, respectively.

CCS CONCEPTS • Information systems → Information retrieval → Information retrieval query processing → Query intent • Computing methodologies → Machine learning → Learning paradigms → Unsupervised learning → Topic modeling.

Additional Keywords and Phrases: Unsupervised Intent Detection, Cross Attention Mechanism, Text Clustering, Deep Clustering, Text Feature Extraction, Spoken Language Understanding.

1 INTRODUCTION

One of the first steps in many conversational AI systems used for parsing utterances in personal assistant systems is the identification of what the user intends to do (the intent) as well as the arguments of the intent (the slots) [1]. For

* Natural Language specialist - sadeghihamid@aut.ac.ir
+ Lecturer in Robotics and AI - saeed.shiryghidary@staffs.ac.uk
† Lecturer in Applied Software Engineering - b.b.bastaki@staffs.ac.uk

example, consider a request such as "*Book a ticket from Boston to Denver*", a first step in fulfilling this request is to identify that the user's intent is to book a flight and that the required source and target arguments of the request are expressed by the terms "*Boston*" and "*Denver*", respectively, as shown in Table 1. These two parts (User Intent Detection (UID) and Slot Filling (SF)), which constitute the core of a goal-oriented dialog system, are called Spoken Language Understanding (SLU) [2]. To design a SLU system, we need two types of labels for the dataset. In the first type, which involves the extraction of intentions according to the objectives in the sentence, one or more labels must be assigned to it. In order to avoid multiple labeling, combination of several labels can be considered as one label, which of course increases the number of labels. In the second type, a label (slot label) must be assigned to each of the words in the sentence. This label and its corresponding value help us to easily search the knowledge base and meet the user's needs.

Table 1 Details of SLU. In the slot filling module, each word with the label "o" means that the word is not useful for searching in the knowledge base.

Input	Book	a	ticket	from	Boston	to	Denver
Slots	o	o	o	o	source	o	target
Intent	Book a flight						
Domain	Airline travel						

This action is known as sequence labeling that is similar to actions such as Part Of Speech (POS) tagging and Name Entity Recognition (NER) [3]. There are a few numbers of SLU datasets available in the literature. As building a proper dataset is very time-consuming, expensive to produce, limited in size, and restricted to a specific domain, it makes them difficult to extend. Moreover, as the intent and slot label sets are usually decided by human experience, we usually do not know the exact intents or slots of a new unlabeled data, therefore the assigned label names may be subjective in some extent [4]. Three common datasets in this area are ATIS [5], SNIPS [6], and DSTC [7]. On the other hand, features extraction and labeling of SLU datasets are known as a complex process, since these datasets consists of textual data that are non-numeric, sequential and there may be ambiguity in text data as each word may be written in several forms [16].

For example, in the *Auto-Dialabel* model proposed in [4] an auto-encoder is used to extract appropriate features from the input data which is then classified by a hierarchical clustering algorithm. This model concentrates on improving clustering rather than increasing feature quality. It is known that most of clustering algorithms are sensitive to high-dimensional data. For instance, the curse of dimensionality can make k-means clustering very slow, and the existence of many irrelevant features may not allow identification of the relevant underlying structure in the data. So, reducing the size of feature vector can also implicitly increase clustering accuracy.

Here, our goal is to classify user intentions using a powerful feature extractor and a suitable clustering method, hence in this paper, we introduce the Bi-LSTM CNN Cross Attention Mechanism (BCCAM) model, which lets us extract features that have maximum information for the minimum length of a vector. A popular technique for overcoming the curse of dimensionality and to prevent overfitting is to regularize or constrain the parameters of the model. In the BCCAM model, we try to meet two goals at the same time: finding the model with the most generality and making the feature vector small.

The K-means algorithm may be the first choice for clustering, but due to the uncertainty in the performance of a specific meter such as Euclidean, other meters such as Cosine, Jaccard, Manhattan, etc. have also been investigated [8]. An important issue in such algorithms is how much the extracted features are compatible with the clusters? Deep Embedding for Clustering (DEC) algorithm that uses neural networks for clustering can find clusters along with feature extraction [9].

For sequential data such as text data usually an attention mechanism is used. Attention mechanism enforces the model to attend to the important parts of a sentence, in response to a specific aspect [10]. Attention has become an effective mechanism to obtain superior results, as demonstrated in image recognition, machine translation, reasoning about entailment and text summarization [11]. The attention mechanism has more flexibility in sequence length change, than Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN), and is more task/data-driven when modeling dependencies. Unlike sequential models, its computation can be easily and significantly accelerated by existing distributed/parallel computing methods [12]. In a work by [13], a model called Inner-attention has been proposed which is inspired by the observation that when human reads one sentence, they usually can roughly form an intuition about which part of the sentence is more important according to their past experience. Due to the differences between CNN and Bi-LSTM neural networks, each has a specific ability to extract information from the text. CNNs can extract morphological features and Bi-LSTMs are able to check for sentence continuity. Therefore, each has a feature that the other lacks, so by taking an attention output from each and applying it to the other, it can produce an output that is far richer than the output of each of them. This idea is applied to the BCCAM model.

Another issue considered in designing the BCCAM is, how to represent input data. The traditional methods such as one hot vector, bag of words and its variants (e.g. TF-IDF) are not good enough due to their excessive memory loss. The question is always "how to divide a sentence into smaller components for representation?" The answer is not unique. A sentence can be broken down into words (word-level) [13] or characters (char-level) [14] or both [15].

After decomposing the sentence into smaller components such as characters and words, each component must be represented as a feature vector, hence Embeddings are usually used for this purpose. Let F be the set of all decomposed components of sentences, the basic idea in feature embedding is to represent each member of F with a real-valued vector of some fixed dimension D. i.e, $f_i \in \mathbb{R}^D \; \forall \, i = 1, \ldots, |F|$. Word embedding forms the core of the text representation as a feature vector in the field of Natural Language Processing (NLP), while the character embeddings are less considered. For this reason, various word embeddings with different architectures have been proposed. Common and popular word embeddings include Glove [16] and Fasttext [17] and Word2Vec tool[1].

The rest of the paper is organized as follows: Section 2 presents background. Section 3 describes the proposed method. Section 4 discusses the findings and evaluates the proposed work with some of the state-of-the-art models and methods. Finally, a conclusion and future work are given in Section 5.

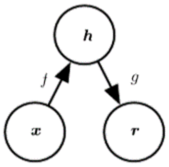

Figure 1 Representation of an Auto-Encoder neural network

[1] https://code.google.com/archive/p/word2vec/

2 BACKGROUND

2.1 Auto-Encoder neural network

Auto-Encoder is a neural network that attempts to reconstruct its input at its output [4]. As shown in Figure 1 a hidden layer h inside the auto-encoder, describes the encoded input. This neural network consists of two parts: an encoder function $h = f(x)$ and a decoder function $r = g(h)$.

Traditionally, auto-encoder neural networks are widely used for dimensionality reduction and feature learning, but the relationship between the latent variable and hidden neural networks has led to the creation of highly efficient variational auto-encoder neural networks.

The training of this network is performed by minimizing the following loss function:

$$L(x, g(f(x))) \qquad (1)$$

Where L is an error function that penalizes $g(f(x))$ when it is not similar to the target output.

2.2 Bidirectional Long Short-Term Memory

Bi-LSTM uses two LSTM neural networks in opposite directions to propagate the hidden state and cell state. In this way, the first LSTM reads the input sequence from beginning to the end and produces a hidden sequence and a cell sequence. On the other hand, the second LSTM does this by reading the sequence from end to the beginning. Finally, to produce the Bi-LSTM output, the hidden sequence of two LSTMs can be concatenated together or combined linearly. If $\vec{h}$ represents the forward LSTM hidden sequence and $\overleftarrow{h}$ represents the backward LSTM hidden sequence, for t = 1 to T the Bi-LSTM output is obtained as follows:

$$\vec{h}_t = H(W_{x\vec{h}}x_t + W_{\vec{h}\vec{h}}h_{t-1} + b_h) \qquad (2)$$

$$\overleftarrow{h}_t = H(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}}h_{t-1} + b_h) \qquad (3)$$

$$bo^c = [\vec{h}, \overleftarrow{h}] \qquad (4)$$

$$bo^l = W_{bo\,\vec{h}}\vec{h} + W_{bo\,\overleftarrow{h}}\overleftarrow{h}_t + b_y \qquad (5)$$

Where the W terms denote weight matrices (e.g. W_{xh} is the input-hidden weight matrix), the b terms denote bias vectors (e.g. b_h is the hidden bias vector) and H is the hidden layer function. H is usually an element-wise application of a sigmoid function and [,] is concatenation operator and $bo^c$and bo^ldenote Bi- LSTM output as concatenation and linear combination, respectively. In our work, bo^l is used as the output.

2.3 One-Dimensional Convolutional Neural Network

1D-CNN is a type of convolutional neural network that its filter size varies in only one dimension [15]. Due to its ability to extract morphological features, it is frequently used for extracting features from text [15]. Suppose that $\mathrm{x} = \{x_i\}_{i=1}^{T}$ is the input sequence that each of its elements is a vector in $\mathbb{R}^P$, where P is the size of the x_i for $i = 1$ to T. Due to the shape of x, it can be considered as a matrix in $\mathbb{R}^{P\times T}$. Now consider f as filter size. Using a filter $W_f \in \mathbb{R}^{P\times f}$, a convolution operation on P consecutive word vectors starting from t-th word outputs a scalar.

$$FM_t = F(W_f \bullet \mathrm{x}_{t:t+P-1} + b_f) \qquad (6)$$

Where $\mathrm{x}_{t:t+P-1} \in \mathbb{R}^{P\times f}$ is the matrix whose $i\text{-}th$ row is $x_i \in \mathbb{R}^f$, and $b_f \in \mathbb{R}$ is a bias. The symbol • refers to the dot product and F is the elementwise activation function. To make the input sequence size the same as the output, we add zero padding between the output elements therefore, it can be concluded $FM \in \mathbb{R}^T$. Now, consider co as the output of this network, using Q filters, the output size will be $Q \times T$.

2.4 Deep Embedding for Clustering

In traditional clustering algorithms such as k-means, the operation of finding clusters and finding features are not synchronous. To solve this problem, Deep Embedding for Clustering (DEC) algorithm is used. In this method, a new layer called clustering layer is added to the encoder section of an auto-encoder, and its parameters are initiated by an ordinary clustering algorithm such as K-means [9]. Then, *student's t-distribution* is used to create a similarity measure between each center and each embedded point.

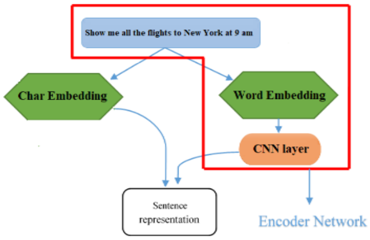

Figure 2 Representing the input sentence as a feature vector using concatenated word-level and char-level embedding

3 PROPOSED METHOD

3.1 Sentence representation

Consider the words "apple" and "Apple." The first word may refer to a fruit and the second word may refer to a company. Therefore, it is possible to change the meaning of a word and the whole sentence by changing a character from lowercase to uppercase or vice versa. Also, by changing one or more characters in a sentence, a new word may be derived that does not exist in the vocabulary that we are using. Hence, it is important to use char-level representation however, one drawback of using the character level representation on its own is that a large number of model parameters will be spent on learning the word structure which declines the accuracy of the model. In our work, to overcome this limitation, the simultaneous use of word-level and char-level representation would be used as it is more efficient.

Word-level embedding contains more information and is affected by char-level embedding during training, so in this architecture we have added a CNN layer after word-embedding (Figure 2). Simultaneously, char-level features are extracted from the input text and concatenated with CNN output. Note that this module uses Fasttext and Glove as pre-trained word-embedding.

3.2 Sentence representation

This model is in fact an auto-encoder that reconstructs the input at the output using a special type of attention mechanism**.** In this model, the features extracted by the method introduced in section 3.1, produces a feature matrix.

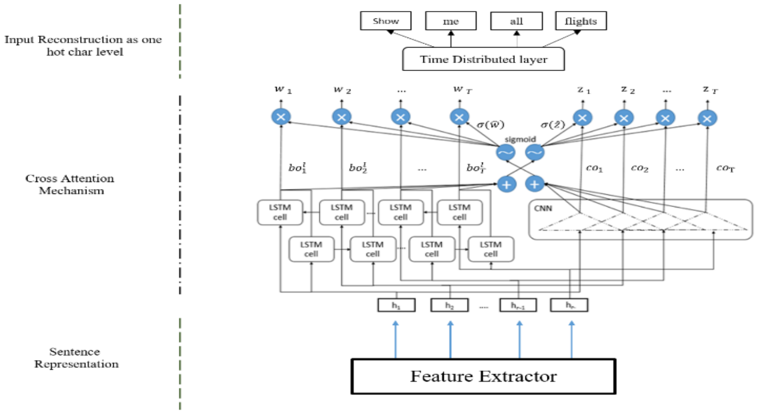

Figure 3 The BCCAM model in three levels. 1) Feature extraction. 2) Attention mechanism. 3) Reconstruction of input at output

This feature matrix is applied to attention mechanism which consists of two neural network layers, a 1D-CNN and a Bi-LSTM and then we try to reconstruct the same input at output. The output of both networks are a sequence of vectors. Now suppose $bo^l = (bo^l{}_1, \dots, bo^l{}_T)$ and $co = (co_1, \dots, co_T)$, which $bo_t \, . co_t \in \mathbb{R}^Q$ for $t = 1$ to T are the outputs of the Bi-LSTM and CNN layers, respectively. So Cross Attention Mechanism is defined as follows:

$$\hat{z} = \sum_{t=1}^{T} bo^l = \left(\sum_{t=1}^{T} bo^l{}_{t,1}, \dots, \sum_{t=1}^{T} bo^l{}_{t,Q}\right) \qquad (7)$$

$$\widehat{w} = \sum_{t=1}^{T} co = \left(\sum_{t=1}^{T} co_{t,1}, \dots, \sum_{t=1}^{T} co_{t,Q}\right) \qquad (8)$$

$$z = \left(\sigma(\widehat{w}) \odot bo_1, \dots, \sigma(\widehat{w}) \odot bo_T\right) \qquad (9)$$

$$w = \left(\sigma(\hat{z}) \odot co_1, \dots, \sigma(\hat{z}) \odot co_T\right) \qquad (10)$$

$$y_{CAM} = [z, w] \qquad (11)$$

As it turns out, the vectors $\hat{z}, \widehat{w} \in \mathbb{R}^Q$ and vectors $z, w \in \mathbb{R}^{T \times Q}$ and also $y_{CAM} \in \mathbb{R}^{T \times 2Q}$. Sigmoid function is also denoted with σ. In order to be able to apply the attention mechanism crosswise to each other, both bo^l and co sequences must have the same shape. Note that y_{CAM} is output of Cross Attention Mechanism (CAM).

Finally, by passing y_{CAM} to a Time Distributed layer, the BCCAM output is obtained as a one-hot vector. Figure 3 shows the complete model diagram.

3.3 Defining encoder part of BCCAM

After training BCCAM auto-encoder neural network, it must be broken down into the decoder and encoder sections. But the main question is where is the best place to do this division? The attention mechanism parameters and char embedding layer parameters affect the word embedding layer and its connected CNN layer. On the other hand, we should look for a point where the model parameters are minimized. In other words, the bottleneck should be found. Looking at the model, it is clear that the CNN layer has the minimum number of parameters. Therefore, due to the bottleneck that exists after the CNN layer connected to the word embedding, this CNN layer inherits all the features of the attention mechanism. Therefore, we cut the network from this point and divide it into two parts, encoder and other (the rest of BCCAM).

3.4 Defining encoder part of BCCAM

After extracting the features, we apply the K-means algorithm with different meters to find the clusters. We avoid using DEC alone because it needs an initial clustering estimate. However, we test each of the K-means outputs with different cluster meters, by DEC. We also apply a bottom-up Hierarchical Clustering algorithm called Agglomerative Clustering. In this algorithm, different meters are examined for merging clusters and the best output clusters are selected manually.

4 EXPERIMENTAL RESULTS

4.1 Datasets

We examined two datasets for text clustering: ATIS and SNIPS. These two datasets have labeled data for both slot filling and intent detection domains. The ATIS dataset contains flight information received from users and is collected by DARPA and is divided into training and test sections, of which 4978 are used as training data and 893 are used as test data. In addition, this dataset is labeled with 26 intent labels and 128 slot labels. The SNIPS dataset is gathered from snips personal audio assistant, where the number of samples for each intent is approximately the same. The dataset is also divided into 13784 training and 700 test samples. This dataset is tagged with 7 intent tags and 72 slot tags.

4.2 Evaluation Metrics

4.2.1 Measure of Clustering Accuracy

The accuracy measure is often used for evaluation of clustering algorithms when labeled data is available. But here the main challenge is that we cannot recognize which cluster is compatible with which label. Therefore, a one-to-one mapping should be created between the identifier of each cluster and its corresponding label in the dataset. In order to solve this problem, all one-to-one functions can be examined and the function that produces the most accuracy is introduced as a mapping function. The following relationship illustrates this subject:

$$acc = \max_{m} \frac{\sum_{i=1}^{N} 1\{l_i = m(c_i)\}}{n} \qquad (12)$$

where, l_i is real data label, c_i is identifier of i −th cluster and m is a one-to-one mapping to convert cluster label to a real data label. According to the high computational complexity of mapping m, one solution is to consider the most repetitive label in each cluster as the equivalent of that cluster. This way, the optimal mapping function can be found in linear time. It is important to note this measure will be independent of the actual value of the labels. That is, if the permutation of a class happens or a cluster changes, there is no change in the output of the evaluation.

4.2.2 Normalized Mutual Information (NMI)

This measure is also applicable when we have the data labels. Like the accuracy measure, this measure is independent of the actual value of the labels. This measure is also a symmetric measure, that is if we substitute the predicted values and the actual values with each other, no change in the obtained estimation will occur.

4.3 Training parameters

To train BCCAM, *Rmsprop* is used as optimizer and the Categorical Cross Entropy function is used as loss function. The batch size and the number of epochs are set to 32 and 150, respectively. The bottleneck layer has also 35×32 ×1 = 1120 features. The filter size for CNN layers is set to 3 or 5. The output vector length in the Bi-LSTM layer and the number of filters for the CNN layer in cross attention mechanism are both equal to 256.

Table 2 Accuracy and NMI for the main algorithms and both data and pre-trained Glove embedding

Accuracy			**NMI**			
ATIS dataset						
K-means	**Hierarchical**	**DEC**	**K-means**	**Hierarchical**	**DEC**	**Algorithm / Meter**
74.4%	**70.1%**	24.5%	15.0%	8.8%	88.3%	**Euclidean**
73.8%	69.8%	47.9%	86.2%	1.6%	80.2%	**Manhattan**
77.9%	16.5%	10.1%	**87.4%**	5.9%	**91.6%**	**Cosine**
SNIPS dataset						
K-means	**OPTICS**	**DEC**	**K-means**	**OPTICS**	**DEC**	**Algorithm / Meter**
21.4%	**22.4%**	33.1%	10.5%	17.1%	22.8%	**Euclidean**
20.6%	-	35.7%	1.4%	-	20.8%	**Manhattan**
46.4%	17.4%	**49.9%**	19.2%	10.1%	**34.6%**	**Cosine**
42.2%	-	-	15.4%	-	-	**Correlation**

Table 3 Accuracy and NMI for the main algorithms and both data and pre-trained Fasttext embedding

Accuracy			**NMI**			
ATIS dataset						
K-means	**Hierarchical**	**DEC**	**K-means**	**Hierarchical**	**DEC**	**Algorithm / Meter**
70.0%	66.6%	24.5%	10.9%	15.6%	**88.0%**	**Euclidean**
86.5%	**60.8%**	40.2%	77.2%	16.5%	75.2%	**Manhattan**
77.2%	25.2%	15.1%	**85.1%**	8.8%	80.4%	**Cosine**
SNIPS dataset						
K-means	**OPTICS**	**DEC**	**K-means**	**OPTICS**	**DEC**	**Algorithm / Meter**
29.3%	22.4%	25.1%	12.5%	10.4%	20.0%	**Euclidean**
39.5%	-	**38.4%**	**43.0%**	-	21.1%	**Manhattan**
35.0%	17.4%	30.9%	8.8%	11.1%	**34.5%**	**Cosine**
14.9%	-	-	29.6%	-	-	**Correlation**

4.4 Results

The BCCAM model is trained based on two pre-trained embedding, two datasets and several state-of-the-art algorithms. Applied to the ATIS and SNIPS datasets. For Glove Embedding (Table 2) the highest accuracy and NMI are related to K-means with cosine meter.

Table 4 Comparison of accuracy measure with other models for ATIS dataset

Models	**Intent Labeling Acc(%)**
Topic model	25.4%
CDSSM vector	20.7%
Glove Embedding	25.6%
Auto-dialabel	84.1%
BCCAM	**86.5%**

Table 5 Results of Gaussian Mixture Model algorithm for different iterations

Acc	**NMI**	**Evlauation / Max-Iteration**
13.8%	14.9%	100
13.1%	15.6%	500
13.7%	15.6%	1000

By switching embedding to Fasttext (Table 3), and retraining the network, the accuracy on ATIS dataset will increase to 86.5, but the NMI will decrease. Also, according to Table 3, for the SNIPS dataset, the highest accuracy and

NMI are for DEC (38.4%) and K-means (49.0%) with Manhattan meter, respectively. Also, according to the results of the models in Table 4, it can be inferred that our model has shown a better performance than other models on the ATIS dataset.

With a little investigation, it can be realized that the accuracy and NMI of these two datasets are far apart. The reason for this is probably because the classes in the SNIPS dataset have a fairly uniform and large size distribution, but the distribution of classes in the ATIS dataset is unevenly distributed. Table 5 also shows accuracy and NMI for the Gaussian Mixture Model algorithm and for Fasttext Pre-trained word-embedding.

5 CONCLUSION AND FUTURE WORKS

The unsupervised method described in this article can be used in conversation systems where we face data shortage or a time-consuming data collection process. With this method, it is possible to tag unlabeled data and apply them for training the model along with labeled data. So, by integrating this method with supervised methods, the performance of Goal Oriented Dialogue systems can be improved.

REFERENCES

[1] Sebastian Schuster, Sonal Gupta, Rushin Shah, Mike Lewis 2016. Cross-lingual transfer learning for multilingual task oriented dialog. arXiv:1609.01454.

[2] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, Geoffrey Zweig. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. IEEE/ACM Transactions on Audio, Speech, and Language Processing 23, 3 (2015), 530-539.

[3] Yoshimasa Tsuruoka, Jun'ichi Tsujii 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05.

[4] Chen Shi, Qi Chen, Lei Sha, Sujian Li, Xu Sun, Houfeng Wang, Lintao Zhang 2018. Auto-Dialabel: Labeling Dialogue Data with Unsupervised Learning. Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing.

[5] Charles T. Hemphill, John J. Godfrey, George R. Doddington. The ATIS Spoken Language Systems Pilot Corpus. Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990, (1990).

[6] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, Joseph Dureau 2018. Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces. arXiv:1805.10190v3.

[7] Matthew Henderson, Blaise Thomson, Jason Williams 2014. The Second Dialog State Tracking Challenge. Proc. of the 15th Annual Meeting of the SIGDIAL.

[8] Anna Huang. Similarity Measures for Text Document Clustering 2008. Proceedings of the New Zealand Computer Science Research Student Conference.

[9] Junyuan Xie, Ross Girshick and Ali Farhadi 2016. Unsupervised Deep Embedding for Clustering Analysis. Proceedings of the 33rd International Conference on Machine Learning.

[10] Bingning Wang, Kang Liu and Jun Zhao 2016. Inner Attention based Recurrent Neural Networks for Answer Selection. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.

[11] Sumit Chopra, Michael Auli, Alexander M. Rush 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

[12] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, Chengqi Zhang 2018. DiSAN: Directional self-attention network for rnn/cnn-free language understanding. 32^{nd} AAAI Conference on Artificial Intelligence.

[13] Yang Liu, Chengjie Sun, Lei Lin, Xiaolong Wang 2016. Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention. arXiv:1605.09090.

[14] Dan Klein, Joseph Smarr, Huy Nguyen, Christopher D. Mannin. Named Entity Recognition with Character-Level Models. Proc. of the 7th Conf. on Natural Language Learning at HLT-NAACL, (2003), 180–183.

[15] Xuezhe Ma and Eduard Hovy 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.

[16] Jeffrey Pennington, Richard Socher, Christopher D. Manning 2014. Glove: Global Vectors for Word Representation. Proc. of the 2014 EMNLP Conf.

[17] Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov. Bag of Tricks for Efficient Text Classification. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: V2, (2017), 427–431