



## Automated Medical Chatbot

---

Krishnendu Rarhi, Abhishek Mishra and Krishnasis Mandal

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

April 24, 2018

# Automated Medical Chatbot

Krishnendu Rarhi (ORCID: [0000-0002-5794-215X](https://orcid.org/0000-0002-5794-215X)), Abhishek Bhattacharya, Abhishek Mishra,  
Krishnasis Mandal

**Abstract.** This paper aims to present a design for a medical Chatbot that provides diagnosis and remedies based on the symptoms provided to the system. The system will be able to measure the seriousness of the diagnosis and if needed, it will connect the user to a doctor available online.

**Keywords:** Chatbot, health.

## 1 Introduction

A Chatbot is a system that can interact with human users with natural language. The vast amount of information that is available on the internet allows Chatbots to provide accurate and efficient information based on the user's requirements. Chatbots are used in domains like Customer Support, Virtual Assistance, Online Trainings, and Online Reservations and also for general conversations.

The proposed Medical Chatbot can interact with the users, giving them a realistic experience of chatting with a Medical Professional. Our Chatbot can detect human message patterns using AIML (Artificial Intelligence Mark-up Language) which is a mark-up language based on XML to build AI applications [1]. It retrieves keywords from the initial messages to know the possible medical problems that the user has, based on their input.

There are few Medical Chatbots that already exist, but they do not provide users with medication to any illness but connect them with a Medical QA Forum and show them similar questions to their symptoms that doctors may have previously answered.

The system was compared with Health Tap which is a popular Facebook Messenger Chatbot. Our motive is to show that the proposed medical Chatbot could be a better alternative to many already existing Chatbots in the domain of medicine.

## 2 Related Work

Many medical Chatbot designs have been proposed in the past few years which aim to provide the user with medicine recommendation after extracting the illness information from the user messages.

A similar paper “Pharmabot: A Pediatric Generic Medicine Consultant Chatbot” proposed by Benilda Eleonor V. Comendador, Bien Michael B. Francisco, Jefferson S. Medenilla, Sharleen Mae T. Nacion, and Timothy Bryle E. Serac provides a design for a stand-alone medical Chatbot that is implemented using MS Access and Visual C# [2]. For using the proposed design, the user has to navigate using the four options provided by the application. This design aims to work by converting the user input to SQL queries and execute it on MS Access to retrieve the solution to the illness [2].

Also a research paper “MedChatBot: An UMLS based Chatbot for Medical Students” proposed by Hameedullah Kazi, B.S. Chowdhry and Zeesha Memon focuses on a design for an AIML based Medical Chatbot. This Chatbot design is implemented using a JAVA based AIML interpreter called Chatter bean [3]. To use the proposed design, the user has to type a message that should contain the illness name and it detects the illness names using AIML patterns. Once the illness is detected, the Chatbot provides the user about the necessary information about the problem [3].

However the previous proposed designs in the past did not focus in understanding the intensity of the illness that the user is suffering through. Our proposed design aims to ask more questions to the user until it gets confident about the probable illness that the user is suffering through. Also our Chatbot design has the concept of threshold level that helps it to detect the intensity of the problem and connects the user directly to the doctor if it feels that the problem is too serious for the Chatbot to handle.

## 3 Functionality

The system focuses on the messages that the user provides while initiating the conversation. The idea behind this is to detect the preliminary symptoms and the problems that the user may be experiencing.

After the Chatbot has collected enough keywords from the initial messages, it now starts leading the conversation by asking questions to the user and trying to shortlist few diseases that the user may be suffering through.

After the Chatbot has shortlisted the possible diseases that the user may have, now it gives a rank to the possible diseases that the user may be suffering from.

When the list of possible diseases, the Chatbot starts questioning the user about how the user is feeling.

Once it gets ample amount of data it finds the most likely disease that the user may be suffering through.

After the Chatbot has found the disease that the user is suffering through, it measures the seriousness of the illness and acts accordingly either by suggesting remedies and

medication to the user or by connecting the user to the doctor if the measure hits the predetermined threshold value.

## **4 Chatbot Design**

A Chatbot needs to be natural at responding to the user messages and therefore it needs to have a sustainable back-end logic to process user inputs and parameters to generate results.

When a user starts interacting with the Chatbot, the Chatbot engine gets activated and captures every messages provided by the user.

The Chatbot intends to use the AIML approach to reply to the user messages and to get the input that it can feed to the engine.

The engine accepts initial symptoms and extracts keywords from the data. Using the keywords extracted from the symptom, our Chatbot engine shortlists some of the most likely illnesses that the user may be suffering through by matching the keywords with the disease tags. Once the engine has shortlisted diseases that the user may have, now it has to narrow down the selection to only one disease. For this, it sorts the list of selected diseases according to the most number of matches with the keywords and tags [4].

Now the engine checks for top 3 symptoms from each of the shortlisted (sorted) selection until all the 3 symptoms matches with the user and it can know that the user has that particular disease. If any two or more disease qualifies for the same, the user is connected directly to the doctor. Now the engine identifies the disease, our Chatbot asks questions to the user related to the symptoms that commonly happens in that disease.

The engine can measure the seriousness of the problem by assigning a predetermined threshold value against every disease. Every symptom also has a seriousness score against it. The engine maintains an Integer variable where it sums up the scores of the symptoms if it matches with the user input [5]. If the score hits a value greater than or equal to the threshold level, the Chatbot would connect the user with a doctor and provide temporary tips and medication until the doctor is available to chat.

The engine also maintains two String Arrays during the chat session named 'Medication' and 'Remedies'. For every symptom that matches with the user input, the corresponding solution gets stored in the respective arrays.

When the Chatbot has finished checking for all the symptoms it would then provide the user with all the Medication and Remedies it has found during the session.

### **4.1 AIML Component Design**

Using AIML our Chatbot can detect patterns from the user messages and would be able to generate correct and meaningful responses. Using AIML pattern detection, our Chatbot can figure out if the user already knows the disease he is suffering through and just wants solution to that disease [5]. Also our Chatbot can detect if the user is

providing the symptom and wants the Chatbot to find the disease he may be suffering through [6].

#### **4.1.1 Pattern Detection**

Our Chatbot needs to understand patterns in the user messages to figure out the need of the user through predefined tags in the AIML component. Our Chatbot can provide solution based on the symptoms provided by the user and also it can provide the solution if the user inputs any disease name.

In AIML Pattern Tag (<pattern>) contains the predefined pattern that the bot should look for in the messages and if it matches, the particular category is executed.

#### **Understanding Pattern Detection Using Snippets**

```
<pattern>I am not well</pattern>  
<pattern>I am sick</pattern>  
<pattern>I am feeling sick</pattern>
```

If we consider the above three patterns, our Chatbot would look for these pattern utterance in the messages provided by the user and if the pattern matches with the messages, the Chatbot would proceed further.

```
<pattern>Tell me remedies for *</pattern>  
<pattern>what medication I should take for *</pattern>  
<pattern>Tell me cure for *</pattern>
```

In the above snippets, the pattern suggests that the user already knows the problem he is suffering from and he just wants to know the medication and cure for the problem. Here star (\*) refers to the illness the user is talking about. Our Chatbot will check if the initial part of the message is “Tell me remedies for” followed by the disease name (star). Then the disease name (star) would be submitted to the Chatbot engine.

Example-

```
<pattern>Tell me remedies for *</pattern>
```

Suppose the user types-

User: Tell me remedies for Migraine Pain

Here Migraine Pain replaces star (\*) in the pattern – Tell me remedies for \*.

The illness “migraine pain” will be submitted to the Chatbot engine to get remedies for Migraine Pain.

```
<pattern>I am feeling like *</pattern>
<pattern>I am having *</pattern>
<pattern>I am suffering from *</pattern>
```

In the above snippets, the pattern suggests that the user has no idea about the illness he is suffering from and all he can tell the Chatbot is how he feels. Here star (\*) refers to the symptom, the user is talking about. Our Chatbot will check if the initial part of the message is “I am feeling like” followed by the symptom (star). Then the symptom (star) would be submitted to the Chatbot engine.

Example-

```
<pattern>I am suffering from *</pattern>
```

Suppose the user types-

User: I am suffering from headache

Here headache replaces star (\*) in the pattern – I am suffering from \*.

The initial symptom “headache” will be submitted to our Chatbot engine to help the bot proceed further to know the exact illness.

#### 4.1.2 Category Classification

In AIML we need to have different categories defined to let the Chatbot differentiate one action from the other. In our Chatbot we focus on two main categories-

- User wants to retrieve solution by providing the name of the disease -

```
<category>
<pattern>Tell me remedies for *</pattern>
<pattern>What medicine should I take for *</pattern>
<pattern>Tell me solutions for *</pattern>
</category>
```

- User wants the Chatbot to find out the disease and then provide the solution -

```
<category>
<pattern>I am feeling like *</pattern>
<pattern>I am having *</pattern>
<pattern>I am suffering from *</pattern>
</category>
```

### 4.1.3 Connecting To The Engine

We assume that our Chatbot engine is setup on a web server and our Chatbot can interact with it using REST APIs. Our Chatbot would call the REST API of the server using GET Method. Our server is designed to generate response in XML format [7].

AIML provides us with SRAIX (<sraix>) tag that helps our Chatbot to call the GET API URL and also parse the results that it receives.

#### Understanding Engine Connection Using Snippets

```
<category>
<pattern>I am feeling like *</pattern>
<pattern>I am having *</pattern>
<pattern>I am suffering from *</pattern>
<template>
<sraix service="XML" hint="code/response">http://chatbot? param=<star/></sraix>
</template>
</category>
```

In the above code snippet, we use the template tag to display response to the user and we use the sraix tag to pass the value of Star (\*) to the URL parameter.

## 4.2 Engine Design

The Chatbot engine is the main back-end logic of our chat Chatbot that accepts input using the Get method through the Web Api [7]. The Chatbot engine interacts with the XML file that contains all the medical data written by the medical professionals. The Chatbot engine accepts symptom and then interacts with the user to find the illness that the user may be suffering from. It can also accept an illness name and suggest cure for the same. It maintains sessions to make the data persists between different API calls [7].

### 4.2.1 Medical Data Storage Design

The data regarding Diseases, Symptoms and Remedies needs to be stored in an organized manner for making it easier for the engine to access it. Our Chatbot stores data in XML format. The idea is that the Medical Professionals would write this data and feed this data to our Chatbot and the Chatbot engine would interact with this data. We have separate divisions for every disease that is present in our records. The data is properly organized and is compatible to be parsed by standard XML parsers [8].

**Table 1.** Tabular Design Of A Disease Record.

NAME OF THE DISEASE/ILLNESS
TAGS:

Threshold
Symptoms
Score= Value
Symptom Check Question
Recommended Medication <ul style="list-style-type: none"> <li>• Med 1</li> <li>• Med 2</li> <li>• etc.</li> </ul>
Recommended Remedy <ul style="list-style-type: none"> <li>• Precaution</li> <li>• Remedy</li> <li>• etc.</li> </ul>

Xml Representation For A Disease Record.

```

<disease name="Migraine">
<tag>Headache</tag>
<tag>Head</tag>
<tag>Eye</tag>
<tag>Pain</tag>
<tag>ache</tag>
<threshold>5</threshold>
<symptom>
<score>2</score>
<question>Do you experience pain in your eyes?</question>
<solution>
<medicine>ibuprofen</medicine>
<medicine>Aspirin</medicine>
<remedy>Stay away from light. Close your eyes and
relax.</remedy>
<remedy>Wash your eyes with cold water</remedy>
</solution>
</symptom>
<symptom>
<score>1</score>
<question>Do you feel like vomiting?</question>
<solution>
<medicine>ibuprofen</medicine>
<medicine>Aspirin</medicine>
<remedy>Drink warm water.</remedy>
<remedy>Stay in a cold place.</remedy>
</solution>
</symptom>
</disease>

```



#### 4.2.2 Shortlisting Diseases

The user provides the Chatbot with symptoms that he is experiencing, the AIML component sends the symptom to the Chatbot engine using Web Api. The symptom is a sentence that is sent in the form of a String [8].

The Chatbot engine shortlists diseases using input string in following ways:-

a. Removing Stop-Words and Punctuation

The engine searches for predefined stop-words and Punctuation marks in the sentence. The stop-word and the Punctuation marks are removed by replacing them with a Null character.

b. Tokenizing Words from the Sentence

Now as the sentence is free from stop-words and punctuation, the sentence is word-tokenized and stored in an Array. It is done by using the space tokenization technique.

c. Assigning Scores to the Diseases

After the engine gets an Array of the Keywords present in the Symptoms, then it needs to assign a Score to all the Diseases based on the relation between Keywords and Tags [9]. The engine iterates a loop for each of the Diseases and for every Tag that matches with the keyword array, it adds one (+1) to the score of that disease.

Formula to calculate Disease Score.

Disease Score= Number of Tags that matches with the Keyword Array
---

d. Disqualifying Diseases

The engine now has a list of diseases with scores allotted to each one of them. The engine would now remove all the diseases from the list with score = 0. This simply means that the Tags of the diseases that does not match with any of the keywords present are disqualified.

e. Sorting the list

The engine would now sort the list in descending order based on the scores allotted to each of the diseases [9].

#### 4.2.3 Detecting Disease

The engine may have shortlisted several diseases by considering the initial symptoms provided by the user. Now the engine has to determine the exact disease that the user must most likely be suffering from in order to suggest remedies to the user. The engine does this by iterating through the sorted list of diseases and asking question to the user from each of the diseases [10].

It is done in the following way:

- a. Our Chatbot asks the first question from each of the diseases in our list to the user. If the user accepts that the symptom matches with his illness, the engine adds one (+1) to the score of the disease.
- b. Similarly the engine does this for the other two question of all the diseases. Now the Chatbot has asked the user the top 3 question form each of the shortlisted diseases.
- c. Now the engine selects the disease with the highest score.
- d. If there are multiple diseases with the same highest score, the engine repeats the above process for few more questions until the last question of any one of the diseases gets encountered.
- e. If the engine successfully manages to find a single disease with the highest score, it's most likely the disease the user may be suffering through. But if the engine still can't find the disease, our Chatbot connects the user directly to the doctor.

#### **4.2.4 Consultation Process**

Once the engine has successfully identified the disease that the user may be suffering from, our Chatbot needs to talk to the user and collect the problems he is suffering from and then accordingly provide the solution based on the problems. To carry out this process, the Chatbot engine needs to maintain two different arrays for medication and remedies. Also an integer variable is used to maintain the seriousness score.

The consultation process takes place in the following manner -

##### **a. Questioning Symptoms**

The Chatbot questions the user about the symptoms from the Disease's XML record. The engine then adds up the score of the symptom to the seriousness score of the disease for every symptom that matches with the user's problem [10]. If the symptom matches with the user's condition, the respective medication fed in that particular symptom gets added in the medication array and the similar thing happens for the remedies array.

##### **b. Threshold Checking**

Sometimes, it may happen that the problem is too serious for the Chatbot to handle. In this case the Chatbot would connect the user directly with the doctor and also provide the doctor with the chat history of the user. By the time the doctor is available to chat, the user is provided by the first-aid solution. To trigger this process, the seriousness score should hit or rise above the threshold level [11].

Formula to check Threshold Level.

$$H = \Sigma (\text{Score}) / \text{Threshold Value}$$

H = It is the decision parameter which is used to check if the threshold level was hit.

$\Sigma$  (Score) = It is the total score of all the symptoms that the user claims to experience.

Threshold Value = It is the upper limit value until which the Chatbot can handle.

Condition for triggering= if  $H \geq 1$  the Chatbot connects the user to the doctor.

c. Providing Solution

After all the symptoms of the particular disease has been questioned to the use, the consultation session is said to be complete. The engine now uses both the medication and remedy array to form a combined solution that could be given to the user. The engine prepares the solution in a XML format. The AIML component uses the Template (<template>) Tag and the SRAIX (<sraix>) Tag to parse the XML response and show it to the user [12].

## 5 Evaluation & Comparison

We will evaluate our Chatbot design using General Word Percentage (GWP) analysis and combining the result with terminology detection test analysis that would show us an average percentage about how many time our Chatbot detects medical terminologies with the increase in general non-medical terminology as compared to the other Chatbots [12].

We are comparing our Medical Chatbot with HealthTap, which is a popular medical Chatbot on Facebook. Through this comparison we aim to demonstrate that our Medical Chatbot can provide solutions to the diseases which the HealthTap Chatbot cannot provide. As our Chatbot is in the design phase, all the features and the chats used in this comparison is expected to happen if our Chatbot design works as expected.

### 5.1 Terminology Detection Test Using General Word Percentage Approach

Using General Word Percentage analysis we would get a ratio of number of unrelated words used over the total number of words in a message [12]. This would give us an idea about how our Chatbot could deal with unrelated words mixed with medical terminologies. We would take some sample corpus to do this test. We would check whether our Chatbot detects medical terminologies messages for our sample corpus.

Sample Corpus:

1. Hello Chatbot! I am feeling good today! Also I am having a headache.
2. I can't handle this headache, also I feel like vomiting. My digestion is not okay.
3. Is brain cancer curable? I think I am suffering from it.
4. I think I have cancer. I am feeling like vomiting. I want remedies for typhoid.
5. I am feeling like vomiting.

**Table 2.** Illustrating Term Detection Ratio.

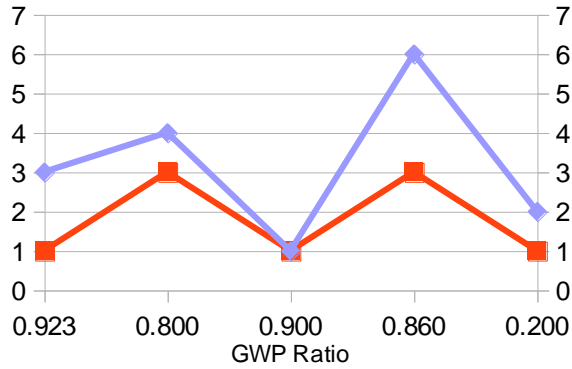
Corpus Number	GWP Ratio	Terms Present	Terms Detected	Term Detection Ratio
1	0.923	1	2	0.500
2	0.800	3	1	0.330
3	0.900	1	0	0.000
4	0.860	3	3	1.000
5	0.200	1	1	1.000
<b>Average:</b>	<b>0.736</b>	<b>1.8</b>	<b>1.4</b>	<b>0.566</b>

#### Ratio Calculation

GWP Ratio= Total Number Of Unrelated Medical Words / Total Number Of Words

Term Detection Ratio= Term Detected / Term Present

**Figure 1.** Visual Representation Of GWP Ratio.



We observe that with an average 73.6% of the unrelated terms present in our sample messages, our Chatbot performed Terminology detection with 56.6% of accuracy. We can observe through the above table (Table 2) and Figure 1 that with the decrease in GWP and the increase in the terms in a message, our Chatbot performed efficiently and was successfully able to detect correct medical terminologies from the message. However the efficiency of terminology detection would depend on the user's typing habit and preciseness of the messages.

## 5.2 Demonstrating a Sample Chat

HealthTap -

**Model 2.** Chat sample of a conversation with HealthTap.

User: Hello
HealthTap: Hi there, let's get your question sent! Enter at least 3 words, but no more than 200 characters.
User: I am having an Headache
HealthTap: Our Doctors have answered similar questions. See if these helps you. <list of questions and answers>
HealthTap: If you don't find your answer in the above cards,

type “send” and we will send your query to a doctor and would inform you when your query is Answered.

### Our Chatbot -

#### Model 3. Chat sample of a conversation with Our Chatbot.

User: Hello

Chatbot: Hi there, tell me how can I help you?

User: I am having an Headache

Chatbot: Are you having pain in your eyes?

User: Yes

Chatbot: Do you feel like vomiting?

User: yes I do

Chatbot: I think you are most likely having a migraine attack

Chatbot: Taking “ibuprofen” 2 tablets would reduce pain in you eyes. Also take “aspirin” 20ml to help you tackle vomiting. Do take a nap and don't forget to wash your eyes with luke warm water. Avoid using digital screen until you feel better.

### **5.3 Understanding The Difference**

- a. HealthTap is a medical Chatbot that allows user to enter the symptom that they feel. The symptom should be in form of a sentence. The sentence contains more than 3 words and not more than 200 words.
- b. HealthTap uses the input sentence to extract keywords and find related answers in their records.
- c. HealthTap does not provide the user with the cure for their problems but it just shows some previously answered question by the doctors related to the symptoms provided.

- d. Our medical Chatbot also accepts symptom in the form of the sentence and it has no restriction of the maximum limit of the words that can be entered.
- e. Our medical Chatbot aims to identify the disease that the user may be suffering through and then it provides the solution directly to the user.
- f. Our medical Chatbot understands the seriousness of the symptom that the user complains about and then it connects the user to the doctor if necessary.
- g. Our medical Chatbot is better than HealthTap in understanding the user symptom, providing the solution with ease and also connecting the user with the doctor if it thinks that the problem is serious to handle.

## **6 Conclusion**

Our Medical Chatbot will have a great impact on the life of its users. It would provide them the advantage of carrying a virtual Doctor in their pockets. It would also give them the freedom to consult a doctor 24/7 and also can get a real doctor's advice if needed. This can be a most popular tool for people with busy schedule as they won't have to hamper their schedule to consult a doctor for minor health queries. This would also be a tool with high utility among elderly and physically disabled people as this can help them get solutions to all their health related issue at their fingertips.

We would bring Doctors and Medical Professionals to our platform to feed the medical data into our records and also to chat with our users when required. Having lots of medical data would make our Chatbot function more efficiently and accurately.

Our Chatbot is in a design phase right now. We would be implementing the whole design into code very soon and we plan to launch it in the next few months. We are implementing the Chatbot engine in JAVA and using Pandorobot as the AIML platform.

We also plan to design and make a device that people could wear and let it collect their important body readings and health data. We would make the device using raspberry pi and would make it compatible with our Chatbot and many others services as well using APIs.

The idea behind comparing our Chatbot with HealthTap was not to show that our Chatbot was better than HealthTap in anyways, but to show that it was different from any Medical Chatbot present till date.

## **7 References**

1. Abu Shawar, B.A., Atwell, E. and Roberts, A. (2005) FAQchat as in Information Retrieval system. In: Human Language Technologies as a Challenge for Computer Science and Linguistics: Proceedings of the 2nd Language and Technology Conference. 2nd Language & Technology Conference, April, 21-23, 2005, Poznań, Poland . Poznań : Wydawnictwo

Poznańskie : with co-operation of Fundacja Uniwersytetu im. A. Mickiewicza , pp. 274-278. ISBN 9788371773419.

2. Comendador, B. E., Francisco, B. M., Medenilla, J. S., Nacion, S. M., & Serac, T. B. (2015). Pharmabot: A Pediatric Generic Medicine Consultant Chatbot. *Journal of Automation and Control Engineering*, 3(2), 137-140. doi:10.12720/joace.3.2.137-140
3. Kazi, Hameedullah & S. Chowdhry, B & Memon, Zeesha. (2012). MedChatBot: An UMLS based Chatbot for Medical Students. *International Journal of Computer Applications*. 55. 1-5. 10.5120/8844-2886.
4. Shawar, BA and Atwell, E (2002) A comparison between Alice and Elizabeth Chatbot systems. University of Leeds, School of Computing research report 2002.19 .
5. Abu Shawar, BA and Atwell, ES (2004) An Arabic Chatbot giving answers from the Qur'an. In: Bel, B and Marlien, I, (eds.) *Proceedings of TALN04: XI Conference sur le Traitement Automatique des Langues Naturelles. TALN04: XI Conference sur le Traitement Automatique des Langues Naturelles, 19-22 April 2004, Fez, Morocco. ATALA* , 197 - 202. ISBN2-9518233-5-5.
6. *Informatica* 31 (2007) 249-268 249 Supervised Machine Learning: A Review of Classification Techniques S. B. Kotsiantis Department of Computer Science and Technology University of Peloponnese, Greece End of Karaiskaki, 22100 , Tripolis GR.
7. TutorBot: An Application AIML-Based for Web-Learning.” *Advanced Technology for Learning (Discontinued)* 2005, ACTA Press, Jan. 2000,
8. A. S, John D (2015) Survey on Chatbot Design Techniques in Speech Conversation Systems. *International Journal of Advanced Computer Science and Applications*. doi: 10.14569/ijacsa.2015.060712
9. De Gasperis, G. (2010). Building an AIML Chatter Bot Knowledge-Base Starting from a FAQ and a Glossary. *Journal of e-Learning and Knowledge Society*, 6(2), 75-83. Italian e-Learning Association. Retrieved November 20, 2017
10. Kurian, Ciji Pearl and George, V I and Bhat, Jayadev and Aithal, Radhakrishna S (2006) ANFIS Model for the Time Series Prediction of Interior Daylight Illuminance. *International Journal on Artificial Intelligence and Machine Learning*, 6 (3). pp. 35-40. ISSN 1687-4854
11. VLDB '99 Proceedings of the 25th International Conference on Very Large Data Bases table of contents Editors : Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez,m, Stanley B. Zdonik, Michael L. Brodie Pages 302-314 Publication Date 1999-09-07 (yyyy-mm-dd) Publisher Morgan Kaufmann Publishers Inc. San Francisco, CA, USA ©1999 ISBN: 1-55860-615-7 Conference VLDBVery Large Data Bases
12. Denoyer L., Gallinari P. (2007) The Wikipedia XML Corpus. In: Fuhr N., Lalmas M., Trotman A. (eds) *Comparative Evaluation of XML Information Retrieval Systems. INEX 2006. Lecture Notes in Computer Science*, vol 4518. Springer, Berlin, Heidelberg