

Emil, the Teacher Bot

WebPortal GUI, Text Processing,
Machine Learning, and Active Learning

Vincenzo Tursi

Vincenzo.Tursi@knime.com

Kathrin Melcher

Kathrin.Melcher@knime.com

Phil Winters

Phil.Winters@knime.com

Rosaria Silipo

Rosaria.Silipo@knime.com



Summary

The list of KNIME use cases has a new addition: how to make a bot, more specifically, how to make a teacher bot. The bot is charged with the task of helping people new to KNIME Analytics Platform to find answers to their initial questions from a large number of technical tutorials.

There is a series of steps involved in building a bot. First of all, a web based user interface to ask the question; next, a set of text processing NLP functions to parse it; and a machine learning model to associate the right group of tutorials to the question that was asked.

For this particular project, we also decided to use an unlabeled dataset to train a supervised machine learning algorithm. This introduced one extra problem to our project - that of adopting an active learning strategy in order to train a supervised algorithm on an unlabeled dataset.

In the next sections, we will describe how we brought together the following data science procedures and resources from [KNIME Analytics Platform](#) to assemble Emil, the teacher bot.

- **A web based user interface**
- **A set of text processing (NLP) functions**
- **A machine learning (ML) model**
- **An active learning procedure to train a supervised ML model on an unlabeled dataset**

The workflows are available on the KNIME EXAMPLES server under *50_Applications/33_Emil_the_TeacherBot*.

Table of Contents

Emil, the Teacher Bot	0
WebPortal GUI, Text Processing, Machine Learning, and Active Learning	0
Summary.....	1
1. The Deployment Workflow	3
Puzzling Together the Pieces	3
Ask the Question: the Web UI.....	4
Understand the Question: Text Processing	5
Find the Right Answer: the Brain	5
Am I Right? The “Feedback” Page	6
Display the Answers: the “Resources” Page	7
The Final Deployment Workflow	8
2. Keyword Extraction for Understanding	8
3. The Ontology.....	10
Educational Resources on the KNIME Website.....	10
Data Science and Machine Learning in a Less than Perfect World ..	11
The Final Ontology	13
Don’t Get Too Attached!	Error! Bookmark not defined.
4. Training the Brain: the Active Learning Cycle.....	15
The Classification Problem	15
The Active Learning Cycle	16
Class Assignment.....	18
GUI for Manual Relabeling	18
The Dreadful Questions.....	20
Active Learning Summary.....	23
5. Conclusions.....	23
References.....	24

1. The Deployment Workflow



Hi! My name is Emil and I am a Teacher Bot.

I was built to point you to the right training material to answer your early questions on how to use [KNIME](#) products.

By the way, I was built entirely by using KNIME. So, I should know where the right answers are to be found in the midst of all the tutorials, videos, blog posts, whitepapers, example workflows, and more, which are available out there.

Puzzling Together the Pieces

It was not so hard to build me. All you need is:

- a user interface - web or speech based - for you to ask your question
- a text parser for me to understand your question
- a brain to find the right training material to answer your question
- a user interface to return the answer
- a feedback mechanism to state whether or not my answer was of any help – which is a nice to have - but not totally necessary.

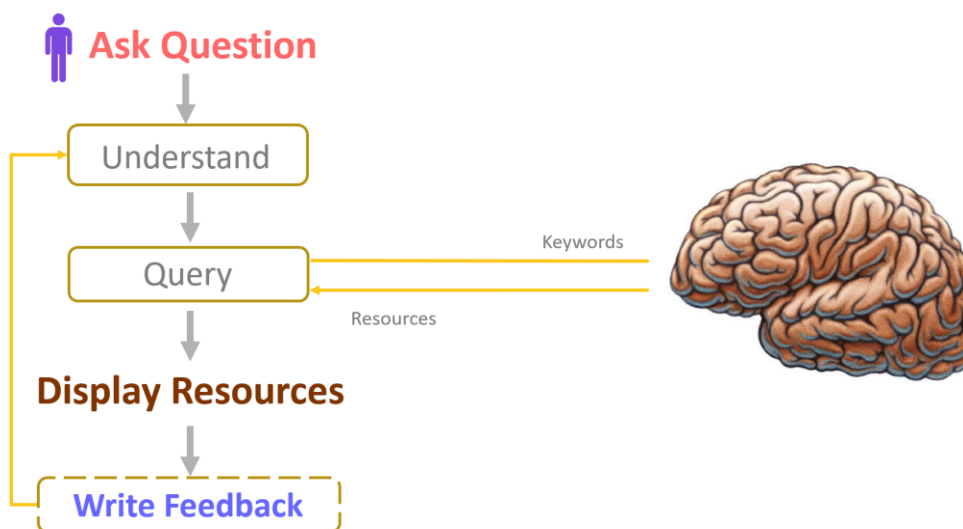


Figure 1.

Emil, the Teacher Bot. Here is what you need to build one: a user interface for questions and answers, text processing to parse the question, a machine learning model to find the right resources, and optionally a feedback mechanism.

Translating these steps into data science terms and KNIME tools, you need:

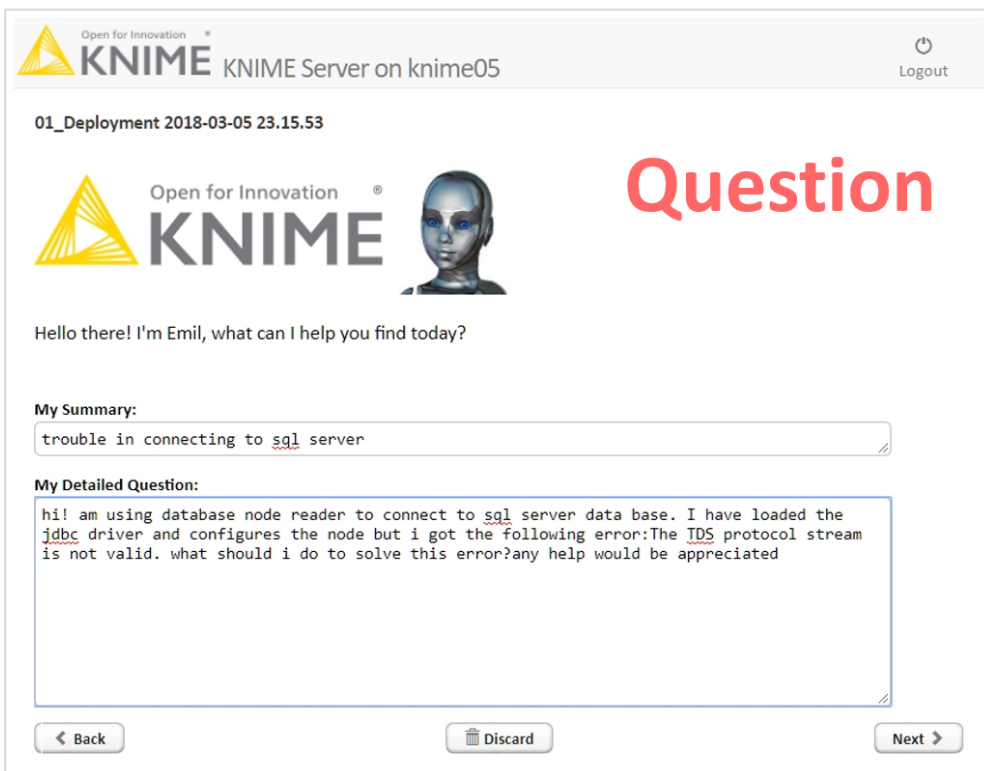
- a web page to ask the question
- some text processing functions
- a trained machine learning model
- a new web page to place the answer
- and, optionally, a feedback logic somewhere in the flow.

Let's start the assembly line.

Ask the Question: the Web UI

[KNIME WebPortal](#) provides the web based user interface (UI), required for the question and answers.

A slick minimalist [Google-like](#) UI was adopted to handle the question; not so much a design choice, but rather due to time and technical competence constraints. Indeed, the web UI shows just an image of me, Emil, at the top, followed by a simple greeting, and most importantly the space for your question.



The screenshot shows the KNIME WebPortal interface. At the top, it says "KNIME Server on knime05" with a "Logout" button. Below this, it displays the date and time "01_Deployment 2018-03-05 23.15.53". The main content area features the KNIME logo, a small image of a person (Emil), and the word "Question" in large red text. A greeting says "Hello there! I'm Emil, what can I help you find today?". There are two text input fields: "My Summary:" with the text "trouble in connecting to sql server" and "My Detailed Question:" with the text "hi! am using database node reader to connect to sql server data base. I have loaded the jdbc driver and configures the node but i got the following error:The TDS protocol stream is not valid. what should i do to solve this error?any help would be appreciated". At the bottom, there are three buttons: "Back", "Discard", and "Next".

Figure 2.

Emil's web based UI, where you can ask a KNIME related question. The question shown here refers to a database connection via JDBC driver.

This web page was obtained via a wrapped metanode, which contains:

- a Text Output node to display the logo, my picture, and the greetings;
- a String Input node to collect a short summary of the question;

- a second String Input node to collect the extended text of the question.

Note. The two String Input nodes produce two text boxes of different sizes. The larger size comes from the enabled option “Multi-line” vs. “Single-line” in the configuration window.

Understand the Question: Text Processing

You have written the question and a summary of the question. Now I need to understand it.

This part is handled by text processing, which includes general text cleaning - such as applying the stop word filter, punctuation erasure, dictionary based tagging and stemming - and a keyword extraction procedure. The keyword extraction procedure reduces your question to the most meaningful words and helps me understand you better. A [chi square keyword extraction algorithm](#) was chosen to take care of this.

Find the Right Answer: the Brain

My ultimate goal is to provide you with the one and only web tutorial that solves your problem. Well, this is hardly possible. Even though you are a beginner to using KNIME, you are likely to ask questions that will refer to topics from two, three, or even four different tutorials. Therefore, I think it is better to provide you with a list of possible helpful tutorials, rather than a single one.

Going further, I think it is best to identify the areas of expertise touched on by your question and, subsequently identify the most relevant tutorial resources relevant to each of these areas. This is what my brain should do: identify the areas of expertise and identify the list of the most relevant articles.

Thus, my brain must consist of a machine learning model (a random forest) and a similarity search feature. The machine learning model is trained to identify the areas of expertise and the similarity search (based on N-gram Tversky/Tanimoto distance) detects the list of the most relevant articles within each area.

In the whole project, defining the training problem, creating a labeled dataset, and building a class ontology were not minor details. Indeed, the lack of labeled data forced the procedure to rely, at least partially, on human corrections via active learning.

Am I Right? The “Feedback” Page

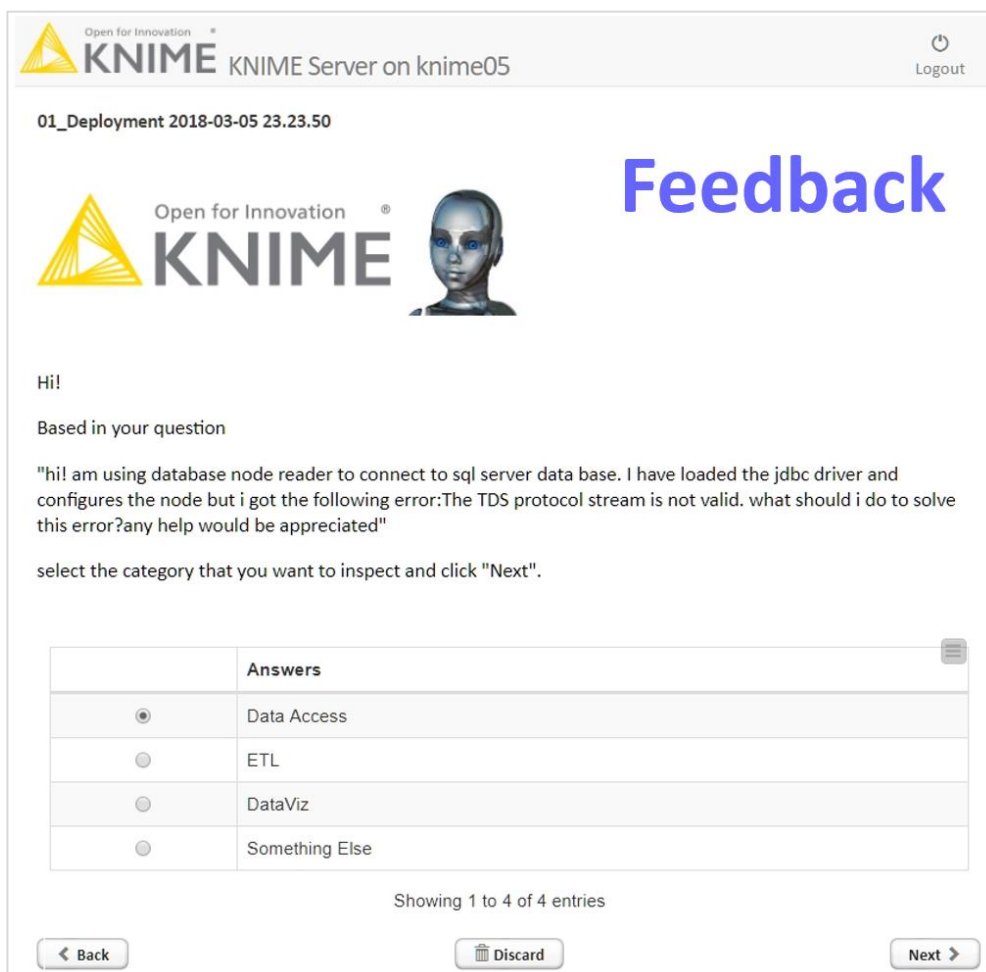
Remember? I would appreciate it, if you could give me some feedback. If my proposed areas of expertise (categories) are wrong, my proposed web resources are also wrong! It would be great to have feedback on that.

Somewhere along our conversation, I would like to ask you whether any of the three top proposed areas of expertise proved to be of any help answering your question.

This is the task of the “Feedback” page. Just like the “Question” page, the “Feedback” page is based on a wrapped metanode, including:

- a Text Output node for logo, image, and repeated question,
- a Value Selection Quickform node to select which, if any, of the proposed categories was helpful.

The last option in the list, called “Something Else”, implies the lack of acceptable answers and refers to a terrible job done on my side.



01_Deployment 2018-03-05 23.23.50

Open for Innovation[®] **KNIME**

Feedback

Hi!

Based in your question

"hi! am using database node reader to connect to sql server data base. I have loaded the jdbc driver and configures the node but i got the following error:The TDS protocol stream is not valid. what should i do to solve this error?any help would be appreciated"

select the category that you want to inspect and click "Next".

	Answers
<input checked="" type="radio"/>	Data Access
<input type="radio"/>	ETL
<input type="radio"/>	DataViz
<input type="radio"/>	Something Else

Showing 1 to 4 of 4 entries

[< Back](#) [Discard](#) [Next >](#)

Figure 3.

The Feedback page. Here you can select one of the proposed categories and then click “Next”. If the following list of top resources is useful, you can terminate the conversation. If the list of top resources is not useful, you can click “Back” to return to this page and select another category. If our conversation is ended, I assume that the last chosen category is your feedback. If option “Something Else” is chosen, I interpret this as an invite to learn more and do better next time. This feedback phase could of course be skipped and the category selection could be hidden from the UI flow.

Note. This feedback page could be omitted. I can also do without your feedback, but - especially at the beginning - your 2 seconds of feedback are of invaluable help to speed up the learning curve.

Display the Answers: the “Resources” Page

Based on the categories identified by my brain and possibly confirmed in the “Feedback” page, I can come up with a list of web resources.

Please, check if any of them help answer your question or if they are useful to simply learn something more about the topic.

If yes, just click “Next” at the end of the page.

If not, use the “Back” button to go back to the “Feedback” page and select a new category for new web resources.

If you have already done that multiple times and you are still looking for an answer to your question, please talk directly to one of the humans who assembled me and use the “Send email” button.

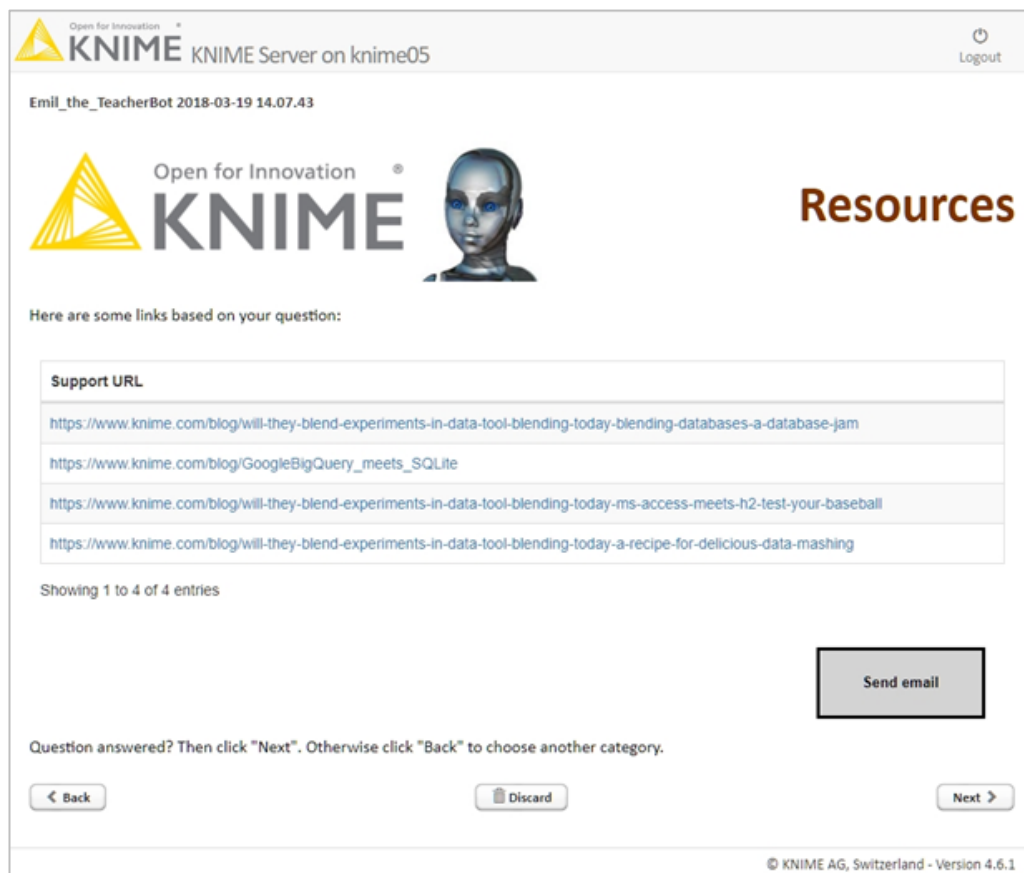


Figure 4.

The UI page with the proposed resources. After selecting a category in the Feedback page, you are presented with the top resources in that area. If you do not find an answer in any of these links, just click “Back” and select a new category. If you do, click “Next”. This terminates our conversation. If you have not found an answer and you have exhausted all suggested categories, please send an email to my creators.

This page is created by using a wrapped metanode again, which contains:

- a Table Editor node to display the list of links
- a Generic Javascript View node to create the “Send email” button.

The Final Deployment Workflow

At the end of the assembly line, there is me, Emil, your Teacher Bot.

The workflow used to assemble me is shown in the figure below and can be found on the EXAMPLES server under:

[50_Applications/33_Emil_the_TeacherBot/Emil_the_TeacherBot](#).

You might recognize the metanodes at the origin of the UI pages.

Hi! My name is Emil and I am a KNIME workflow.

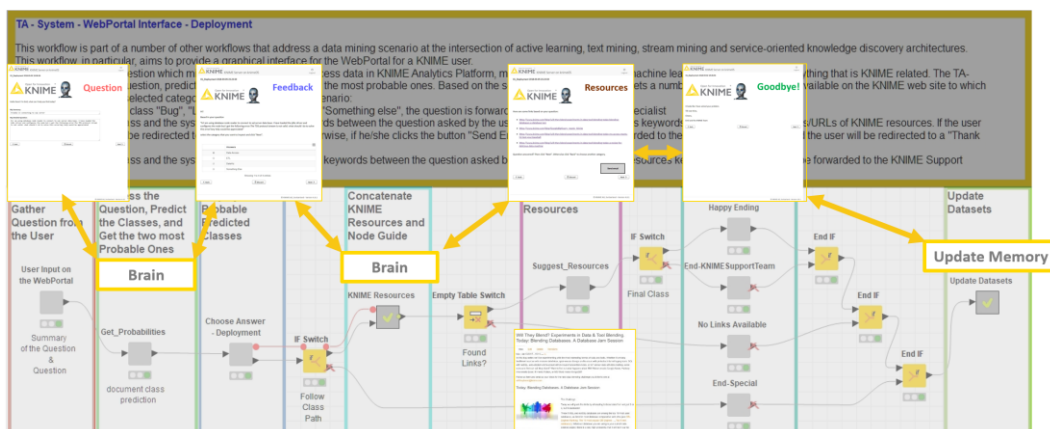


Figure 5.

This is the workflow behind Emil the teacher bot. This workflow reads your question, parses it, queries a machine learning model for the answers, and returns the answers back to you via a KNIME WebPortal web based UI. It also includes a feedback mechanism. This workflow is available on the EXAMPLES server under: [50_Applications / 33_Emil_the_TeacherBot](#)

2. Keyword Extraction for Understanding

A part of the deployment workflow is dedicated to understanding. This is obviously a crucial step, because if I cannot understand your question I am unlikely to be able to answer it.

Understanding consists mainly of text processing operations: text cleaning, Part-Of-Speech (POS) tagging, tagging of special words, lemmatization, and finally keyword extraction - and keyword extraction in particular.

Keywords are routinely used for many purposes, like retrieving documents during a web search or summarizing documents for indexing.

Keywords are the smallest units that can summarize the content of a document and they are often used to pin down the most relevant information in a text.

Automatic keyword extraction methods are wide spread in [Information Retrieval \(IR\)](#) systems, [Natural Language Processing \(NLP\)](#) applications, [Search Engine Optimization \(SEO\)](#), and [Text Mining](#). The idea is to reduce the representation word-set of a text from the full list of words - i.e. the list that results from the [Bag of Words](#) technique - to a handful of keywords. The advantage is clear. If keywords are chosen carefully, the text representation space is drastically reduced, while the information content is not.

Scientific literature suggests several techniques for automatic keyword extraction, based on statistics, linguistics, and machine learning. The [KNIME Text Processing extension](#), available in [KNIME Analytics Platform](#), implements some of these automatic keyword extraction techniques:

- **Chi-Square keyword extraction** [1], implemented in the [Chi-Square Keyword Extractor](#) node;
- **Keygraph keyword extraction** [2], implemented in the [Keygraph Keyword Extractor](#) node;
- **TF-IDF based keyword extraction** [3], which has to be implemented manually using the two nodes TF and IDF.

All keyword extraction algorithms include the following phases:

- *Candidate generation*. Detection of possible candidate keywords from the text.
- *Property calculation*. Computation of the properties and statistics required for ranking.
- *Ranking*. Computation of a score for each candidate keyword and sorting in descending order of all candidates. The top n candidates are finally selected as the n keywords to represent the text.

The algorithm available in the Chi-Square Keyword Extraction [1] node was used for the “understanding” part of my brain.

For a more detailed review of the algorithms for keyword extraction available in KNIME Analytics Platform, check the blog post [“Keyword Extraction for Understanding”](#).

3. The Ontology

Remember? It all starts with a question, your question. After parsing it and understanding it with the help of the KNIME Text Processing extension, queries are sent to the machine learning model (my brain) for possible answers.

Educational Resources on the KNIME Website

My ultimate goal is to provide you with the one and only web tutorial that perfectly answers your question. Let's start then with the tutorial material that is available on the [KNIME website](#).

First of all, of course, the [e-learning course](#). The e-learning course consists of 7 chapters (as of today, July 11, 2018). There is a good chance the topic you are looking for is here as you'll find information on everything from installing [KNIME Analytics Platform](#) and its extensions to data access, ETL procedures to machine learning algorithms, and data visualization to flow control.

Another very much visited resource is the [KNIME blog](#). The KNIME blog started in 2014 with the mission of producing nuggets of data science, machine learning, and [KNIME tools](#).

Then, there is the [Node Guide](#). Similarly, to the blog, this resource contains even more atomic pieces of information on single nodes and single use cases.

Finally, there are the books from [KNIME Press](#), covering basic and advanced KNIME functionalities, text processing, data blending, ETL, etc.

As this tutorial repository consists of some 400 pages, I am very likely to find what you need, and probably more. The problem is that this "what you need" is not sorted and not easy to find. Entering some meaningful keywords in the site search box, will probably lead you to a forum thread asking a similar question to yours. By the way, users asking questions rarely know the meaningful keywords for a search.

In addition, if you try a conversational style, the site search engine will lose you completely. In contrast, I can understand your more complex text and I should – if trained properly – be able to match the right resources to the keywords of your question.

My brain needs training in order to match question and resources, i.e. ultimately the right resource URLs. So, can this be done by training a machine learning model?

The questions in the [KNIME Community Forum](#) could be used as training examples. Indeed, while it is true that the KNIME Forum should answer support questions, it is also true that a lot of questions start with “I am a beginner with KNIME...” This implies that the KNIME Community Forum is actually used also as an educational tool. A machine learning model could then be trained to associate the right resource URLs to the questions in the KNIME Forum.

All of the content of these educational pages on the KNIME site as well as the questions posted on the KNIME Forum can be downloaded using the open source [Palladian web crawling extension](#) for [KNIME Analytics Platform](#).

Data Science and Machine Learning in a Less than Perfect World

In a perfect world, there is a dataset, a set of classes, and labels for all records in the dataset. Using the provided class system and the corresponding labels, a training set and a test set can be created and a model can be trained on the training set and evaluated on the test set. This procedure can be repeated until performance is satisfactory.

In the real world, there is a dataset, tons of missing values, no labels, and sometimes even no clue about which class system to use. This describes our case!

Data in a perfect world ...

classes

Training set

1	2	1	0	4	2	1	0
2	3	1	5	6	2	0	0
6	2	0	0	2	3	1	1
1	5	6	2	3	3	0	3

Data in a less than perfect world ...

classes

Training set

	2	1	0		2	1	
	2	3	1	5	6		
6	2		0	2	3	1	
5		2	3	0			

Figure 6.

Data in a perfect world on the left and data in a less than perfect world on the right. In a real life project often datasets miss values, labels, and even a class system.

We have the forum questions on the one hand, which we want to use as training set, and the tutorial pages on the other, to be used as classes. Unfortunately, questions in the forum are not always linked to any of the

tutorial pages, i.e. labels in the dataset are not provided. This hints at unsupervised machine learning, with some kind of clustering or distance based assignment.

As a matter of fact, the first attempt to build my brain relied on a [Tanimoto similarity measure](#), matching the keywords extracted from each question with the keywords extracted from the resource pages. Results were sub-optimal. The a priori distribution of resource topics and the low number of extracted keywords constrained the model to opt for resource documents belonging to the most represented tutorial topic: installation. Therefore, questions were often redirected to installation resource pages. Experimenting with other similarity measures did not change the assignment strategy considerably.

So, if the first unsupervised attempt was disappointing, I hear you say, what are the other options? [Active learning](#) could be an alternative option to refine the label system. Active learning starts with a dataset with potentially faulty labels, trains a model, extracts a subset of the original dataset for manual relabeling, retrains the model, and repeats until no significant change in model performance is observed.

The dataset resulting from the label assignment based on the Tanimoto similarity measure could be the starting point of an active learning procedure.

The next step would be to train a supervised model on it, i.e. on a training set with approx. 400 possible classes, i.e. the 400 page URLs of tutorial resources. This is clearly not feasible, especially considering the fact that the number of questions available on the KNIME Forum between 2013 and 2017 barely exceeds 5000. We need to rethink the class system. Let's go back to the beginning.

My ultimate goal is to provide you with the one and only web tutorial page that answers your question. Well, this is hardly possible. Even if you are a beginner to KNIME, you are likely to ask questions that require materials from two, three, or even four different tutorials to get the right answer. It is probably better to provide you with a list of potentially helpful tutorials, rather than just one. Furthermore, it might even be best to identify the areas of expertise touched on by your question and then identify the most relevant tutorial resources for each one of these areas.

This is what my brain should do: identify the areas of expertise and, from these, identify the list of the most relevant articles.

Therefore, the machine learning model should be coupled with a similarity search feature. The machine learning model should be trained to identify the areas of expertise and the similarity search feature should identify the list of the most relevant articles within each area. This opens up a whole new set of problems:

1. Selection and training of a machine learning model
2. Definition of the areas of expertise, i.e. the class system (the ontology), on which to train the model
3. Set up of an active learning framework to iteratively refine the labels in the training set
4. Implementation of a similarity search procedure

Let's focus on the ontology.

The Final Ontology

A class system is also referred to as a class [ontology](#).

An ontology is a set of entities, each with a name, properties, and relations to the others. A class ontology is a set of classes, each with a name, properties and relations. The most famous class ontologies can be found in medicine, such as anatomy ordered classes, or in biology, as the inter related animal classes. In a data science problem, an ontology is a set of classes used to train a machine learning algorithm. To get more information about ontologies and why they are needed, you can check R. Shane's post "[Why Ontologies?](#)"

As a result of machine learning algorithm limitations, a class ontology should contain a limited number of classes, especially with a training set of reduced size. In our case, we are looking for an abstraction of the ~400 tutorial pages into 20 or maybe 30 high level classes.

One of the main principles in data science (as in life, too) is to avoid reinventing the wheel. If something can be reused, then let's reuse it!

For example, the [KNIME e-learning course](#), which has been organized into seven chapters: Installation and Introduction; Data Access; ETL; Data Export or Deployment; Data Visualization; Machine Learning and Predictive Analytics; and Control Structure. These chapters could be used as the first seven classes of the desired ontology, to identify questions about installation, data access, and so on.

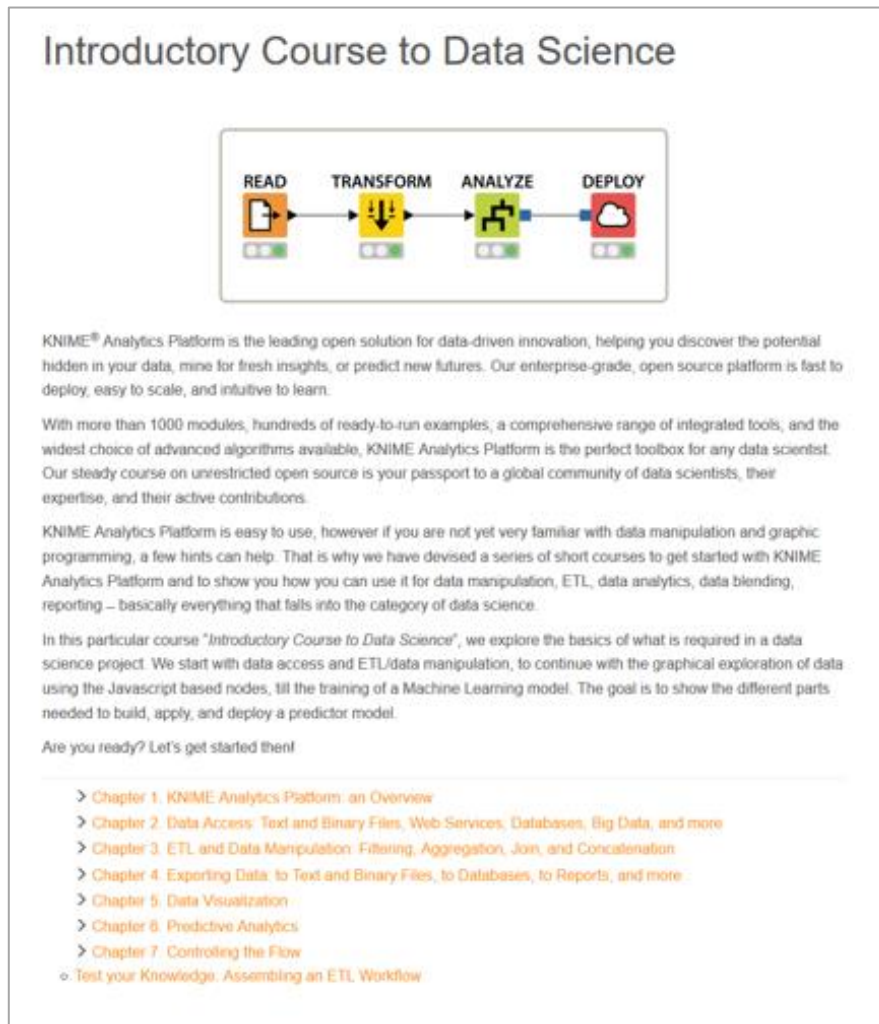


Figure 7.

"Introductory Course to Data Science" on the KNIME site is organized into seven main chapters.

Other popular resources on [the KNIME website](#) include a number of extensions - such as [Text Processing](#), [Image Processing](#), [Big Data](#), and [Reporting](#) – documentation about [KNIME Server](#), and use cases in the form of simple [solutions](#) or more complex [whitepapers](#). Let's add these additional six classes to the ontology, for a total of 13 classes.

However, it soon becomes evident that 13 classes are not yet enough to describe all of the questions stored in the KNIME Forum! Looking closer, we also find questions about node development, integration (mainly with R and Python), life science topics, and software performance. In addition, a few posts are simply announcements of partner extension releases, bug reports, or copyright questions. These additional seven classes, taken from observation, have been added to the initial ontology, to reach a total of 20 classes (Fig. 8). I am sure this final ontology of 20 classes could be further refined and perfected.

Even though defining a class ontology is not a code writing or workflow building activity per se, it is nevertheless of vital importance for the success of the project. Indeed, this was the breakthrough point of this whole project.



Figure 8.

The final class ontology used to train a supervised model to categorize questions from the KNIME Forum.

4. Training the Brain: the Active Learning Cycle

Let's talk now about the making of Emil's brain.

As already described in the previous sections, we used the questions from the [KNIME Forum](#) posted between 2013 and 2017. Questions were imported and stored as unanswered, since only a few answers contained links to educational resources and only some of those links referred to up-to-date educational material.

At the same time, we also adopted a class ontology with 20 classes.

Emil's brain's goal then became two fold:

1. Associate the right class from the ontology to the keywords summarizing each question
2. Within each predicted class, explore educational resources on the KNIME site and extract the top four most relevant ones.

Let's focus on goal # 1: associate the right class from the ontology to each question; that is to the keywords summarizing each question.

The Classification Problem

The difficulty of this problem lies in the dataset being **unlabeled**.

Here, we could have opted for an unsupervised algorithm, with the hope of matching the final clusters with the ontology classes, or we could have enforced a supervised training by using the [active learning](#) technique.

Since [unsupervised learning](#) requires the definition of an appropriate number of clusters and, by definition of unsupervised, does not even try to match the desired classes, we leaned towards the second option: the supervised learning plus active learning approach.

A third option of course could have been to manually label the whole dataset, but that is expensive and time consuming. Given the budget constraints, this was not a viable option for this project.

The Active Learning Cycle

The active learning cycle starts with a labeled training set. It is not important, at first, how correct the labels are, as long as a labeled dataset of whatever quality is available.

The initial Labeled Dataset

We calculated the [N-gram Tversky distance](#) between the keywords extracted from each question and the keywords extracted from each tutorial page on the KNIME site. The class of the closest tutorial page was then assigned to the question. This was our starting point as a labeled dataset. The workflow:

“02_AL_First_Try_Assign_Classes_via_Distance” is available on the EXAMPLES Server under:

`knime://EXAMPLES/50_Applications/33_Emil_the_TeacherBot/`.

On this however mislabeled dataset, we trained the supervised model of choice: a [Random Forest](#) with 100 trees.

Detecting the Frontier Points

After training the model, the points lying at the frontier of the predicted class groups must be identified. The assumption is that the most uncertain predictions refer to the frontier points.

Uncertain predictions are those predictions where the highest probability of the predicted class is not very distant from the second highest probability of the second predicted class. Quantifying: if the difference between the two highest class probabilities produced by the random forest for a given question is lower than 0.2, the predicted class is considered “uncertain”.

Thus, at the end of each training procedure, the 10% most uncertainly classified questions are extracted. These are the frontier questions across predicted classes.

The workflow “03_AL_Training_Subset_Uncertain_Classes” extracts such frontier questions and it is available on the EXAMPLES Server under: 50_Applications / 33_Emil_the_TeacherBot / 02_ActiveLearning.

Relabeling the Frontier Points

These top 10% uncertainly classified questions were manually relabeled by one of our teammates, represented by the thinking statue in Figure 9. For this re-labelling phase, the workflows “04_AL_Re-label_Uncertain_Classes” and “05_AL_Labeling” were used.

These workflows are also available on the KNIME EXAMPLES Server under

knime://EXAMPLES/50_Applications/33_Emil_the_TeacherBot/02_ActiveLearning.

Extending Labels to Neighbor Points.

The manual labels are then extended to the closest neighbor points and from there to the neighbors of the neighbors using a [k Nearest Neighbor \(k-NN\) algorithm](#) with $k=1$. The workflow showing this aspect is “06_AL_Extend_Expert_Classes_with_kNN” and is available under *knime://EXAMPLES/50_Applications/33_Emil_the_TeacherBot/02_ActiveLearning* on the EXAMPLES Server.

Repeat

With this freshly relabeled training set, we retrain our supervised model. Again, the 10% questions with the most uncertain predictions are extracted and manually relabeled by the thinking teammate. And so on.

The active learning cycle is shown in Figure 9.

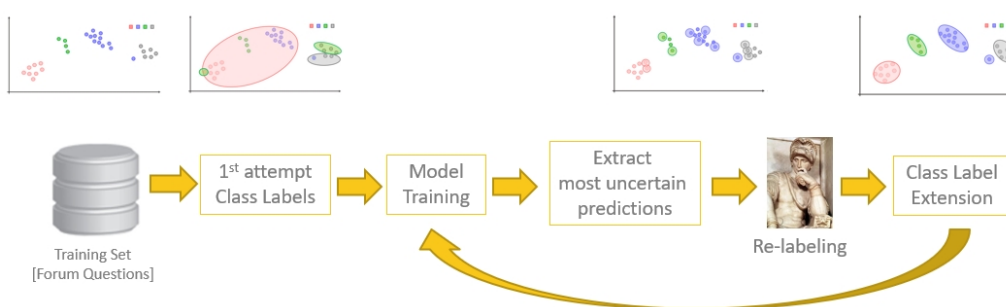


Figure 9.

The Active Learning Cycle used to build a labeled dataset to train the machine learning model for the brain of Emil, the Teacher Bot.

Class Assignment

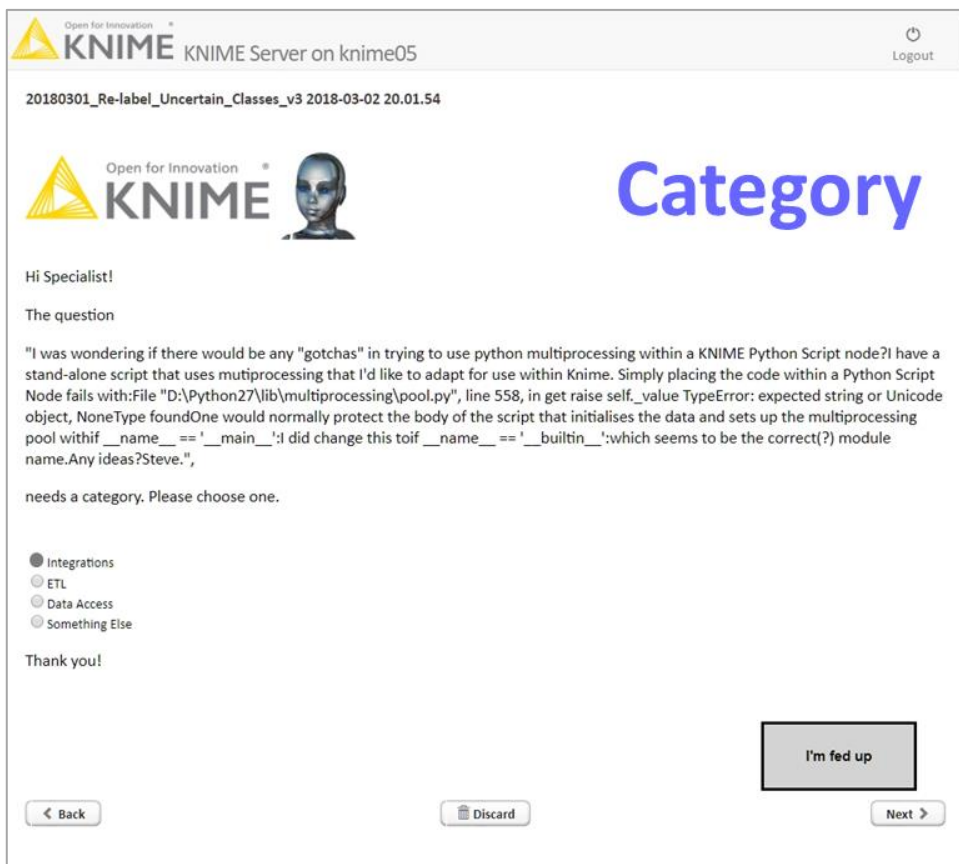
After training the random forest model, a Math Formula node computed the difference between the three highest class probabilities for each question. Finally, the subset of the 10% most uncertainly classified questions - i.e. with the closest distance between the top three classes - was extracted. This subset was then used as input for the manual relabeling.

GUI for Manual Relabeling

Two web GUIs were employed to manually relabel the uncertainly answered questions on KNIME WebPortal:

- one web page to confirm or reject the most likely class as the question label;
- in case of rejection, another web page to manually insert the new required labels.

The first web page, named "Category", asking whether to confirm or reject the top predicted class as the question label, is reported in Figure 10a; the corresponding sub-workflow is shown in Figure 10b.



20180301_Re-label_Uncertain_Classes_v3 2018-03-02 20.01.54

Open for Innovation[®] KNIME

Hi Specialist!

The question

"I was wondering if there would be any "gotchas" in trying to use python multiprocessing within a KNIME Python Script node? I have a stand-alone script that uses multiprocessing that I'd like to adapt for use within Knime. Simply placing the code within a Python Script Node fails with: File "D:\Python27\lib\multiprocessing\pool.py", line 558, in get raise self._value TypeError: expected string or Unicode object, NoneType found One would normally protect the body of the script that initialises the data and sets up the multiprocessing pool with if __name__ == '__main__': I did change this to if __name__ == '__builtin__': which seems to be the correct(?) module name. Any ideas? Steve."

needs a category. Please choose one.

☒ Integrations
☐ ETL
☐ Data Access
☐ Something Else

Thank you!

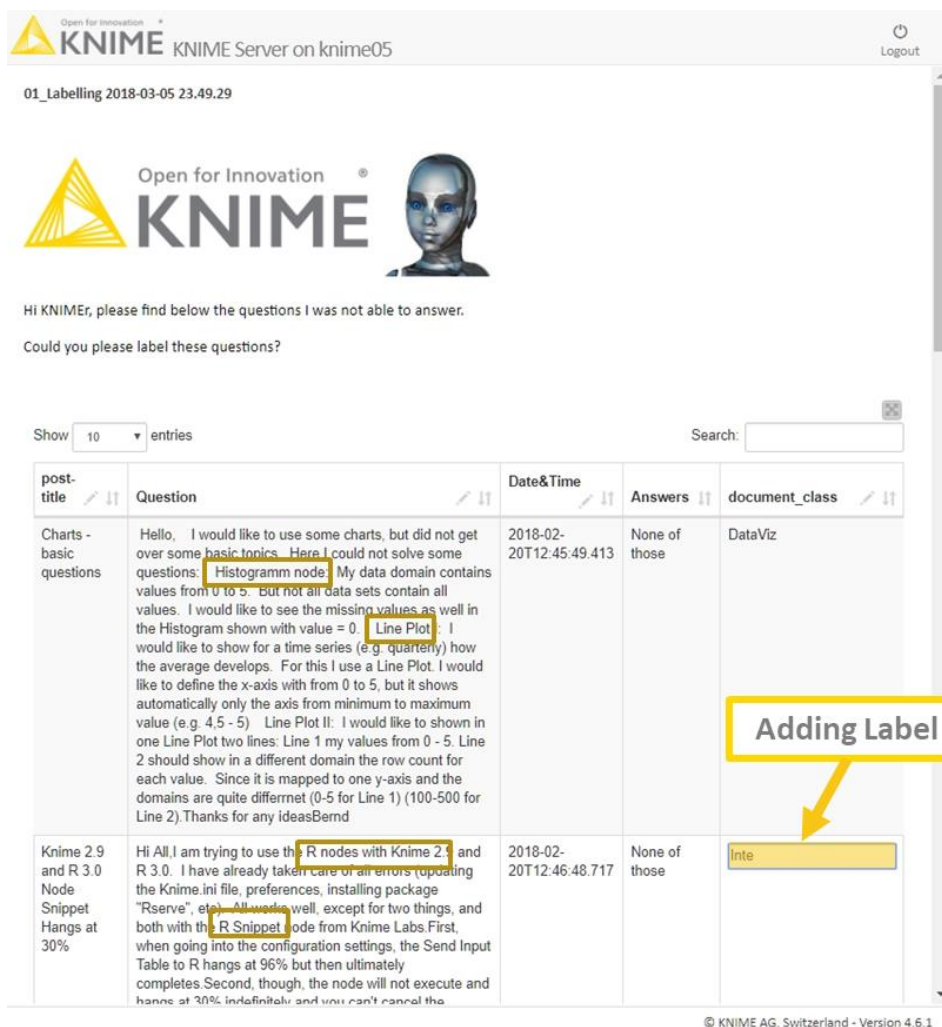
Figure 10a.

This page allows confirming or rejecting the most likely class as the question label. Selecting the "Something Else" option defers the class assignment to the next step. The "I'm fed up" button stops this labeling process.

Figure 10b.

Content of the wrapped metanode producing the web page in Figure 10a. The wrapped metanode allows confirming or rejecting the most likely class as label. The Text Output node, the Value Selection node, and the Generic JavaScript View node serve to generate elements of the web page.

The second web page assigning new labels to unlabeled questions is shown in Figure 11a; the corresponding sub-workflow is shown in Figure 11b.



01_Labelling 2018-03-05 23:49:29

Open for Innovation[®] **KNIME**

Hi KNIMEr, please find below the questions I was not able to answer.

Could you please label these questions?

Show 10 entries Search:

post-title	Question	Date&Time	Answers	document_class
Charts - basic questions	Hello, I would like to use some charts, but did not get over some basic topics. Here I could not solve some questions: Histogramm node: My data domain contains values from 0 to 5. But not all data sets contain all values. I would like to see the missing values as well in the Histogram shown with value = 0. Line Plot: I would like to show for a time series (e.g. quarterly) how the average develops. For this I use a Line Plot. I would like to define the x-axis with from 0 to 5, but it shows automatically only the axis from minimum to maximum value (e.g. 4.5 - 5). Line Plot II: I would like to shown in one Line Plot two lines: Line 1 my values from 0 - 5. Line 2 should show in a different domain the row count for each value. Since it is mapped to one y-axis and the domains are quite different (0-5 for Line 1) (100-500 for Line 2). Thanks for any ideasBernrd	2018-02-20T12:45:49.413	None of those	DataViz
Knime 2.9 and R 3.0 Node Snippet Hangs at 30%	Hi All, I am trying to use the R nodes with Knime 2.9 and R 3.0. I have already taken care of all errors (updating the Knime.ini file, preferences, installing package "Rserve", etc.). All works well, except for two things, and both with the R Snippet node from Knime Labs. First, when going into the configuration settings, the Send Input Table to R hangs at 96% but then ultimately completes. Second, though, the node will not execute and hangs at 30% indefinitely and you can't cancel the	2018-02-20T12:46:48.717	None of those	Inte

© KNIME AG, Switzerland - Version 4.6.1

Figure 11a.

Web page to manually insert new labels.

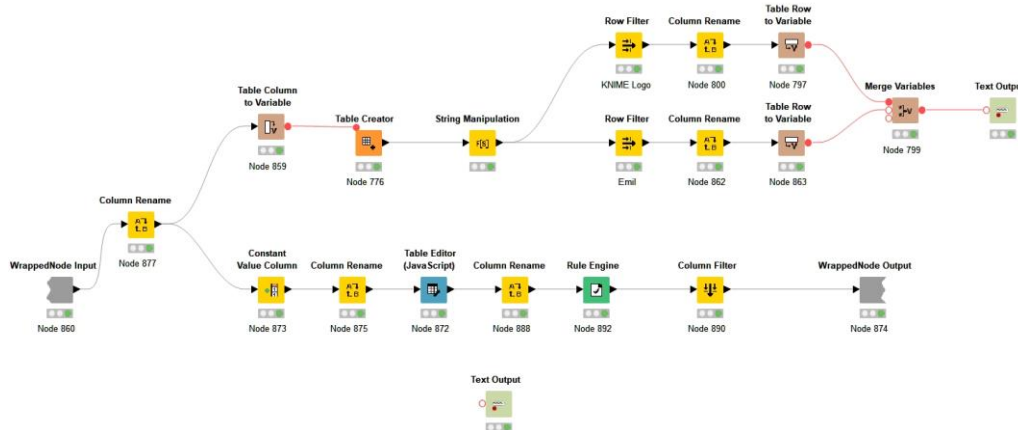


Figure 11b.

The sub-workflow contained in the wrapped metanode generating the web page in Figure 11a. The produced page allows manual insertion of new labels. The Text Output nodes and the Table Editor (Java Script) node serve to generate elements in the web page.

The two workflows implementing this part of the active learning cycle are: “04_AL_Re-label_Uncertain_Classes” and “05_AL_Labelling” and are available on the EXAMPLES Server at the following path knime://EXAMPLES/50_Applications/33_Emil_the_TeacherBot.

Note. Thanks to the now editable fields of the Table Editor (JavaScript) node, changes to all selected labels could be applied within just one web page.

The Dreadful Questions

Now the dreadful questions that come at the end of each project.

1. How many iterations are needed to reach a reasonably labeled training set?
2. How did the class distribution in the dataset evolve across iterations?
3. And, finally, did it work? That is, did we reach a reasonably labeled training set to produce a model with good enough performances?

How many active learning iterations are needed?

This is one of those questions that does not have an exact answer. It depends too heavily on the data and on the classification problem. Usually, active learning iterations are run until no more changes in the class distribution on the training set are observed. For this project, three iterations were sufficient to reach a stable class distribution on the training set. Which leads to our next question.

How did class distribution change across iterations?

For each iteration, we represented the class distribution on the training set with a word cloud. Words are the class names. Word size is the number of points assigned to that class. The word clouds for iteration 0 (starting point), 1, and 2 are reported in Figure 12.

As a starting point we assigned classes to questions on the basis of the N-gram Tversky smallest distance between the closest tutorial page and the question. Since on the KNIME site, pages describing how to install KNIME Analytics Platform abound, the most frequently assigned class to the forum questions was “Installation”, which is also the dominant word in the word cloud at iteration # 0.

After the first iteration (iteration # 1) of manual labeling and k Nearest Neighbor assignment, the most numerous class became “ETL”, which makes more sense if considering that most questions on the KNIME Forum refer to ETL, i.e. data manipulation, operations in data science projects.

The situation did not change much after iteration # 2. Therefore, this is where the active learning cycle stopped.

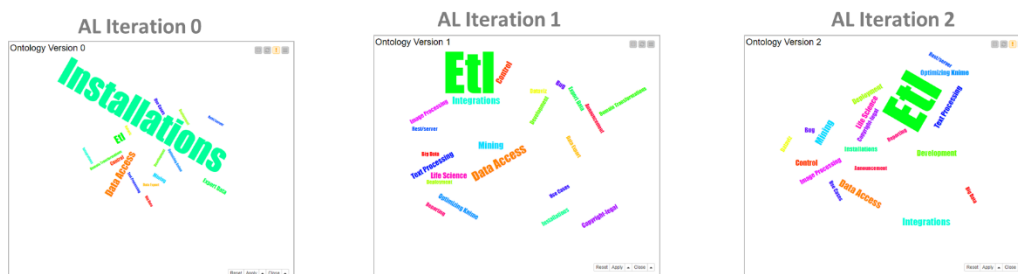


Figure 12.

Word cloud of class frequency in the training set at iteration 0, 1, and 2 of the active learning cycle.

And, finally, did it work?

It is hard to define a metric for success here. The usual sensitivity, specificity, accuracy, Cohen’s Kappa won’t work, since we refer to a potentially wrongly labeled dataset. A 90% accuracy on a faulty test set only tells us that the model has learned to generalize possibly faulty rules on unseen data.

Indeed, in our project, accuracy on the test set decreased with the progressing of the active learning cycle, which could be a very discouraging factor. Let’s walk through the active learning cycle using one single question as an example. The example question is:

Summary of the Question: *Trouble in connecting to SQL Server*

Detailed Question: *Hi! I am using database node reader to connect to SQL Server database. I have loaded the JDBC Driver and configured the node but I got the following error: the TDS protocol stream is not valid. What should I do to solve this error? Any help would be appreciated.*

The right class would be “Data Access” and the answer should lead to database tutorials.

The right class would be “Data Access” and the answer should lead to database tutorials.

After iteration 0, the top two proposed classes are “Installation” and “ETL”. In phase two, the tutorial pages on the KNIME site with the closest distance to the original question end up being “The Workflow Coach” and “The EXAMPLES Server”. These pages are not necessarily wrong answers, since the Workflow Coach and the EXAMPLES server cover a bit of all topics; but they are not very specific either.

After iteration 1, the proposed predicted class, i.e. the output class with highest probability, is “Data Access”, which then leads to tutorial pages [“Twitter meets PostgreSQL”](#) and [“Blending Databases. A Database Jam Session”](#). Both answers would definitely be more pertinent to the original question.

After iteration 2, proposed output class and final proposed tutorial pages have not changed.

Note. Accuracy is not the only measure to a successful project!

While it is hard to quantify success in this case, it is easy to see that, at least for this single question, the final answer becomes more correct despite a decreasing accuracy. The same happened for a number of other questions we inspected.

Note. A common starting point for an active learning procedure consists of manually labeling a few random records from the most populated regions. This improves the quality of the dataset already from iteration 0.

Active Learning Summary

We have shown here how to train a supervised model with an iteratively adjusted supervision, via an active learning cycle. The idea behind active learning is that, even if you do not have a reliably labeled dataset, you can still start from whatever dataset you have and iteratively and partially relabel the data, hoping that the extension process to the neighboring points will improve the quality of the dataset.

The usage of web pages on KNIME WebPortal for confirming / rejecting labels and editing new ones has made the human intervention within the cycle easier and more efficient.

This active learning procedure has allowed us to train Emil's brain with supervision, even if a set of labeled questions was not available.

Hi, I am Emil and I can find out which tutorial pages are most suitable for answering your question.

5. Conclusions

In the project described in this workflow, we have assembled Emil, the teacher bot, by puzzling together some of the functionalities available in [KNIME Analytics Platform](#). That is, a web based user interface to ask the question; a set of text processing NLP functions to parse it; and a machine learning model to associate the right group of tutorials to the asked question.

The bot task was to provide helpful links to technical tutorials to help people who are new to KNIME progress in their learning curve.

Puzzling together the pieces was actually the easy part of the project. The main difficulty consisted in the adoption of an unlabeled dataset to train a supervised machine learning model. We solved this issue by adopting an active learning procedure to progressively label and relabel a subset of significant points in the data set.

Workflows are available on the KNIME EXAMPLES server under *50_Applications/33_Emil_the_TeacherBot*

References

- [1] Matsuo, Y., Ishizuka, M. (2004). [Keyword extraction from a single document using word co-occurrence statistical information](#). *International Journal on Artificial Intelligence Tools* 13, (157–169).
- [2] Oshawa, Benson, and Yachida (1998). [KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor](#)
- [3] Ramos, J. (2013). [Using TF-IDF to Determine Word Relevance in Document Queries](#).
- [4] Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application. *Journal of Documentation*, Vol.28, pp. 11-21.