Learning Ticket Similarity with Context-sensitive Deep Neural Networks

Durga Prasad Muni, Suman Roy, Yeung Tack
Yan John John Lew Chiang, Navin Budhiraja
Infosys Limited
#44 Electronic City, Hosur Road
Bangalore, India 560100
{DurgaPrasad_Muni,Suman_Roy,Yeung_Chiang,Navin.Budhiraja}@infosys.com

Iheb Ben Abdallah
Computer Science and Electrical Engineering,
Ecole CentraleSupelec
Grande Voie des Vignes, Châtenay-Malabry
Paris, France 92290
Iheb.Benabdallah@supelec.fr

ABSTRACT

In Information Technology Infrastructure Library (ITIL) services a sizable volume of tickets are raised everyday for different issues to be resolved so that the service can be delivered without interruption. An issue is captured as summary on the ticket and once a ticket is resolved, the solution is also noted down on the ticket as resolution. It is required to automatically extract information from the description of tickets to improve operations like identifying critical and frequent issues, grouping of tickets based on textual content, suggesting remedial measures for them etc. In an earlier work we have proposed deep learning based recommendation algorithm for recovering resolutions for incoming tickets through identification of similar tickets. In this work we use similar deep neural based framework to compute the similarity between two tickets by considering context information. In particular, we append the feature representation of tickets with context information to be fed as input to deep neural network. Our learning algorithm seems to improve the performance of similarity learning using the traditional techniques. In particular the context-enriched DNN approach on average improves the performance by 5-6% in comparison to simple DNN-based approach.

CCS CONCEPTS

•Computing methodologies → *Neural networks;* **•Applied Computing → Document management and text processing;**

KEYWORDS

Deep Learning; Neural Network; Deep Neural Network; Ticket similarity; Short Text Similarity; Context Information

ACM Reference format:
Durga Prasad Muni, Suman Roy, Yeung Tack Yan John John Lew Chiang, Navin Budhiraja and Iheb Ben Abdallah. 2016. Learning Ticket Similarity with Context-sensitive Deep Neural Networks. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 9 pages.
DOI: 10.475/123_4

This work was done when Iheb Ben Abdallah was an intern at Infosys Ltd during July-Aug, 2017.

Conference'17, Washington, DC, USA

DOI: 10.475/123_4

1 INTRODUCTION

Ticketing system forms a core component for the problem and configuration management for Information Technology Infrastructure Library (ITIL) services. Vast number of tickets are raised on the ticketing system by users with a view to resolve issues/concerns faced by them while using different support systems. These incident data in the form of tickets can be used for different purposes such as SLA calculation, forecasting, optimum resource level checking, performance metrics computation etc. A ticketing system tries to minimize the business impact of incidents by addressing the concerns of the raised tickets. The incident tickets record symptom description of issues, as well as details on the incident resolution using a range of structured fields such as date, resolver, categories, affected servers and services and a couple of free-form entries outlining the description/summary of issues, note by users/administrators etc. Once a ticket is resolved, the solution is also noted down on the ticket as resolution as texts. Manual screening of such a huge volume of tickets would be laborious and time-consuming. One needs to extract information automatically from the description of tickets to gain insights in order to improve operations like identifying critical and frequent issues, grouping of tickets based on textual content, suggesting remedial measures for them and so forth.

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [16]. They typically use artificial neural networks with several layers - these are called deep neural networks. Deep neural networks (DNN) are becoming popular these days for providing efficient solutions for many problems related to language and information retrieval [4, 6, 7]. In this work we use context sensitive Feed Forward deep neural network (FF-DNN) for computing ticket similarity. In an earlier work we have proposed an automated method based on deep neural networks for recommending resolutions for incoming tickets through identification of similar tickets. We use ideas from deep structured semantic models (DSSM) for web search for such resolution recovery. We take feature vectors of tickets and pass them onto DNN to generate low dimensional feature vectors, which helps compute the similarity of an existing ticket with the new ticket. We select a couple of tickets which has the maximum similarity with the incoming ticket and publish their resolutions as the suggested resolutions for the latter ticket.

We modify this framework of similarity computing by taking into account context information. Context may appear in various forms ranging from neighboring sentences of a current sentence in

a document [19] to topics hidden within the sentence [15] and to the document containing the sentence in question [12]. As neural networks are normally trained with local information it makes sense to integrate context into them. Global infomration which may be embedded in this context information can often be instrumental in guiding neural networks to generate more accurate representations. In this task we consider topics associated with tickets as the context information and feed them as vectors to the deep networks along with the feature vectors for tickets. The neural network outputs a low dimensional vector for each of the input vectors. These two low-dimensional vectors for a ticket are combined to compute the similarity between two tickets. A schematic diagram of our method is shown in Figure 2.

We employ this context-driven similarity technique to three semantic similarity tasks: *contextual ticket similarity* with respect to a tuple in which we aim to predict similarity between a given pair of tickets in a tuple, *ticket ranking* in which aim to retrieve semantically equivalent tickets with respect to a given test ticket, and *resolution recommendation* in which we aim to suggest resolutions for a given ticket [22]. We carry out an extensive experimentation on these tasks. Our technique shows an appreciable improvement of 5-6% over non-context-based DNN approach for most of these semantic similarity tasks. The contributions of our work lie in proposing an approach of injecting context into deep neural networks and showing an input of such additional information to deep neural networks improves the representation of them for various similarity tasks.

1.1 Related Work

Neural networks are effectively used to compute semantic similarity between documents [8, 23, 26]. Deep learning has also been used to find similarity between two short texts. Hu et.al. [11] have used convolutional neural networks for matching two sentences. The approach could nicely represent the hierarchical structures of sentences with their layer-by-layer composition and pooling and thus capture the rich matching patterns at different levels. Lu and Li [20] have proposed a deep architecture that can find a match between two objects from heterogeneous domains. In particular, they apply their model to match short texts meant for task such as finding relevant answers to a given question and finding sensible responses for a tweet. In [27], convolutional neural network has been used for ranking pairs of short texts, wherein the optimal representation of text pairs and a similarity function are learnt to relate them in a supervised manner. Long Short-term memory (LSTM) also used to find similarity between sentences in [3, 21].

The idea of using FF-DNN in our work originates from work on learning deep structured latent models for web search [9, 10, 13]. Motivated by these ideas we have used deep neural network models to recommend resolutions for tickets in ITIL services [22]. We project existing tickets and an incoming ticket to a common low dimensional space and then compute the similarity of the new ticket with other tickets. We select the ticket which has highest similarity with the new ticket and pick up the resolution of the former as the recommended resolution for the new ticket. In this work, we integrate context into deep neural networks for computing similarity between two tickets (the motivation came from [2]). which is a

IncidentType	Category	SubCategory	ApplicationName	SubmitDateTi	ClosedDateTi	LastModified	Statu	Priorit	Summary	Note
AS/400 - User Profile	Administration	Configuration	AS400 Legacy - Manufa	9/1/2015 17:20	9/4/2015 0:51	9/4/2015 0:51	Closed	P4 - Stan	AS/400 ID Requests - NBTY	09-04 AM -
AS/400 - Software De	Software	Application Errors	AS400 Legacy - Wareho	9/17/2015 22:23	9/21/2015 22:02	9/23/2015 19:00	Closed	P3 - Mod	Inventory/BU Location - Update error (Please see Attached Screenshot)	09-23 PM -
AS/400 - User Profile	Administration	Configuration	AS400 Legacy - Wareho	5/18/2015 19:19	5/28/2015 0:13	5/28/2015 0:13	Closed	P4 - Stan	?Need access to be authorized to use miscellaneous	05-28 AM -
AS/400 - Software De	Software	Application Functi	AS400 Legacy - Wareho	6/29/2015 21:17	7/6/2015 5:10	7/7/2015 19:00	Closed	P4 - Stan	We been trying to get a CPO to be unlocked or fixed	07-07 PM -
AS/400 - Software De	Software	Application Errors	AS400 Legacy - Wareho	8/3/2015 18:39	8/7/2015 0:34	8/10/2015 0:00	Closed	P4 - Stan	Problem with G.T.S menu - (See Attached)	08-07 AM -
AS/400 - Software De	Software	Software Enhance	AS400 Legacy - Wholes	6/10/2015 6:16	6/29/2015 5:41	6/29/2015 5:41	Closed	P4 - Stan	Tammy needs to receive emails through the winspo	06-29 AM -
AS/400 - User Profile	Administration	Configuration	AS400 Legacy - Wareho	6/11/2015 9:20	6/16/2015 9:32	6/16/2015 9:32	Closed	P4 - Stan	Can we please have Andrew's Regular Out Queue ch	06-16 AM -
Oracle - ORPOS POS (	Software	Application Errors	AS400 Legacy - Retail	9/22/2015 12:14	#############	#############	Closed	P4 - Stan	From: Andrew Boland [mailto:AndrewBoland@hollandandbarrett.com]	10-23 PM -
AS/400 - Software De	Software	Configuration	AS400 Legacy - Retail	6/9/2015 9:26	6/11/2015 9:26	6/11/2015 9:26	Closed	P4 - Stan	Bonus Rebate Expiring Email - File Due 6/11	06-11 AM -
AS/400 - Software De	Software	Configuration	Legacy RCA	10/6/2015 12:35	#############	#############	Closed	P4 - Stan	Jenny Stayte Sent: Tuesday, October 06, 2015 6:34 AM	10-22 AM -

Figure 1: Snapshot of relevant parts of incident ticket data

novelty of our work. In addition to feature vector, a context vector for a ticket is also injected into the deep neural network to obtain a combination of two low dimensional vectors which are then used to compute the similarity of a pair of tickets. Recently Amiri *et al.* have used an extended framework of deep auto encoders with context information to learn text pair similarity in [2]. In that the authors use context information as low dimensional vectors which are injected to deep autoencoders along with text feature vectors for similarity computation. While the authors use auto encoders for finding similarity of texts, in this work we use feed forward deep neural network as it provides a more suitable framework to compute the similarity between tickets.

Organization of the paper The paper is organized as follows. We describe the ticket schema that we consider in Section 2. The Feed forward Deep Neural Network (FFDN) used for text similarity is introduced in Section 3. The context-sensitive Feed forward Deep Neural Network is introduced in Section 4. We describe our approach to compute similarity between two tickets using context-sensitive FFDNN in Section 5. Experimental results are discussed in Section 6. Finally we conclude in Section 7.

2 TICKET DATA SET

We consider incident tickets with similar schema which are frequent in ITIL. These tickets usually consist of two fields, fixed and free form. Fixed-fields are customized and inserted in a menu-driven fashion. Example of such items are the ticket's identifier, the time the ticket is raised or closed on the system or, if a ticket is of incident or request in nature. Various other information are captured through these fixed fields such as category of a ticket, employee number of the user raising the ticket etc, and also Quality of Service parameters like response time, resolution time of the ticket etc. There is no standard value for free-form fields. The concern/issue for raising a ticket is captured as "call description" or "summary" as free-formed texts, - it can be a just a sentence that summarizes the problem reported in it, or it may contain a detailed description of the incident. By using this freely generated part of tickets, administrators can get to know about unforeseen network incidents and can also obtain a much richer classification. A small note is recorded as resolution taken for each ticket. A small part of ticket data is shown in Figure 1.

2.1 Feature vector creation from Ticket Data

We assume a free field of a ticket to contain a succinct problem description associated with it in the form of a summary (or call description). We extract features from the collection of summaries of tickets using light weight natural language processing. As a pre-processing we remove the tickets which do not contain a summary/resolution. In the beginning we lemmatize the words in the summary of tickets. Then we use Stanford NLP tool to parse the useful contents in the summary of the tickets and tag them as tokens. Next we set up some rules for removing tokens which are stop words. We compute document frequency (DF)[1] of each lemmatized word. We discard any word whose DF is smaller than 3. A ticket summary may contain some rare words like, name of a person (user who raised the ticket) and some noise words. By discarding words with DF less than 3, we can remove these rare words which do not contribute to the content of ticket summary. In this way, the feature vector size could be reduced significantly. We perform some other pre-processing and select bi-grams and tri-grams as terms (keyphrases), the details of which are described in [24]. Finally, a profile of a ticket is given as, $T = (x_1, \ldots, x_n) = \vec{x}$, where $x_1, \ldots, x_n$ are the appropriate weights for the chosen words $w_1, \ldots, w_n$ respectively from the summary of T. We shall use TF*IDF [25] of a word as its weight[2].

2.2 Relational schema on fixed elements

The fixed field entries of a ticket can be represented using a relational schema. For that we shall consider only a limited number of fixed fields of a ticket for choosing attributes that reflect its main characteristics (the domain experts' comments play an important role in choosing the fixed fields), for example, the attributes can be, - application name, category and sub-category. They can be represented as a tuple: Ticket(application name, category and sub-category). Each of the tuples corresponding to entries in the fixed fields in the ticket can be thought of an instantiation of the schema. Examples of rows of such schema can be, (AS400 - Legacy Manufacturing, Software, Application Errors), (AS400 Legacy - Retail, Software, Application Functionality Issue) etc. The relation key can vary from 1 to number of distinct tuples in the schema. One such key can hold several Incident IDs, that is, it can contain several tickets with different IDs.

3 FEED FORWARD DEEP NEURAL NETWORK FOR TEXT SIMILARITY

Deep learning (DL) is based on algorithms that learn multiple levels of representation and abstractions in data. It typically uses artificial neural networks with several layers. These are called deep neural networks.

3.1 Architecture of Feed forward Deep Neural Network

Feed forward Deep Neural Networks (FF-DNNs) have multiple hidden layers. Each layer in a feed forward Deep Neural Network (FF-DNN) adds its own level of non-linearity that can solve more complex problems. The FF-DNN used in this paper to find similar texts process feature vectors of texts, works in two stages. First, the FF-DNN maps high-dimensional sparse text feature vector of a document (text) layer by layer into low-dimensional feature vector. In the next stage, the low dimensional feature vectors are passed through cosine similarity function nodes to compute the similarity between two texts.

[1]Document frequency of a word is the number of tickets (ticket summaries) containing the word in the data set (corpus) [17]

[2]TF*IDF is a popular metric in the data mining literature [17]

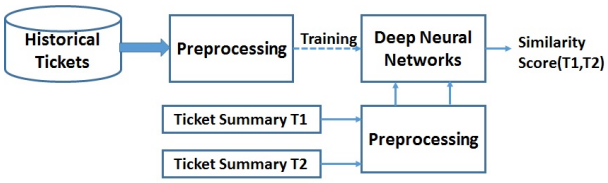

Figure 2: A schematic diagram of our Approach

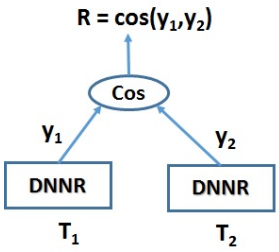

Figure 3: Architecture of Feed Forward Deep Neural Network for text similarity

The architecture of feed forward deep neural network FF-DNN [13] that we use for our purpose here is shown in Fig 3. Given a text (document) the objective of the FF-DNN model is to find the text (document) from the existing texts (documents) that is the most similar *wrt* the new one. To fulfill this goal, we can train the FF-DNN with a set of texts along with the similar and dissimilar texts.

3.2 Structure of the FF-DNN

We have used the FF-DNN shown in Fig 3 to find similarity between the new ticket and a set of existing tickets in [22]. Prior to computing similarity, the FF-DNN reduces the high dimensional feature vectors representing summaries of tickets into low-dimensional vectors. To accomplish this it uses DNNR, a multilayer feed-forward deep neural network that reduces dimension.

The structure of the DNNR is given in Fig 5. The input layer of DNNR consists of n number of nodes where n is the size of the feature vector of document. Let there be $N-1$ number of hidden layers. It has one output layer which is N^{th} layer of the network (excluding input layer). Let $\vec{x}$ be the input feature vector, y as output vector. Let h_i, $i = 1, 2, \ldots, N-1$ be the ith intermediate hidden layer, $\mathbf{W}_i$ be the i^{th} weight matrix and b_i be the i^{th} bias vector. We have then

$$\begin{aligned} h_1 &= \mathbf{W}_1 \vec{x} \\ h_i &= f(\mathbf{W}_i h_{i-1} + b_i), i = 2, 3, \ldots, N-1. \\ y &= f(\mathbf{W}_N h_{N-1} + b_N) \end{aligned} \tag{1}$$

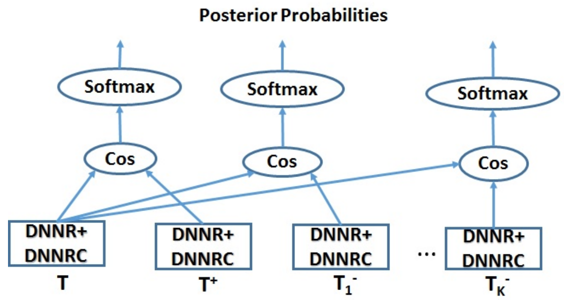

Figure 4: Structure of Feed Forward Deep Neural Network during training

We consider $tanh(z)$ as the activation function at the output layer and at the hidden layers. The activation function is defined as

$$f(z) = tanh(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}} \tag{2}$$

The output of DNNR is passed through a cosine similarity function as shown in Fig 3.

4 CONTEXT-SENSITIVE FFDNN

We now extend FFDNN to incorporate context information about inputs. For each ticket T in the training set represented with its feature vector as $\vec{x} \in \mathbb{R}^n$, we have generated a context vector $\vec{c}_x \in \mathbb{R}^k$ containing contextual information about the input. The input and the target task can determine the nature of the context vector.

We need to pass the context vector $\vec{c}_x$ through DNNRC that reduces the dimension. Here DNNR and DNNRC have the same structure, but they accept different weight and bias parameters, see Fig 5 (indexed by $\mathbf{W}$ and $\mathbf{V}$ respectively). For producing low-dimensional vectors through DNNRC we follow the same approach as discussed for the basic FFDNN captured by Eqn 1. Here we assume $l_i, i = 1, 2, \ldots, N-1$ be the ith intermediate hidden layer representation of DNNRC, $\mathbf{V}_i$ be the i^{th} weight matrix and d_i be the i^{th} bias vector. The FFDNN maps the inputs $\vec{c}_x$ to the context-sensitive representations $l_2, l_3, \ldots l_{N-1}$ at hidden layers $2, 3, \ldots, N-1$, and a context-sensitive output y_c given by Eqn 3.

$$\begin{aligned} l_1 &= \mathbf{V}_1 \vec{c}_x \\ l_i &= f(\mathbf{V}_i l_{i-1} + d_i), i = 2, 3, \ldots, N-1. \\ y_c &= f(\mathbf{V}_N l_{N-1} + d_N) \end{aligned} \tag{3}$$

Each ticket T is now represented as a context rich feature vector $(\vec{y}, \vec{y}_c)$ for further analysis.

4.1 Training of the FF-DNN

The structure of the Context based FF-DNN during training is given in Fig 4. To train the (context based) FF-DNN, we take a set of M ticket summaries, $\{T_m : m = 1, 2, \ldots, M\}$. Each of the summary of ticket T_m is coupled with one similar ticket T_i^{m+} and three dissimilar tickets $T_{i'}^{m-}$. These four similar and dissimilar tickets are represented by a set $\mathbf{T_m}$.

The given ticket T_m and each of the similar and dissimilar tickets are fed to the DNNR one by one. The structure of the context-sensitive FFDNN that we use is shown in Fig 6. Let (y^m, y_c^m) be the combined output feature vector for T_m and (y^i, y_c^i) be the combined output feature vector for T_i.

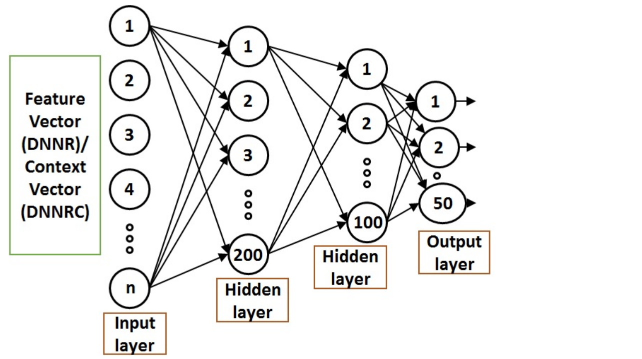

Figure 5: DNNR or DNNRC: Part of DNN that reduces the dimension

The cosine similarity between the output vector (y^m, y_c^m) and another output vector (y^i, y_c^i) are computed using Eqn 4 given by the cosine similarity $R(T_m, T_i)$ as below:

$$\begin{aligned} R(T_m, T_i) &= \cos(y^m, y^i) + \lambda \cos(y_c^m, y_c^i) = \frac{{y^m}^T y^i}{||y^m|| ||y^i||} \\ &+ \lambda * \frac{{y_c^m}^T y_c^i}{||y_c^m|| ||y_c^i||}, \ \lambda \in [0, 1] \end{aligned} \tag{4}$$

These R-values of similar and dissimilar tickets *wrt* T_m are supplied to the Softmax function as shown in Fig 4. The Softmax function computes posterior probabilities [13]. The posterior probability for $R(T_m, T_i^m)$ is given in Eqn 5 as below.

$$P(T_i^m | T_m) = \frac{exp(\gamma R(T_m, T_i^m)}{\sum_{T_{i'}^m \in \mathbf{T_m}} exp(\gamma R(T_m, T_{i'}^m))}, \tag{5}$$

where γ is the smoothing parameter in the Softmax function. As our objective is to find the most similar ticket for a given ticket T_m, we maximize the posterior probability for the similar (or positive) tickets. Alternatively, we minimize the following loss function

$$L(\Omega) = -\log \prod_{(T_m, T_i^{m+})} P(T_i^{m+} | T_m) \tag{6}$$

where Ω denotes the set of parameters $\{\mathbf{W}_i, b_i, \mathbf{V}_i, d_i : i = 1, 2, \ldots, N\}$ of the neural networks. $L(\Omega)$ is differentiable *wrt* Ω as it is continuous and its (partial) derivatives are also continuous (see Section A). So, the FF-DNN can be trained using gradient-based numerical optimization algorithm. The parameters in Ω are updated as

$$\Omega_t = \Omega_{t-1} - \epsilon_t \frac{\partial L(\Omega)}{\partial \Omega}|_{\Omega = \Omega_{t-1}}, \tag{7}$$

where ϵ_t is the learning rate at the t^{th} iteration, Ω_t and Ω_{t-1} are the model parameters at the t^{th} and $(t-1)^{th}$ iteration, respectively. For other details see [13, 22].

4.2 Context Extraction

Context relates to the information content present in the summary of tickets. As context information is not available with the ticket description we resort to topic models [5, 29] to obtain contexts for

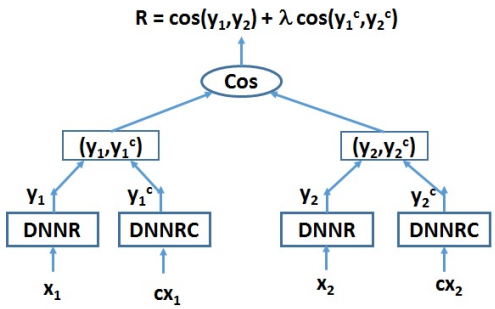

Figure 6: Structure of Context Sensitive Feed Forward Deep Neural Network for text similarity

each individual ticket. Given feature vector representation of tickets $T_1, \ldots, T_n$ as $\vec{x}^1 = (x_{1,1}, \ldots, x_{1,m}) \ldots, \vec{x}^n = (x_{n,1}, \ldots, x_{n,m})$ respectively we can compute the ticket-term matrix $\mathbf{X}$ as

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,m} \\ \vdots & \vdots & \vdots \\ x_{n,1} & \cdots & x_{n,m} \end{bmatrix}$$

The matrix $\mathbf{X}$ is of dimension $n \times m$ where the number of tickets and terms are n and m respectively. Recall each element $x_{i,j}$ here denotes the TF*IDF value of term j in Ticket i. Generally the rows of $\mathbf{X}$ are normalized to have unit Euclidean length. Assuming the given ticket corpus to have k topics the goal is to factorize $\mathbf{X}$ into two non-negative matrices $\mathbf{C}$ (of dimension $n \times k$) and $\mathbf{D}$ (of dimension $k \times m$). Here $\mathbf{C}$ is the ticket-topic matrix in which each row specifies each ticket as a non-negative combination of topics. It is desired that each topic will be associated with few minimum terms as possible and hence, $\mathbf{D}$ can be assumed to be sparse. Subsequently, we consider the following minimization problem:

$$\underset{\mathbf{C},\mathbf{D} \geq 0}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{X} - \mathbf{CD}\|_F^2 + \alpha \|\mathbf{D}\|_1 + \beta \|\mathbf{C}\|_F^2, \tag{8}$$

where $\|\cdot\|_F$ denotes the square root of the squared sum of all the elements in the matrix (also called Frobenius norm), and $\|\cdot\|_1$ is the L_1-norm. The last two terms in Equation 8 are the regularization terms; and α and β are two regularization parameters which are used to control the strength of regularization. The sparse NMF formulation can be easily solved using the Block Coordinate Descent (BCD) method by breaking the original problem into two sub-problems, the details of which can be found in [14].

We adopt an approximation technique to obtain context vector for test instances by using the fitted NMF model. As the fitted model is $\mathbf{X} = \mathbf{CD}$ we can get $\mathbf{C} = \mathbf{X}\mathbf{D}^T\left(\mathbf{D}\mathbf{D}^T\right)^{-1} = \mathbf{X}\mathbf{F}$, where $\mathbf{F} = \mathbf{D}^T\left(\mathbf{D}\mathbf{D}^T\right)^{-1}$. For new ticket T_{new} we construct the feature vector as $\vec{x}_{new} = (x_1^{new} \cdots x_n^{new})$. Then one can get context vector $\vec{c}_{new} = \vec{x}_{new}\mathbf{F}$. If any of the entries in $\vec{c}_{new}$ is non-positive then it is made to be equal to zero.

5 TICKET SIMILARITY USING CONTEXT SENSITIVE FFDNN

We adopt the following approach for computing similarity scores for a pair of input tickets endowed with their corresponding context information. For a pair of tickets T_1 and T_2 with their feature vector representations $\vec{x}_1$ and $\vec{x}_2$, we obtain their context representations $\vec{cx}_1$ and $\vec{cx}_2$ respectively using the method mentioned above. Given these context enriched representation of two ticket vectors $(\vec{x}_1, \vec{cx}_1)$ and $(\vec{x}_2, \vec{cx}_2)$ we pass them as inputs to the already constructed DNNs, DNNR and DNNRC respectively as shown in Figure 3, which produce outputs (y_1, y_1^c) and (y_2, y_2^c) respectively. Then we compute the similarity between these two tickets using Eqn 4.

$$\begin{aligned} Sim_{DNN}(T_1, T_2) = R(T_1, T_2) &= \cos(y_1, y_2) + \lambda \cos(y_1^c, y_2^c) \\ &= \frac{{y_1}^T y_2}{||y_1||||y_2||} + \lambda * \frac{{y_1^c}^T y_2^c}{||y_1^c||||y_2^c||}, \ \lambda \in [0, 1] \end{aligned} \tag{9}$$

6 EXPERIMENTAL RESULTS

We discuss about the experiments that we conduct on IT maintenance ticket data belonging to Infosys Ltd. We implement these deep learning-based approaches for ticket similarity using Java. Also we use neuroph-core-2.92 library[3] to implement FF-DNN.

6.1 Ticket data

We have used data sets from three domains in ITIL services to validate our methodology. These domains are Sports and Apparel (SnA), Food and Beverages (FnB, in short) and Retail (Ret in short). The data from SnA domain portrays issues related to sports and apparel industry. It has about 5011 tickets. As mentioned earlier, the ticket summaries are preprocessed and are represented by TF*IDF feature vectors. FnB tickets contain information related to food and beverages sector and contains 50666 tickets. The Retail data captures information on issues related to services to customers through multiple channels of distribution. This data includes 14379 tickets which contains data related to services to customers. The details of these data sets are given in Table 1.

Table 1: Statistics of Ticket Data from Different Domain

Domain	Total Tickets	No. of tuples	Input Feature vector dimension	Context vector dimension
SnA	5011	42	2044	400
FnB	50666	82	1584	320
Ret	14379	40	1052	200

6.2 Data Partition

We randomly pick 10% of each data set (tickets) as the test set and 90% of the data set as the training set. Again in the training set out of 90% data we reserve 20% as a sample of M texts for training neural networks as described in Section 4.1, the rest 70% are used to pick similar and dissimilar tickets to compare with those M tickets. That is, for each ticket of the training set $(T_m, m = 1, 2, \ldots, M)$, we take one similar ticket and three dissimilar tickets from the remaining 70% of the data set (with repetition).

[3] can be downloaded from http://neuroph.sourceforge.net/download.html

6.3 Semantic similarity tasks

We apply this ticket similarity framework to three semantic similarity tasks. We also validate the results on these tasks.

Contextual ticket similarity. We adopt a semi-supervised approach for predicting similarity score between two tickets enriched with context information. These scores can be used to compute the semantic similarity of a group of tickets in the same tuple. Recall that on using the DNN-centric method described in Section 5 we can compute the similarity between two tickets. For validation purposes, we need to determine the similarity score between two tickets. But unfortunately, for the current corpus we do not have any similarity score (ground truth) corresponding to a pair of tickets which is readily available. So, we assume two tickets T_1 and T_2 are similar if both the following conditions are satisfied.

- Both T_1 and T_2 belong to same tuple. In other words, T_1 and T_2 have same attributes for the corresponding chosen fixed fields (*e.g.*, category, sub-category, application name and incident type).
- The cosine similarity score between T_1 and T_2 exceeds a threshold value of 0.4, that is, $cos(T_1, T_2) \geq 0.4$.

For validation of our context vector based FFDNN for ticket pair similarity, we conduct the following experiment. We take a set of pairs of similar tickets and a set of pair of dissimilar tickets from test set using the concept of similarity proposed above. Let each such pair $P_i = (T_i^1, T_i^2)$ be fed to the context-based FFDNN from which the similarity score $R_i = \text{Sim}_{DNN}(T_i^1, T_i^2)$ can be computed using Eqn 9. Let the average similarity score for this set of pairs of similar ticket be Λ_{sim}. Similarly, we can compute the similarity score R_i for each pair of dissimilar tickets. For dissimilar set of pairs of tickets, let the average similarity score be Λ_{dis} . We expect that Λ_{sim} should be sufficiently larger than Λ_{dis}.

Ticket Ranking. Using this framework for a given ticket we can find out tickets which are very similar to the former. Towards that, for a given ticket T in tuple τ we compute the DNN-based similarity score for each pair of ticket $\text{Sim}_{DNN}(T, T_i)$ for T_i in the tuple τ. Based on this value we can find top-k similar tickets ($k = 1, 3, 5, 10$) for a given ticket T.

Resolution recommendation for incoming tickets. For a given ticket T once we find top-k similar tickets we pick out the resolution corresponding to each of these k tickets. Then we publish these resolutions as the suggested resolution for the ticket T. We validate the model on resolution recommendation using the test set as follows. We compare the actual resolution of each test ticket with the recommended resolution(s) using semantic similarity [18] score ranging from 0 to 1. In this approach (with SS approach) the similarity of two short sentences is computed based on descriptive features of these sentences. Then we can also compute the average semantic similarity score over all recommended cases. As resolutions are short text probably the SS approach is not not able to reflect the similarity content of a pair of resolutions properly.

6.4 Performance Analysis

The *FF-DNN approach* facilitates for various sizes of the model with many hidden layers, and each layer in turn, contains different number of nodes. However in this paper, we have considered models with two hidden layers. That means, the number of total layers excluding the input layer is $N = 2 + 1 = 3$. We may consider a very deep neural network with many hidden layers. But it may fail to perform better due to poor propagation of activations and gradients [28].

We train the FF-DNN with 100 epochs or iterations. We also train the model with different values of learning rate parameter ϵ (Eqn 10). Out of these trained models, we consider the model for which the learning curve (loss function vs epoch) has been steadily decreasing.

In this experiment, we have taken $\lambda = 0.3$ (Eqn. 4). So, the value of ticket pair similarity varies between 1 and 1.3 for context based FFDNN.

The sizes of DNNR and DNNRC for different data sets are given in Table 2. The notation $2044 - 1000 - 500 - 250$ means the size of input feature vector is 2044 and the first, second hidden layers and the output layer contain 1000, 500 and 250 units respectively. In a given layer, we roughly take half of units that of the previous layer.

Table 2: Size of DNNR and DNNRC

Domain	DNNR Size	DNNRC size
SnA	2044-1000-500-250	400-200-100-50
FnB	1584-800-400-200	320-160-80-40
Ret	1052-500-250-125	200-100-50-25

Table 3: Ticket pair similarity for SnA Data Set

Approach	Λ_{sim}	Λ_{dis}	$(\Lambda_{\text{sim}} - \Lambda_{\text{dis}})$
FFDNN	0.850	0.544	0.306
Context FFDNN	1.116	0.723	0.393

Table 4: Ticket pair similarity for FnB Data Set

Approach	Λ_{sim}	Λ_{dis}	$(\Lambda_{\text{sim}} - \Lambda_{\text{dis}})$
FFDNN	0.669	0.001	0.668
Context FFDNN	1.031	0.409	0.622

Contextual ticket similarity: As explained in section 6.3, we compute Λ_{sim} for a set of pairs of similar tickets and Λ_{dis} for a set of pairs of dissimilar tickets. For our experiment, we have taken the set of 100 pairs of similar tickets and 100 pairs of dissimilar tickets for comparison. These two average similarity scores Λ_{sim} and Λ_{dis} are given in Tables 3, 4, and 5 for the three data sets. The context-based FFDNN performed better for SnA and Ret data as compared to FFDNN. As the domain FnB contains much larger number of tickets than SnA and Ret probably while picking dissimilar tickets we pick up tickets which are much more dissimilar than the given ticket in comparison with the other two domains. Also because of higher number of tickets adequate number of topics could not have been properly captured.

Ticket Ranking: We compare the actual summary of each test ticket with the top k summaries using semantic similarity (with SS

Table 5: Ticket pair similarity for Ret Data Set

Approach	Λ_{sim}	Λ_{dis}	$(\Lambda_{\text{sim}} - \Lambda_{\text{dis}})$
FFDNN	0.904	0.345	0.559
Context FFDNN	1.178	0.475	0.703

Table 6: Ticket Summary Ranking for SnA Data Set

Approach	top@1	top@3	top@5	top@10
FFDNN	0.497	0.531	0.538	0.547
Context FFDNN	0.524	0.559	0.566	0.573
% of improvement	5.4	5.3	5.2	4.8

Table 7: Ticket Summary Ranking for FnB Data Set

Approach	top@1	top@3	top@5	top@10
FFDNN	0.784	0.811	0.814	0.815
Context FFDNN	0.780	0.796	0.819	0.822
% of improvement	−0.5	−1.8	0.6	0.9

Table 8: Ticket Summary Ranking for Ret Data Set

Approach	top@1	top@3	top@5	top@10
FFDNN	0.657	0.718	0.736	0.747
Context FFDNN	0.689	0.762	0.778	0.789
% of improvement	4.9	6.1	5.7	5.6

approach) score which ranges from 0 to 1. Then we compute the average semantic similarity score over all test tickets, see Tables 6, 7, and 8. It can be seen that context-based FFDNN performs better by approximately 5% over the simple FFDNN approach for the data sets SnA and and by 6% for the data set Ret. However, the performance goes down for FnB data set. This might be because of the same reason for contextual similarity task.

Resolution recommendation for incoming tickets: As given in section 6.3, we recommend resolutions for a new ticket. The results are given in Tables 9, 10, and 11. It shows that context based FFDNN has performed marginally better than FFDNN.

To determine the effectiveness of semantic similarity(SS) approach, we manually evaluated recommended resolutions with FFDNN approach for two data sets. We inspect the actual resolution of each ticket and the corresponding recommended resolutions for top 10 case. We use three similarity scores of 0, 0.5 and 1. If the meaning of a pair of actual resolution and recommended language appear to be the same (using meta language oriented informal semantics) then we assign a similarity score of 1 to this pair. If we find that the meaning of the elements of this pair are not exactly same, but there is some match then we provide a score of 0.5 to this pair. Otherwise (in case of the resolutions completely differing in their meaning) we score this pair 0. As before, we calculate the average manual similarity score over all test tickets. Table 12 shows that manual scoring of the similarity is slightly higher than automated evaluation by SS approach.

Table 9: Resolution Recommendation for SnA Data Set

Approach	top@1	top@3	top@5	top@10
FFDNN	0.366	0.459	0.485	0.511
Context FFDNN	0.361	0.458	0.484	0.517

Table 10: Resolution Recommendation for FnB Data Set

Approach	top@1	top@3	top@5	top@10
FFDNN	0.399	0.497	0.529	0.559
Context FFDNN	0.392	0.498	0.532	0.562

Table 11: Resolution Recommendation for Ret Data Set

Approach	top@1	top@3	top@5	top@10
FFDNN	0.513	0.604	0.633	0.662
Context FFDNN	0.513	0.606	0.645	0.672

Table 12: Evaluation of recommended resolutions with FFDNN for top@10

Domain	Manual Evaluation	SS Evaluation
SnA	0.564	0.511
Ret	0.710	0.662

7 CONCLUSIONS

In this work we have integrated context with deep neural network to compute similarity between tickets used in ITIL services. Our learning algorithm seems to improve the performance of similarity computation without contexts using deep network only. Also we could see some improvement in the representation of other similarity tasks. In future we would like examine other context models like PoS tag, word dependency information, word sense, domain ontology and integrate with similar learning framework for performing semantic similarity tasks as discussed in the paper. This will help handle automation of ticketing systems in different stages in addition to automation of several incident management, monitoring and event management tasks.

A GRADIENT DESCENT

We now formulate the gradient descent algorithm in our framework. This formulation is based on the one given in [22]. However, we modify the derivation by taking into account context vectors.

The DNN is trained using gradient-based numerical optimization algorithms [13] because $L(\Omega)$ is differentiable *wrt* Ω. Ω consists of weight matrices W_k and V_k and bias vectors $\mathbf{b}_k$ and $\mathbf{d}_k$, $k = 2, 3, ..., N$. The parameters in (Ω) are updated as

$$\Omega_t = \Omega_{t-1} - \epsilon_t \frac{\partial L(\Omega)}{\partial \Omega}|_{\Omega=\Omega_{t-1}}, \tag{10}$$

where ϵ_t is the learning rate at the t^{th} iteration, Ω_t and Ω_{t-1} are the model parameters at the t^{th} and $(t-1)^{th}$ iteration, respectively.

A part of this derivation was presented in [22]. In this work, we consider individual tickets instead of pair of tickets to compute similarity. Moreover, we consider context vectors also along with the ticket vectors.

Let M be the number of the ticket summaries (T_m). For each of the M tickets, we consider the combination of a similar (positive) ticket summary T_i^{m+} and three dissimilar (negative) ticket summaries $T_j^{m-} : 1 \leq j \leq 3$ for training the DNN. We can denote each m-th combination (T_i^{m+}, T_j^{m-}) as $\mathbf{T_m}$.

Then we can write

$$L(\Omega) = L_1(\Omega) + L_2(\Omega) + \cdots + L_m(\Omega) + \cdots + L_M(\Omega), \tag{11}$$

$$\text{where } L_m(\Omega) = -\log P(T_i^{m+}|T_m), \;\; 1 \leq m \leq M \tag{12}$$

$$\text{and, } \frac{\partial L(\Omega)}{\partial \Omega} = \sum_{m=1}^{M} \frac{\partial L_m(\Omega)}{\partial \Omega} \tag{13}$$

$$\text{On simplifying, } L_m(\Omega) = \log\left(1 + \sum_{T_j^{m-}} exp\left(-\gamma \Delta_j^m\right)\right) \tag{14}$$

where $\Delta_j^m = R(T_m, T_i^{m+}) - R(T_m, T_j^{m-})$

The gradient of the loss function w.r.t the N-th weight matrix W_N is

$$\frac{\partial L_m(\Omega)}{\partial W_N} = \sum_{T_j^{m-}} \alpha_j^m \frac{\partial \Delta_j^m}{\partial W_N} \tag{15}$$

where

$$\frac{\partial \Delta_j^m}{\partial W_N} = \frac{\partial R(T_m, T_i^{m+})}{\partial W_N} - \frac{\partial R(T_m, T_j^{m-})}{\partial W_N} \tag{16}$$

and

$$\alpha_j^m = \frac{-\gamma \; exp(-\gamma \Delta_j^m)}{1 + \sum_{T_{j''}^{m-}} exp(-\gamma \Delta_{j''}^m)} \tag{17}$$

Let y_m and y_i be the outputs of DNNR with T_m and T_i ticket summaries. Let y_m^c and y_i^c be the outputs of DNNRC with context vectors c_m and c_i of T_m and T_i ticket summaries respectively.

$$\begin{aligned}\frac{\partial R(T_m, T_i^m)}{\partial W_N} &= \frac{\partial}{\partial W_N}\left[\frac{y_m^T y_i}{||y_m||||y_i||} + \lambda \frac{{y_m^c}^T y_i^c}{||y_m^c||||y_i^c||}\right] \\ &= \delta_{y_m}{}^{(T_m,T_i)} h_{N-1,T_m}^T + \delta_{y_i}{}^{(T_m,T_i)} h_{N-1,T_i}^T\end{aligned} \tag{18}$$

where,

$$\delta_{y_m}{}^{(T_m,T_i)} = (1 - y_m) \circ (1 + y_m) \circ (uvy_i - avu^3 y_m)$$

$$\delta_{y_i}{}^{(T_m,T_i)} = (1 - y_i) \circ (1 + y_i) \circ (uvy_m - auv^3 y_i)$$

$$a = y_m^T y_i, u = \frac{1}{||y_m||}, v = \frac{1}{||y_i||}$$

The operator '∘' denotes the element-wise multiplication.

The gradient of the loss function w.r.t the N-th weight matrix V_N of DNNRC is

$$\frac{\partial L_m(\Omega)}{\partial V_N} = \sum_{T_j^{m-}} \alpha_j^m \frac{\partial \Delta_j^m}{\partial V_N} \tag{19}$$

where

$$\frac{\partial \Delta_j^m}{\partial V_N} = \frac{\partial R(T_m, T_i^{m+})}{\partial V_N} - \frac{\partial R(T_m, T_j^{m-})}{\partial V_N} \tag{20}$$

$$\begin{aligned}\frac{\partial R(T_m, T_i^m)}{\partial V_N} &= \frac{\partial}{\partial V_N}\left[\frac{y_m^T y_i}{||y_m||||y_i||} + \lambda \frac{{y_m^c}^T y_i^c}{||y_m^c||||y_i^c||}\right] \\ &= \lambda(\delta_{y_m^c}^c{}^{(c_m,c_i)} l_{N-1,c_m}^T + \delta_{y_i^c}^c{}^{(c_m,c_i)} l_{N-1,c_i}^T)\end{aligned} \tag{21}$$

where,

$$\delta_{y_m^c}^c{}^{(c_m,c_i)} = (1 - y_m^c) \circ (1 + y_m^c) \circ (u^c v^c y_i^c - a^c v^c {u^c}^3 y_m^c)$$

$$\delta_{y_i^c}^c{}^{(c_m,c_i)} = (1 - y_i^c) \circ (1 + y_i^c) \circ (u^c v^c y_m^c - a^c u^c {v^c}^3 y_i^c)$$

$$a^c = {y_m^c}^T y_i^c, u^c = \frac{1}{||y_m^c||}, v^c = \frac{1}{||y_i^c||}$$

For hidden layers, we also need to calculate δ for each Δ_j^m. We calculate each δ in the hidden layer k through back propagation as

$$\begin{aligned}\delta_{k,T_m}{}^{(T_m,T_i)} &= (1 + h_{k,T_m}) \circ (1 - h_{k,T_m}) \circ W_{k+1}^T \delta_{k+1,T_m}{}^{(T_m,T_i)} \\ \delta_{k,c_m}^c{}^{(c_m,c_i)} &= (1 + l_{k,c_m}) \circ (1 - l_{k,c_m}) \circ V_{k+1}^T \delta_{k+1,c_m}^c{}^{(c_m,c_i)} \\ \delta_{k,T_i}{}^{(T_m,T_i)} &= (1 + h_{k,T_i}) \circ (1 - h_{k,T_i}) \circ W_{k+1}^T \delta_{k+1,T_i}{}^{(T_m,T_i)} \\ \delta_{k,c_i}^c{}^{(c_m,c_i)} &= (1 + l_{k,c_i}) \circ (1 - l_{k,c_i}) \circ V_{k+1}^T \delta_{k+1,c_i}^c{}^{(c_m,c_i)}\end{aligned} \tag{22}$$

with

$$\delta_{N,T_m}{}^{(T_m,T_i)} = \delta_{y_m}{}^{(T_m,T_i)}, \delta_{N,c_m}^c{}^{(c_m,c_i)} = \delta_{y_m^c}^c{}^{(c_m,c_i)}$$
$$\delta_{N,T_i}{}^{(T_m,T_i)} = \delta_{y_i}{}^{(T_m,T_i)}, \delta_{N,c_i}^c{}^{(c_m,c_i)} = \delta_{y_i^c}^c{}^{(c_m,c_i)}$$

The gradient of the loss function w.r.t the intermediate weight matrix, $W_k, k = 2, 3, \ldots, N-1$, can be computed as

$$\frac{\partial L_m(\Omega)}{\partial W_k} = \sum_{T_j^{m-}} \alpha_j^m \frac{\partial \Delta_j^m}{\partial W_k} \tag{23}$$

where

$$\begin{aligned}\frac{\partial \Delta_j^m}{\partial W_k} &= \frac{\partial R(T_m, T_i^{m+})}{\partial W_k} - \frac{\partial R(T_m, T_j^{m-})}{\partial W_k} \\ &= \delta_{k,T_m}{}^{(T_m,T_i)} h_{k-1,T_m}^T + \delta_{k,T_i}{}^{(T_m,T_i)} h_{k-1,T_i}^T \\ &\quad -\delta_{k,T_m}{}^{(T_m,T_j)} h_{k-1,T_m}^T - \delta_{k,T_j}{}^{(T_m,T_j)} h_{k-1,T_j}^T\end{aligned} \tag{24}$$

The gradient of the loss function w.r.t the intermediate weight matrix, $V_k, k = 2, 3, \ldots, N-1$, can be computed as

$$\frac{\partial L_m(\Omega)}{\partial V_k} = \sum_{T_j^{m-}} \alpha_j^m \frac{\partial \Delta_j^m}{\partial V_k} \tag{25}$$

where

$$\begin{aligned}\frac{\partial \Delta_j^m}{\partial V_k} &= \frac{\partial R(T_m, T_i^{m+})}{\partial V_k} - \frac{\partial R(T_m, T_j^{m-})}{\partial V_k} \\ &= \lambda(\delta_{k,c_m}^{c}{}^{(c_m,c_i)} l_{k-1,c_m}^T + \delta_{k,c_i}^{c}{}^{(c_m,c_i)} l_{k-1,c_i}^T \\ &\quad -\delta_{k,c_m}^{c}{}^{(c_m,c_j)} l_{k-1,c_m}^T - \delta_{k,c_j}^{c}{}^{(c_m,c_j)} l_{k-1,c_j}^T)\end{aligned} \tag{26}$$

Similarly, the gradient of loss function w.r.t bias can be derived. The partial derivation of $R(T_m, T_i^{m+})$ wrt bias b_N and b_k, $k = 2, 3, ..., N-1$ can be derived as:

$$\frac{\partial R(T_m, T_i^m)}{\partial b_N} = \delta_{y_m}{}^{(T_m,T_i)} + \delta_{y_i}{}^{(T_m,T_i)} \tag{27}$$

$$\frac{\partial R(T_m, T_i^m)}{\partial b_k} = \delta_{k,T_m}{}^{(T_m,T_i)} + \delta_{k,T_i}{}^{(T_m,T_i)} \tag{28}$$

The partial derivation of $R(T_m, T_i^{m+})$ wrt bias d_N and d_k, $k = 2, 3, ..., N-1$ can be derived as:

$$\frac{\partial R(T_m, T_i^m)}{\partial d_N} = \delta_{y_m^c}^{c}{}^{(c_m,c_i)} + \delta_{y_i^c}^{c}{}^{(c_m,c_i)} \tag{29}$$

$$\frac{\partial R(T_m, T_i^m)}{\partial d_k} = \delta_{k,c_m}^{c}{}^{(c_m,c_i)} + \delta_{k,c_i}^{c}{}^{(c_m,c_i)} \tag{30}$$

REFERENCES

[1] ACL'15 2015. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1 & 2: Long and Short Papers*. The Association for Computer Linguistics. http://aclweb.org/anthology/P/P15/

[2] Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, and Hal Daumé III. 2016. Learning text pair similarity with context-sensitive autoencoders. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, Vol. 1. 1882–1892.

[3] Chao An, Jiuming Huang, Shoufeng Chang, and Zhijie Huang. 2016. Question Similarity Modeling with Bidirectional Long Short-term Memory Neural Network.. In *Proceedings of IEEE First International Conference on Data Science in Cyberspace*.

[4] Yoshua Bengio. 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* 2, 1 (2009), 1–127.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research* 12 (2011), 2493–2537.

[7] Li Deng, Xiaodong He, and Jianfeng Gao. 2013. Deep stacking networks for information retrieval. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'13, Vancouver, BC, Canada*. 3153–3157.

[8] Cícero Nogueira dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning Hybrid Representations to Retrieve Semantically Equivalent Questions, See [1], 694–699. http://aclweb.org/anthology/P/P15/

[9] Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM'10*. 1139–1148.

[10] Jianfeng Gao, Kristina Toutanova, and Wen-tau Yih. 2011. Clickthrough-based latent semantic models for web search. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'11*. 675–684.

[11] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.

[12] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, ACL'12: Long Papers*. 873–882.

[13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13*. 2333–2338.

[14] Da Kuang, Jaegul Choo, and Haesun Park. 2015. Nonnegative matrix factorization for interactive topic modeling and document clustering. In *Partitional Clustering Algorithms*. Springer, 215–243.

[15] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML'14*. 1188–1196.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature* 521 (2015), 436–444.

[17] Jure Leskovec, Anand Rajaraman, and Jeff Ullman. 2014. *Mining of Massive Datasets* (2nd ed.). Cambridge University Press.

[18] Yuhua Li, David McLean, Zuhair Bandar, James O'Shea, and Keeley A. Crockett. 2006. Sentence Similarity Based on Semantic Nets and Corpus Statistics. *IEEE Trans. Knowl. Data Eng.* 18, 8 (2006), 1138–1150.

[19] Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical Recurrent Neural Network for Document Modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP'15*. 899–907.

[20] Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*. 1367–1375.

[21] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity.. In *AAAI*. 2786–2792.

[22] D. P. Muni, S. Roy, Y. T. Y. J. John L. Chiang, A. J-M Viallet, and N. Budhiraja. 2017. Recommending resolutions of ITIL services tickets using Deep Neural Network. In *Proceedings of the 4th IKDD Conference on Data Science, CODS*.

[23] Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes, See [1], 1793–1803. http://aclweb.org/anthology/P/P15/

[24] S. Roy, D. P. Muni, J-J. Yeung T. Y., N. Budhiraja, and F. Ceiler. 2016. Clustering and Labeling IT Maintenance Tickets. In *Service-Oriented Computing - 14th International Conference, ICSOC 2016, Banff, AB, Canada, Proceedings*. 829–845.

[25] G. Salton and C. Buckley. 1988. Term Weighing Approaches in Automatic Text Retrieval. *Information Processing and Management* (1988).

[26] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval'15*. 373–382.

[27] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 373–382.

[28] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in neural information processing systems*. 2377–2385.

[29] Keith Stevens, W. Philip Kegelmeyer, David Andrzejewski, and David Buttler. 2012. Exploring Topic Coherence over Many Models and Many Topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*. 952–961.