Artificial Intelligence User Support System for SAP Enterprise Resource Planning

Vladimir Vlasov[1], Victoria Chebotareva[1] , Marat Rakhimov[1] and Sergey Kruglikov[2,3]

[1]JSC Sberbank, 620000, Ekaterinburg, Kuibysheva 67, Russia
vevlasov@sberbank.ru
[2]Ural Federal University, Ekaterinburg, Russia
[3]Krasovsky Institute of Mathematics and Mechanics of RAS, Yekaterinburg, Russia
svk@imm.uran.ru

Abstract. An intelligent system for SAP ERP user support is proposed in this paper. It enables automatic replies on users' requests for support, saving time for problem analysis and resolution and improving responsiveness for end users. The system is based on an ensemble of machine learning algorithms of multiclass text classification, providing efficient question understanding, and a special framework for evidence retrieval, providing the best answer derivation.

Keywords: Multi-class Text Classification, Natural Language Processing, Question Answering, Automated User Support, SAP ERP

1 Problem Description

Enterprise resource planning (ERP) is business process management software that allows organization to use a system of integrated applications to manage the business and automate many functions, related to technology, services and human resources.

For a large scale business running complex IT system such as SAP ERP, it is typical to have hundreds of thousands of users' requests annually reported to the help desk verbally by telephone or via specialized help desk systems, such as Service Manager (SM) or Service Desk.

There are usually two types of problems that are reported: errors, arising from incorrect users' actions or system failures and requests for information, advice on a particular user case. Both of these types of problems interrupt a normal course of operations requiring 1-2 days for a problem being solved, leading to additional costs on support team maintenance.

Users' requests are messages in natural language, containing some kind of information about the problem, which is relevant in the users opinion. In fact, types of problem descriptions range from a deep problem analysis conducted by a professional user to less informative and sometimes useless or misleading messages.

The ability to understand human posed questions and to provide relevant answers refers to one of the most challenging AI problems – Question Answering (QA). QA is

a computer science discipline within the fields of information retrieval and natural language processing. In comparison with the open-domain QA systems, such as IBM Watson, designed to challenge human champion in the Jeopardy! Game [1], dealing with various problems of producing sensible answers out of general knowledge: from lexical answer type determination to massively parallel hypothesis scoring, closed-domain QA systems deal with specific knowledge which can be formalized in ontologies.

The problem of implementing machine learning algorithms to help desk automation is described in [2].

Thus building an intelligent user support system is a closed-domain QA problem. In this paper we present a description of such QA system with application to a particular domain of SAP ERP user support, based on machine learning algorithms of text classification.

2 Intelligent user support system architecture

The high-level architecture of the intelligent user support system is presented in figure 1.

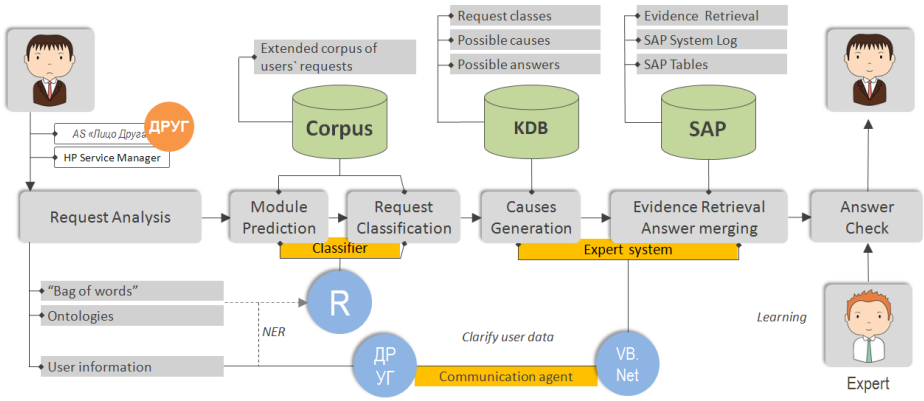

Fig. 1. The high-level architecture of the intelligent user support system for SAP ERP

The main steps in a high-level architecture of the proposed intelligent user support system.

1. Content acquisition including creation, cleaning and extending of a corpus of users' requests.
2. Request analysis including SAP module determination and deep request classification.
3. Knowledge database development in order to map classes of requests on sets of possible causes.
4. Generation of a set of possible causes for the particular type of problem.

5. Gathering evidence from SAP to evaluate hypothesis about candidate causes of problem.
6. Answer merging w.r.t. available information from SAP tables, system log and user information.
7. Answer checked by an expert. If:

- Correct, then the answer sent to user.
- Incorrect, then the request and the correct class[1] added to the training set in R. An algorithm relearned and updated.

3 Request Classification Algorithm

A classification of requests is a standard text classification task performed by various machine learning algorithms providing multi-class classification [3, 4, 5, 6].There were attempts to implement more sophisticated algorithms for this task such as recurrent convolutional neural network [7], but the increase in accuracy was not significant.

Raw data comes from SAP ERP support database of one large Russian company. All requests are primarily divided into six main categories (modules) in accordance with business processes realized in SAP ERP environment. Requests out of each module have been labeled by human experts tby one of the classes from the set of classes (*C*).

Main categories (modules) are the following.

- Purchases and agreements.
- Property and material management.
- Budget planning and execution.
- Real estate management and construction.
- Business trips and hospitality expenses and Payments.
- User Access management.

A class label is a short text of typical SAP ERP user problem: an error message or general user request problem type. Each class is constructed in a manner, that requests attributed to one class, share the same set of possible causes and corresponding answers.

Some class labels examples in the Purchases and Agreements category: «How to execute the request for supply», «Error: applicant is unavailable», «Specification not displayed in agreement»,“no resource found when creating a request for supply”, etc.

Users can express one problem in different ways, for example, by the following examples. Consider a user is faced with a problem “no resource found when creating a request for supply”. Then request can be formulated as: “when creating Request for Supply I cannot select the required resource” or “when creating Request for Supply the program does not find the resource”

[1] New classes of requests potentially can arise.

The number of classes by categories is : $l_1 = 81$, $l_2 = 43$, $l_3 = 26$, $l_4 = 23$, $l_5 = 33$, $l_6 = 9$. Total number of classes is $l = 215$. Each request can be related only to one class. The number of requests in the labeled sample is the following: $m_1 = 4168$, $m_2 = 2824$, $m_3 = 1698$, $m_4 = 867$, $m_5 = 1609$, $m_6 = 1388$. Total number of requests in a labeled sample $m = 12554$.

The key aspect to an efficient classification algorithm is the preparation of the text corpus [2]. Corpus – is a complete collection of texts of user requests, containing raw texts with class labels. Cleaning and extension of a corpus include a sequence of procedures, developed in R via different packages of text mining ("tm"), stemming ("SnowballC"), splitting, and combining data ("plyr"):

1. Conversion to lowercase.
2. Deletion of doubled whitespaces, numbers, punctuation.
3. Deletion of stopwords ("and", "or", "good morning", "thank", "please").
4. Reducing words to their word stems, base form (stemming).
5. N-gram retrieval (n-gram – is a contiguous sequence of n items from a given sequence of text), ("create new request for supply", "insufficient budget").
6. Unification of synonymic constructions

Building a term-document matrix (TDM), containing each request as a vector of numerical attributes, corresponding to tokens encountered inside requests.

In the course of the work, it was concluded that not all stopwords should be removed. The word "not/no" turned out to be meaningful within the framework of our task. And for module "User Access management" prepositions proved to be important. Without them you cannot correctly determine the class.

To reduce the number of words in corpus and to improve accuracy of classification you can combine words into artificial terms. You can combine words in accordance with one of three principles: acronym expansions: "RFS" – "request for supply", synonyms in the sense of the Russian language: "storekeeper" – "warehouse manager" and synonymous words in the context of SAP: "budget indicator red" – "insufficient budget". Such synonyms where chosen manually. Users can formulate one problem in different ways, sometimes with the most unusual words.

A TDM is a sparse matrix, i.e. most of its elements are zeros. To prevent overfitting and ensure classes generalization and interpretability, we should remove sparse terms for each class. Selection of a appropriate sparsity threshold parameter is a subject to optimization. This value of a threshold delivering the best average accuracy is chosen for each class. We use sparsity thresholds 60% for all classes. Also we use TF-IDF to assess the importance of words in requests. But this method did not give a good result in our problem: specific words from small classes remained invaluable. To eliminate this imbalance, the TF-SLF was applied. This method is based on the fact, that the term is important within a category if it occurs in most documents of this category. This approach allows you to estimate the weight of keywords for all classes and reduce the weight of words that are key for several classes.

We train each of the following algorithms on a training set and test it on a test set randomly sampled from each class in proportion of 75%/25%.

1. Naïve Bayes - NB
2. k Nearest Neighbors - kNN
3. Support Vector Machine - SVM
4. Multinominal Logistic Regression or Maximum Entropy – MaxEnt.

A brief description of each algorithm used for classification is presented in the following section.

3.1 Naïve Bayes Algorithm (NB)

NB – is one of the simplest yet effective method of multi-class classification [5] based on Bayesian Rule. The probability of class c given document d is determined by the following equation (1):

$$p(c|d) = \frac{p(d|c) \cdot p(c)}{p(d)} \tag{1}$$

Thus, a class of request is determined by (2):

$$\begin{aligned} c' &= \underset{c \in C}{\operatorname{argmax}}(p(c|d)) \\ &= \underset{c \in C}{\operatorname{argmax}}\big(p(d|c) \cdot p(c)\big) \\ &= \underset{c \in C}{\operatorname{argmax}}\big(p(x_1, x_2, \dots, x_n|c) \cdot p(c)\big) \end{aligned} \tag{2}$$

where $x_1, x_2, \dots, x_n$ – is a vector representation of d ("bag of words" representation).

Making an assumption of feature x_i conditional independence, which is obviously wrong (terms «budget» and «control» more often appear together), but allows joint probability $p(x_1, x_2, \dots, x_n|c)$ to be represented as a product of probabilities $p(x_i|c)$. Then formula (2) takes its final representation (3):

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}}\, p(c_j) \prod_{x \in X} p(x|c) \tag{3}$$

The probabilities in (3) are substituted by their frequency-based estimators: $P(c_j)$ is estimated as a ratio of requests related to class c_j to total number of requests, $p(x|c)$ is estimated as a fraction of times term x appears among all terms in requests of class c_j.

Classification made by NB is a good benchmark for other more sophisticated algorithms.

3.2 K Nearest Neighbors (KNN)

KNN – is an algorithm that returns the class which contains the most similar training request to the one that is being classified. The use of different similarity functions forms different types of kNN. We use the standard Euclidean distance (4).

$$\rho(u, x_i) = \left(\sum_{j=1}^{n} \left| u^j - x_i^j \right|^2 \right)^{1/2} \tag{4}$$

where x_i^j – i-th neighbor of the object u.

For each u, its neighbors from the training sample can be arranged in ascending order w.r.t. ρ. If $c_u^{(i)}$ – is the class of i-th neighbor of object u, then the class of u is determined by the equation (5):

$$c_{kNN} = \underset{c \in C}{\operatorname{argmax}} \sum_{i=1}^{m} [c_u^{(i)} = c] \cdot w(i, u) \tag{5}$$

Choosing $w(i, u)$ one can form various types of kNN algorithm: exponentially weighted NN, Parzen window kNN, etc. [8]. The number k of NN is modelled by leave-one-out cross-validation.

3.3 Support Vector Machine (SVM)

SVM – is the non-probabilistic classification method introduced by V. Vapnik, which constructs a hyperplane in a high-dimensional feature space by empirical risk minimization [8]. SVM – is a binary classification algorithm. Traditional way of performing multiclass classification with SVM is combining several binary "one-against-all" or "one-against-one" SVM classifiers [5]. We have k (by the number of classes) similar "one-against-all" optimization tasks (6):

$$\begin{gathered} \min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ w^T \phi(x_i) + b \geq 1 - \xi_i, if\ y_i = m, \\ w^T \phi(x_i) + b \leq -1 + \xi_i, if\ y_i \neq m, \\ \xi_i \geq 0 \end{gathered} \tag{6}$$

where y_i- class of x_i, ϕ – kernel function. C – penalty parameter.
Thus a class of request is derived by (7):

$$c_{SVM} = \underset{c \in C}{\operatorname{argmax}}((w^c)^T \phi(x) + b^c) \quad (7)$$

As long as in the text classification problem classes are usually linearly separable, SVM with linear kernel should perform well for text categorization [5].

3.4 Maximum Entropy (MaxEnt)

Multinominal logistic regression or maximum entropy [9] – probabilistic log-linear classifier, maximizing log-likelihood of a set of weighted features from input data. The class c of the request x is determined as maximum probability $p(c|x)$ that is computed by (8)

$$p(c|x) = \frac{\exp\left(\sum_{i=1}^{N} w_i f_i(c,x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=1}^{N} w_i f_i(c',x)\right)} \quad (8)$$

where $f_i(c,x)$ – i-th binary feature of request x given class c. This feature takes the value of 1, if a term is present in a request and the class is c, and zero otherwise.

Special regularization methods are also used to fix the problem of overfitting. In our work we used the implementation of MaxEnt algorithm in R written by T. P. Jurka [6, 10].

The results in terms of accuracy and its variation on local tests are presented in Table 1. The results for parameters with the best performance on the test set are reported.

Table 1. Mean and Variation of accuracy on local tests (%)

Support group (SAP module / process)	NB	kNN	SVM	MaxEnt
Purchases and agreements	76.4 (2.1)	85.4 (1.4)	**87.8 (1.9)**	87.1 (2.1)
Property and material management	76.6 (2.5)	86.6 (1.6)	89.5 (2.1)	**90.3 (1.5)**
Budget planning and execution	64.6 (3.4)	77.4 (3.5)	85.6 (1.5)	**87.3 (2.3)**
Real estate management and construction	88.8 (2.9)	91.1 (3.2)	**93.5 (2.9)**	92.4 (2.9)
Business trips and hospitality expenses and Payments	88.3 (1.9)	92.2 (2.1)	**94.5 (1.8)**	93.9 (1.4)
User Access management	82.1 (2.6)	91.9 (2.98)	**95.1 (2.6)**	56.3 (3.6)

As it is shown in table 1, there is no single choice for the type of the classification algorithm. MaxEnt and SVM are the most popular choice. So by far we completed the steps that allowed the proposed system to understand what kind of problem a user was trying to express in his request.

4 Answering the request and error analysis

After defining the problem class the system gives the expert a prepared answer for this class. The answer for class can be unambiguous (for the request “how to get the role of a material responsible person”) or may consist of several recommendations depending on the cause of the problem (for the request “no resource found when creating a request for supply”). In this later case it is necessary to conduct additional analysis. The analysis consists of hypotheses testing for the candidate causes of the problem. Hypotheses testing imply check-listing a set of rigid rules predefined by the expert in order to reduce the number of potential causes and the corresponding answers.

For each class of requests there are certain parameters to retrieve from SAP ERP system and to compare with the values from the user request. In case of missing user parameters in the initial request, the system either looks them up in a system log or asks user to send information.

After the evidence has been retrieved the process of “pruning the trees” of candidate causes is performed and a single answer (perhaps combined of several reasons and recommendations) is given. The answer is then verified by a human expert and in case it is incorrect – reclassified by an expert.

Thus the process of forming a response can be represented as follows:

- Collecting necessary data to test hypotheses about the cause of the problem (from a user request).
- Requesting SAP database for necessary parameters specified for each class.
- Verification of criteria.
- Rejection of incorrect hypotheses.
- Formulation of the final answer.

For example we will take the class from the module "Purchases and agreements": “no resource found when creating a request for supply”. The data required for the hypothesis testing are the number of the resource and the number of request for supply. If these data are present in the message, we can proceed to step 2. In case of the absence of necessary information, the system will send a request to the user or the system looks it up in a system log. When the number of request for supply and resource are obtained, we proceed to testing the hypotheses. In this case there are 3 of them.

1. Expiration date of the resource > date of consumption in the request for supply.
2. Invalid resource number.
3. A resource cannot be used in the request for supply.

With respect to a certain request ID, for each of hypotheses the system retrieves specific parameters (Expiration date of the resource, number of the resource, number of the request for supply) and in case the conditions of the hypothesis are matched, it is regarded as true.

If this hypothesis is confirmed, then the user receives a particular response, such as: “Good day! Select another resource”.

As for incorrectly classified appeals, they are analyzed. The training set and the classification algorithm is updated on weekly basis with regard to the error analysis. The most common reasons of errors in classification of requests are the following:

- Uncommon synonym for the specific term was used in the request. In this case this synonym is added to a term definition.
- Presence of keywords from different classes in one request. Usually this is a problem of long requests, when the user formulates several questions in one request. Those requests are treated individually by an expert.

5 Conclusions

The paper introduces a machine-powered user support system for SAP ERP. It describes the main steps of the process of corpus preparation, problem understanding and answer derivation combined in a simple architecture, providing the system to evolve.

The system is scalable on various problems of closed-domain question answering in different spheres and requires joint efforts of human experts for data preparation, building a knowledge database and text mining techniques and machine learning algorithms of multi-class classification to potentially reach a high-level accuracy on understanding and answering natural language requests.

References

1. Ferrucci, D., Brown, E. et al.: Building Watson: An Overview of the DeepQA Project. AI Magazine, (2010).
2. Logan, D., Kenyon, J.: HELPDESK: Using AI to Improve Customer Service. In Proceedings AAAI-92
3. Bijalwan, V. Kumar V., P. Kumari and Pascual J.: KNN based Machine Learning Approach for Text and Document Mining. In International Journal of Database Theory and Application Vol.7, No.1, pp.61-70, (2014)
4. Jurafsky, D.: Text Classification and Naïve Bayes.
5. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In Proceedings of the European Conference on Machine Learning. Springer, (1998)
6. Nigam,K., Lafferty,J. and McCallum,A.: Using maximum entropy for text classification. IJCAI-99Workshop on Machine Learning for Information Filtering, pp. 61–67, (1999)
7. Lai, S., Xu, L., Liu, K., Zhao,J.: Recurrent Convolutional Neural Networks for Text Classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

8. Cortes,C., Vapnik,V.: Support-vector network. Machine Learning, 20: pp. 273–297, (1995)
9. Jurafsky, D., Martin, J. H.: Speech and Language Processing. Lecture.
10. Jurka, T. P.: Maxent: An R Package for Low-memory Multinomial Logistic Regression with Support for Semi-automated Text Classification.