Deliverable 3.1

Categorization Tool

Grant Agreement number: **250467**

Project acronym: **ATLAS**

Project type: **Pilot B**

Deliverable D 4.1 Categorization Tool 19.05.2012

Project coordinator name, title and organisation:
Anelia Belogay, CEO; Diman Karagiozov, CTO
Tetracom Interactive Solutions
Tel: +35924950444
Fax: +35924950443
E-mail:anelia@tetracom.com, diman@tetracom.com
Project website address: www.atlasproject.eu

Authors:
Anelia Belogay, Diman Karagiozov, Radostin Surilov (Tetracom)
Project co-funded by the European Commission within the ICT Policy Support Programme

Dissemination level: **PUBLIC**

Document history

Revision	Date	Author	Description
0.1	5 May 2012	Diman Karagiozov	initial version, table of contents
0.2	19 May 2012	Radostin Surilov	algorithms, references
0.3	03 June 2012	Diman Karagiozov	i-Publisher integration and categorization UI
0.7	28 June 2012	Diman Karagiozov	Tests
1.0	30 June 2012	Anelia Belogay	Final version

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of contents

Overview

The document classification is a task which assigns a document to one or more categories or classes in a taxonomy. As the automation of this process is of great importance for modern applications, a variety of methods have been developed during the last several years.

The methods for automatic classification can be split into two groups – statistical algorithms and structural algorithms. Examples for statistical algorithms are Regression and Naïve Bayes. The structural algorithms can be further divided into Rule Based (Decision Trees, Production rules), Distance Based (kNN, Centroid) and Neural Networks (Marmanis, Babenko, 2009).

Single-label classification is concerned with learning from a set of documents, which are associated with a single label (class) - *l* from a set of labels - *L*. In multi-label classification each document can be associated with more than one label from *L*. If *L* contains exactly two labels, the learning problem is called binary classification, and if *L* contains more than two labels, the problem is called multi-class classification (Tsoumakas, Katakis, 2007).

The automatic categorization tool (ACT), developed in the scope of the ATLAS project, focuses on the multi-label multi-class automatic categorization task. Furthermore, a convenient graphical user interface for building and evaluating categorization models has been provided as part of the integration between the ACT and the content management system component in ATLAS, namely i-Publisher.

This document is organized as follows:

- the “Automatic categorization tool” chapter focuses on the requirement, software architecture and implemented features of the ACT;
- the “Algorithms” chapter enlists the categorization algorithms which have been integrated in the ACT. The API for integrating new algorithms is well documented;
- the “Testing” chapter describes the testing and evaluation infrastructure, regression tests strategy and the ACT performance on the Reuters-21578[1] corpus and the EUDocLib[2] set of documents;
- the final chapter provides ideas for extending the ACT with additional categorization algorithms and better exploitation of the ACT results in the WP5 (“Text summarization”) and WP6 (“Machine translation”) related tasks.
- “Appendix A” describes the integration of the ACT in the i-Publisher ATLAS component.

Automatic Categorization Tool

This chapter describes the requirements for the ACT and the architecture of the categorization tool and depicts the integration patterns in i-Publisher. Finally, we present a short walkthrough through the i-Publisher user interface with respect to the categorization

[1] http://about.reuters.com/researchandstandards/corpus/

[2] EUDocLib (http://eudoclib.atlasproject.eu/) is a proof-of-concept web site developed entirely with i-Publisher. The website contains more than 140’000 EURLex (http://eur-lex.europa.eu/) documents, categorized within 3 subsets of EuroVOC (http://eurovoc.europa.eu/) thesaurus.

functionalities in ATLAS.

Requirements

Environment and user requirements

The ATLAS automatic categorization tool is used in a multilingual and multi-domain environment. Furthermore, the volume of the categories varies - from less than 100 in i-Librarian[3] and Reuters-21578 to more than 5'000 in EUDocLib. The volume of the training and test data also varies - from less than 10 documents per category in i-Librarian and an average of 100 training instances in Reuters-21578, to more than 20'000 training documents per category in EUDocLib.

Software requirements

The following software requirements have been derived from the above-mentioned user requirements and environmental settings. The ACT tools should:

- be language independent, in order to address the multilingual aspects of the data;
- be domain independent, so that they can be used for building categorization models for various domains with variable categorization label sets;
- provide means of choosing optimal categorization model settings, in order to address the variable sizes of training corpora and categorization labels sets;

Furthermore, the ACT should be integrated within i-Publisher in order to enable the users to manage the categorization label sets, training data and models, as well as to suggest the most appropriate labels for one or more documents to the user.

Finally, the ACT should support the i-Publisher users to organize their content even if there is minimal or no training data.

ACT features

The ACT features message-based network-distributed communication; various feature types for the categorization feature space; various feature space reduction techniques; an extensible classifier (algorithm) infrastructure; three available classifiers and one clustering algorithm; model builder; model applier and results assembler. All these features and components are described in the following subchapters. Additionally, ACT is integrated in ATLAS in the backend and GUI layers.

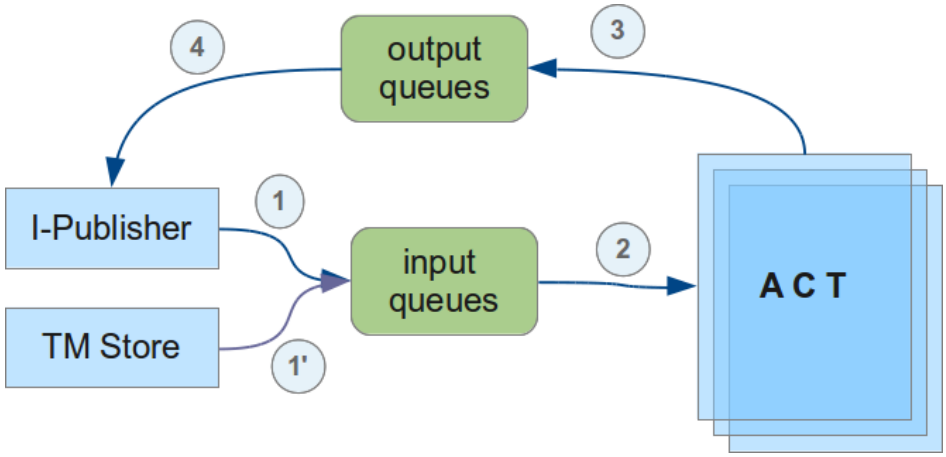
Asynchronous communication

The automatic categorization tasks (training and predicting) usually require substantial hardware resources, therefore the classical request-response pattern cannot be applied. We adopted a network-distributed architecture, based on an asynchronous message processing framework (ActiveMQ or RabbitaMQ[4]). All requests for model building, label predictions and document clustering are firstly sent to **input queues**. Software components, designed

[3] i-Librarian (www.i-librarian.eu) is a website implemented completely with i-Publisher. Being a digital library, a categorization tool is of a great benefit for organizing the content. The categorization functionalities in i-Librarian feature the supervised and unsupervised machine learning algorithms.

[4] ApacheMQ - http://activemq.apache.org ; RabbitMQ - http://www.rabbitmq.com/

to handle **one specific type of messages** check out a message from an input queue and process the message. The result is sent to an **output queue** once the processing of the task is over. Then i-Publisher collects the result from the output queue and stores it in a database. The diagram below depicts this workflow.



This top-level architecture has an important advantage - horizontal scalability can be easily achieved by adding another ACT engine which connects to the same input and output queues.

Features types

We use the output of the language processing chains, developed in WP4[5] (Language Processing Chains), in order to address the multilingual aspect of the processed data. Each entity (sentence, token, noun phrase and named entity) is represented by its unique identifier (ID). In this way, each document is presented as a sequence of natural numbers and in this way the input to the ACT becomes language independent. Another benefit of presenting the documents as a sequence of numerical features is that we can use high performance and low-memory footprint JAVA collections.

The categorization feature space can be based on the following types of features:

- token - each token is identified by its lemma, PoS and word sense;
- lemma - the same as token but the PoS and word sense information is not used;
- noun phrase - the document is presented as a sequence of its noun phrases only;
- head nouns - the document is represented as a sequence of the head nouns of the noun phrases in this document;

Currently, a mixed type feature space is **not** supported by the ATLAS categorization tool.

Feature space reduction

As the dimension of the feature space can be huge, we provide several feature space reduction techniques (FSRT). It is possible to chain several FSRT, e.g. firstly the feature space is reduced by the first FSRT in the chain, then by the second, and so on. The

[5] WP4 (Language Processing Chains) has ended in month 21 (December 2011). The results from this work package are harmonized chains of heterogeneous NLP tools which provide tokenization, sentence splitting, POS tagging, lemmatization, noun phrase chunking and named entities recognition in Bulgarian, English, German, Greek, Polish and Romanian. Detailed description of the NLP tools, the integrated chains, quality and performance of the LPCs can be found in deliverable D4.1.

currently supported FSRT are:

- top_N - this technique assigns weight to each feature in the space and retains only the best N features. N could be a number, e.g. 5000, or a percentage, e.g. 5%, of the feature space;
- prune - this technique selects only features that are common for more than "**prune_below**" and less than "**prune_above**" documents;
- chi2 - this technique implements the **χ2** distribution[6] with one degree of freedom to test how important one feature is for a given category/label. For example, **chi2_100** selects the 100 best (according to the **χ2** score) for each category in the model.

Classifiers

Once the feature space is defined and optionally reduced, each classifier (classification algorithm) builds a normalized weighted structure for each category. This structure is either represented as a vector in the feature space (for centroid classifier) or as a probability distribution (for naive bayesian and entropy classifiers). The categorization model is actually a serialized version of the set of all weighted structures.

Building a model

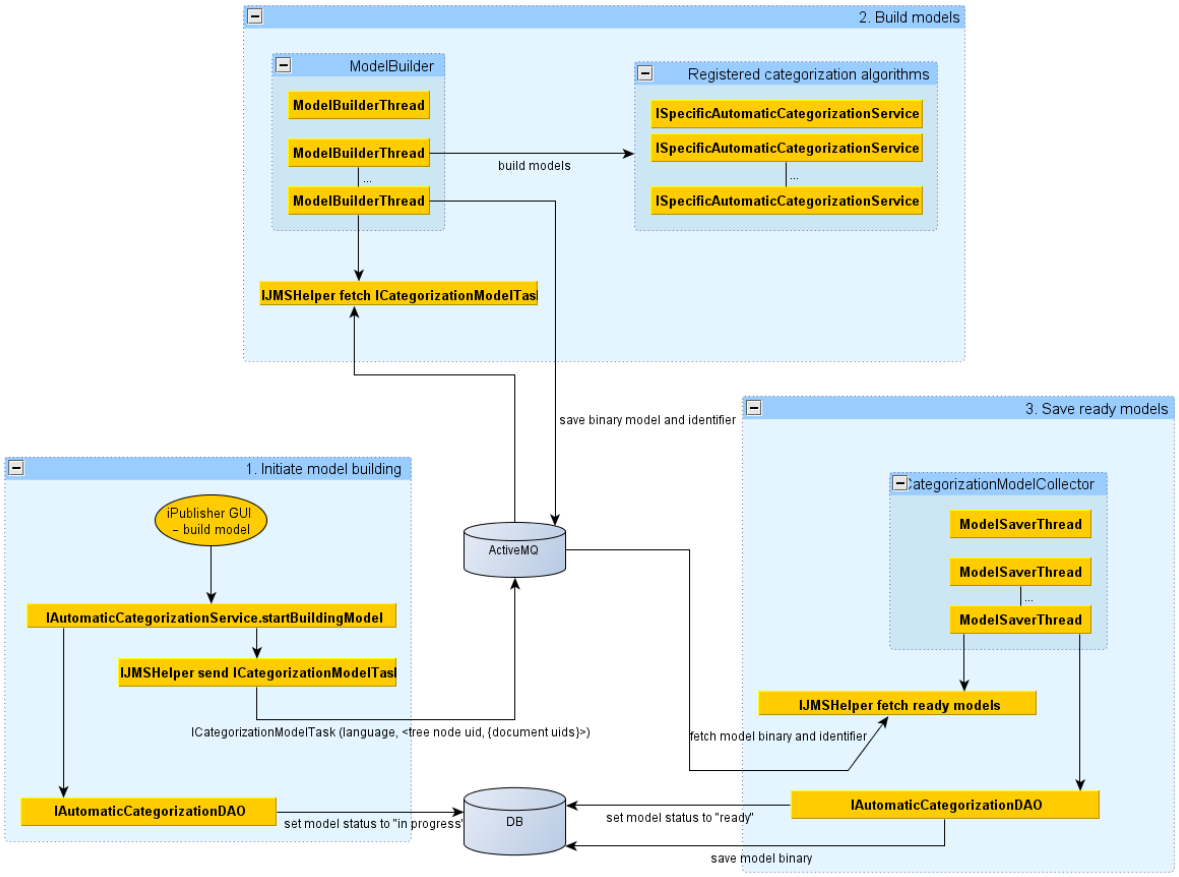
Firstly, the i-Publisher user decides on the model feature type and dimension reduction techniques. Secondly, the i-Publisher backend sends the request for building a specific model to an input queue.

The ACT then checks out the message and builds a model for each of the available/ configured classifiers (algorithms). This process fetches the training data, builds the feature space, reduces it, forms the vectors for each category and normalizes them. The set of vectors is then serialized and sent to an output queue.

Finally, i-Publisher fetches the built model from the output queue, saves it in the database and marks the model as "ready-to-be-used".

The schema below outlines the main steps in the building of a categorization model. It is important to note that each classifier uses its own model.

[6] http://en.wikipedia.org/wiki/Chi-squared_distribution



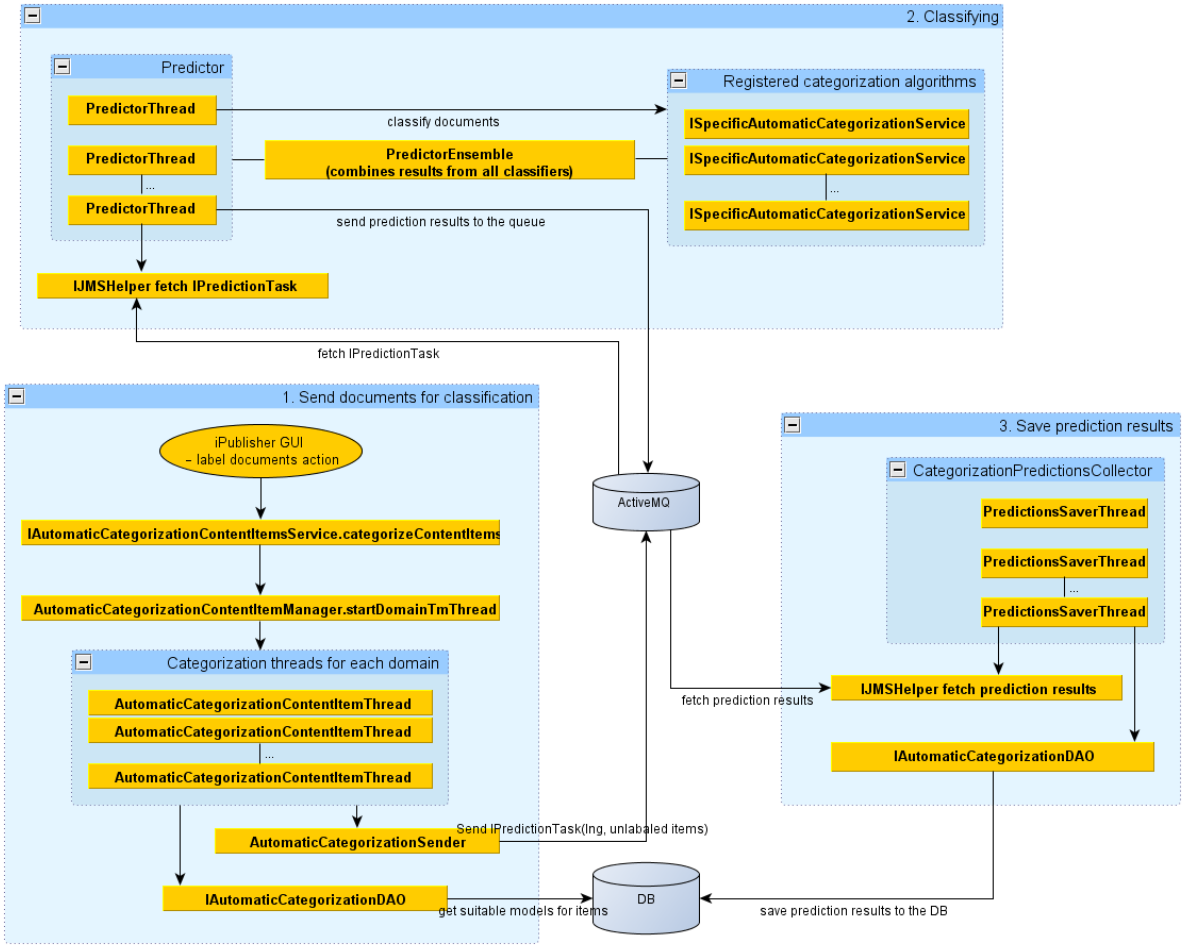
Using a model

The process of using a model consists of the following steps:

1. the user (i-Publisher, i-Librarian, or anonymous from another ATLAS component) triggers and action for predicting the categories of one or more content items;
2. according to the configuration of the initiating component, the categorization action is either immediately sent for execution or queued. The second option is useful when a batch of content items are being categorized. The optimal batch size was experimentally set to 128 content items;
3. for each classifier
    a. each document is presented with its features (matching the feature type of the model);
    b. a normalized weighted structure (vector or probability distribution) is created for each document;
    c. each structure is “matched” against each category in the model and a category score is calculated. The score is either a distance or a probability;
    d. the result contains the score for each category for each document in the batch.
4. the results from all classifiers are assembled and the top-N categories for each document in the batch are sent to an output queue;
5. i-Publisher then connects the messages from that output queue and stores the results in the database.

The diagram below depicts the process of predicting categories for one document or a batch

of documents.



Algorithms

The categorization algorithms described here are available in the ACT. Individual evaluation for each of the algorithms has been performed on the Reuters-21578 corpus. The results of the evaluations can be found in the “Performance results” subchapter.

CFC and CFC-modif

The Class-Feature-Centroid (CFC) and its modified version (CFC-modif) classifiers are based on the work by Hu Guan, Jingyu Zhou and Minyi Guo from the Shanghai Jiao Tong University.

The Centroid-based approaches feature short training time and testing time due to their computational efficiency. However, the accuracy of centroid-based classifiers is inferior to SVM, mainly because the centroids found during construction are far from perfect locations.

The Class-Feature-Centroid (CFC) classifier for multi-class, single-label text categorization is built from two important class distributions: inter-class term index and inner-class term index. CFC defines a novel combination of these indices and employs a denormalized cosine measure to calculate the similarity score between a text vector and a centroid. Experiments

on the Reuters-21578 corpus show that CFC consistently outperforms the state-of-the-art SVM classifiers on both micro-F1 and macro-F1 scores. CFC is more effective and robust than SVM particularly when data is sparse.

Relative entropy

This classifier, based on the Kullback–Leibler divergence[7] (also information divergence, information gain, relative entropy, or KLIC), is a non-symmetric measure of the difference between two probability distributions - P and Q. KL measures the expected number of extra bits required to code samples from P when using a code based on Q, rather than using a code based on P. Typically, P represents the "true" distribution of data, observations, or a precisely calculated theoretical distribution. The measure Q typically represents a theory, model, description or an approximation of P.

The text categorization is performed using the Kullback-Leibler distance between the probability distribution of the document and the probability distribution of each category. Using the same representation of categories, experiments show a significant improvement when the above mentioned method is used. The KLD method achieves substantial improvements over the *tfidf*[8] performing method.

Naive Bayesian

In simple terms, a naive Bayes classifier[9] assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability of this fruit being an apple.

Despite their naive design and apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of naive Bayes classifiers. An advantage of the naive Bayes classifier is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix. Our experiments showed that using Naive Bayesian classifier in a combination with Laplace smoothing[10] significantly improves the quality of the classifier.

LDA (experimental)

[7] http://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence
[8] http://en.wikipedia.org/wiki/Tf*idf
[9] http://en.wikipedia.org/wiki/Naive_Bayes_classifier
[10] http://en.wikipedia.org/wiki/Laplace_smoothing

LDA is an experimental algorithm, which uses generative model[11] for dimension reduction and applies machine learning techniques to obtain the appropriate categories. The experimental algorithm used for space reduction is based on the Latent Dirichlet Allocation (LDA) model developed by David Blei. In the this approach we treat LDA as an algorithm which reduces the number of features representing a document down to the K 'most meaningful'. LDA represents each document as the vector of those features (all features values sum to one, most significant one has the highest value). Suppose we have assigned those LDA features to the text. Such an assignment is a vector in R^K, where K is the number of features. In this case we face a typical problem of mapping K LDA features to N categories. You can consider this task the problem of the machine learning and use such methods as support
vector machine, k nearest neighbors, etc.

Bisecting K-Means clustering

K-means (MacQueen, 1967) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a smart way because a different location causes a different result. According to the literature, bisecting k-means clustering has better performance than the standard k-means algorithm and comparable quality to the hierarchical clustering.

Currently, the ACT implements the bisecting k-means for clustering and in-house topic modelling technique. LDA topic modelling will be also available in the ACT by the end of the project.

Adding a new classifier

The ACT implementation is OSGi[12]-based and adding a new classifier actually means that a new service, implementing the ISpecificAutomaticCategorizationService (com.tetracom.atlas.textmining.categorization.api.service.ISpecificAutomaticCategorizationService) interface should be made available in the ACT OSGi container.

TheISpecificAutomaticCategorizationService interface has two self-explanatory methods **buildModel** and **useModel**:

[11] Generative model for a document is based on the assumption, that documents can be created by sampling from hidden random variables, which describe theirs structure. Generative model is a concept drawn from statistical models and so topics models are examples implementing broader class of formalisms used to describe the different phenomena (e.g. modeling of the DNA, proteins, evolution of the population).

[12] http://www.osgi.org/Main/HomePage

src ▸ com.tetracom.atlas.textmining.categorization.api.service ▸ ISpecificAutomaticCategorizationService ▸

```
package com.tetracom.atlas.textmining.categorization.api.service;

import java.util.List;

public interface ISpecificAutomaticCategorizationService {

    public IBinaryModel createModel(
            UUID modelUid,
            String lngCode,
            Map< UUID, Set< UUID >> trainingItems,
            String algorithmIdentifier
    ) throws Exception;

    public Map< UUID, Map< UUID, Double >> useModel(
            IBinaryModel model,
            UUID modelUid,
            String lngCode,
            List< UUID > unlabeledItems,
            int threadSequence
    ) throws Exception;

    public String getAlgorithmName();
}
```

Sample implementation of the classifier service can be found in the following plugins:

- com.tetracom.atlas.textmining.categorization.algorithms;
- com.tetracom.atlas.textmining.categorization.mallet.lda;
- com.tetracom.atlas.textmining.categorization.mulan;
- com.tetracom.atlas.textmining.categorization.unizd.

Testing

Series of evaluation and regression tests have been performed in order to ensure the stable consistent high-quality work of the automatic categorization tool in ATLAS. This chapter describes the testing environment and corpuses, provides an overview of the performance and quality results and finally elaborates on regularly executed regression test scenarios.

Training/Testing corpora

The ATLAS ACT is theoretically language independent and it should not matter what training/test corpus will be used for the evaluation of the tool. Furthermore, there are no freely-available corpora, annotated for the text categorization task. Therefotre we focus the initial evaluation of the ACT on the Reuters-21578 corpus.

Reuters-21578 collection Apte' includes 12,902 documents for 90 classes, with a fixed splitting between test and training data (3,299 vs. 9,603). The categories are presented as different directories. The set of files (one for each document,) associated with the target category, are stored in each directory. The non-labeled documents from the Reuters corpus are stored in the directory "*unknown*". The document file names are increasing non-repeating numbers for fast document indexing. Two different main directories (test and training) store the training/testing data.

The corpus was firstly imported in ATLAS in the form of content items from two content types - ReutersTrainingItem (7768) and ReutersTestItem (3019 items). The items in the *"unknown"*

category were excluded from the experiments. Secondly, the imported content items were processed by the English LPC and various categorization models were built.

The evaluation procedure compares the manually categorized test items to the automatically suggested categories (folders). The precision, recall and f1-measure are provided for each category in the model. The evaluation is completed with the micro and macro f1-measures for the whole model.

Performance results

The conducted experiments aim:

- to ensure that the implementation and integration of different classifiers is properly done and the quality of each individual classifier is inline with the state-of-the-art results;
- to completely exploit the semantic annotations made available from WP4 (Language Processing Chain) and to study the influence of the feature type on the classifier quality.

A table with detailed information on each of the experiments can be found at https://docs.google.com/spreadsheet/ccc?key=0AuWVg_Q7c1mudExmNVJBMHgyTXdaeWhiTGhlcEZlX1E.

No	Feature type	Feature reduction	Features count	Lables	CFC-modif		Relative Entropy		Naive Baesyan		Enseemled	
					micro f1	macro f1	micro f1	macro f1	micro f1	macro f1	micro f1	macro f1
1	token	top_100	39017	1	0.8256	**0.7936**	**0.8763**	0.7154	0.8360	0.6384	0.8524	0.7248
2				2	0.8924	**0.8654**	**0.9313**	0.8088	0.8970	0.7126	0.9078	0.8208
3				3	0.9232	**0.8915**	**0.9477**	0.8453	0.9222	0.7528	0.9276	0.8638
4		top_20	7800	1	0.8068	**0.7170**	**0.8675**	0.6882	0.8343	0.6354	0.8478	0.7183
5				2	0.8803	**0.8528**	**0.9245**	0.7734	0.8880	0.6738	0.8994	0.7870
6				3	0.9101	**0.8674**	**0.9440**	0.8164	0.9152	0.7320	0.9232	0.8474
7		top_10	3900	1	0.7790	0.6817	**0.8632**	0.6536	0.8434	**0.6828**	0.8434	0.6828
8				2	0.8632	**0.7687**	**0.9168**	0.7253	0.8890	0.6890	0.8967	0.7677
9				3	0.8981	**0.8257**	**0.9380**	0.7806	0.9135	0.7182	0.9182	0.8216
10		top_5	1950	1	0.7438	0.5688	**0.8528**	0.6281	0.8374	**0.6345**	0.8374	0.6841
11				2	0.8310	0.6811	**0.9064**	**0.7253**	0.8944	0.6811	0.8860	0.7670
12				3	0.8719	0.7409	**0.9333**	**0.7627**	0.9155	0.7175	0.9118	0.7907
13		top_1	391	1	0.5650	0.3946	0.7795	0.4936	**0.8050**	**0.5312**	0.7365	0.5221
14				2	0.6575	0.5052	0.8578	0.5982	**0.8797**	**0.65**[illegible]	0.8145	0.6256
15				3	0.7123	0.5514	0.8958	0.6583	**0.9052**	**0.6745**	0.8501	0.6541
16	head	top_100	4694	1	0.5516	0.3627	0.6408	0.4432	**0.7305**	**0.4985**	0.7166	0.4839
17				2	0.6794	0.4871	0.7947	0.5577	**0.8343**	**0.6099**	0.7989	0.6022
18				3	0.7343	0.5640	0.8423	0.6311	**0.8729**	**0.6635**	0.8374	0.6545
19		top_50	2347	1	0.5461	0.3654	0.6383	0.4371	**0.7329**	**0.4966**	0.7127	0.5028
20				2	0.6685	0.4808	0.7934	0.5537	**0.8323**	**0.6228**	0.7962	0.6002
21				3	0.7266	0.5487	0.8393	0.6215	**0.8723**	**0.6682**	0.8376	0.6695
22		top_20	938	1	0.5115	0.3451	0.6213	0.4044	**0.7305**	**0.4862**	0.7033	0.4822
23				2	0.6311	0.4266	0.7155	0.5198	**0.8278**	**0.5566**	0.7866	0.5738
24				3	0.6970	0.4985	0.8257	0.5951	**0.8682**	**0.6354**	0.8243	0.6288
25		top_5	234	1	0.4003	0.2656	0.5351	0.3175	**0.7087**	**0.4400**	0.6274	0.3472
26				2	0.5273	0.3432	0.6331	0.4311	**0.8020**	**0.5315**	0.7264	0.4796
27				3	0.5919	0.3989	0.7601	0.4935	**0.8460**	**0.5634**	0.7647	0.5559
28		**chi2_100**	6271	1	0.8128	**0.7547**	**0.8712**	0.7244	0.8615	0.7048	**0.8551**	**0.7432**
29				2	0.8886	**0.8460**	**0.9316**	0.7933	0.9215	0.7938	**0.9168**	**0.8311**
30				3	0.9252	**0.8862**	**0.9530**	0.8271	0.9420	0.8331	**0.9383**	**0.8779**
31				1	0.8070	**0.7456**	**0.8601**	0.7075	0.8547	0.6917	0.8581	0.7709

The following conclusions can be made based on the presented results:

- all implemented classifiers perform properly and their quality is comparable with the state-of-the-art achievements on the Reuters-21578 corpus;
- naive-bayesian classifier is more suitable when the number of features is small (less than 1000); the relative entropy and cfc-modif provide better results with the feature space is bigger (more than 4000 features);
- as expected, reducing the feature space decreases the overall quality of the model but the deviation is acceptable, especially when the feature space is reduced 5 or 10 times;

- all models based on head nouns perform worse than token-based models. The quality of the head nouns models is not significantly worse than the quality of the token-based models; thus head nouns models could be an option for ATLAS installations on modest hardware;
- the quality of models using the chi-2 feature reduction technique is comparable (or slightly better) than the *tf.idf* top_N feature reduction. However, the complexity of the chi-2 test is $O(n^2)$. One should use the chi-2 reduction having in mind that the models are (re)built rarely;

Regression tests

The quality of the results of the categorization engine strongly depends on the quality of the training and test data. In order to assess the quality of the categorization algorithms, the tests are performed on the well-known and scientifically recognized Reuters-21578 corpus. We record a precision, recall and f1-measure for all 90 categories, as well as the micro- and macro- f1-measure for the whole data set and use them as a template to be compared with the results provided by the newly built model. A test fails if the difference between the recorded and the newly calculated f1-measure is greater than a predefined constant (threshold).

We test the multilingual aspects of the automatic categorization for each of the 80 top-level categories used in i-Librarian. The automatic test builds the categorization models for each language and checks their validity by categorizing the **training** documents. This test fails if the f1-measure is not in a predefined interval for each category. The upper threshold for this predefined interval is set to 0.93 in order to avoid model overfitting.

Conclusion

Automated text categorization is an important technique for many web applications, such as document indexing, document filtering and cataloging web resources. Many different approaches have been proposed for the automated text categorization problem. Despite the great demand for such functionalities integrated in the modern CMS, there are no robust production- and integration- ready solutions.

The ATLAS automatic text categorization features message-based network distributed communication; various feature types for the categorization feature space; various feature space reduction techniques; extensible classifier (algorithm) infrastructure; three available classifiers and one clustering algorithm; model builder; model applier and results assembler. Finally, ACT is seamlessly integrated in ATLAS in the backend and GUI layers.

References

Ferrucci, A. Lally, 2004, UIMA: an architectural approach to unstructured information processing in the corporate research environment. Natural Language Engineering 10, No. 3-4, 327-348

G. Tsoumakas, I. Katakis, 2007, Multi-Label Classification: An Overview, International Journal of Data Warehousing and Mining, 3(3):1-13.

Marmanis, Babenko, 2009: Algorithms of the Intelligent Web. Manning, 2009.

Hu Guan , Jingyu Zhou , Minyi Guo, 2009, A class-feature-centroid classifier for text categorisation, Proceedings of the 18th international conference on World wide web, Madrid, Spain, April 20-24, 2009 [doi: 10.1145/1526709.1526737]

Qian, H. "Relative Entropy: Free Energy Associated with Equilibrium Fluctuations and Nonequilibrium Deviations." 8 Jul 2000. http://arxiv.org/abs/math-ph/0007010.

Bigi, Brigitte, 2003, Using Kullback-Leibler Distance for Text Categorization, Advances in Information Retrieval, Lecture Notes in Computer Science vol. 2633. Springer Berlin / Heidelberg.

Peter Harrington (2012). Machine Learning in Action. Manning Publications Co.

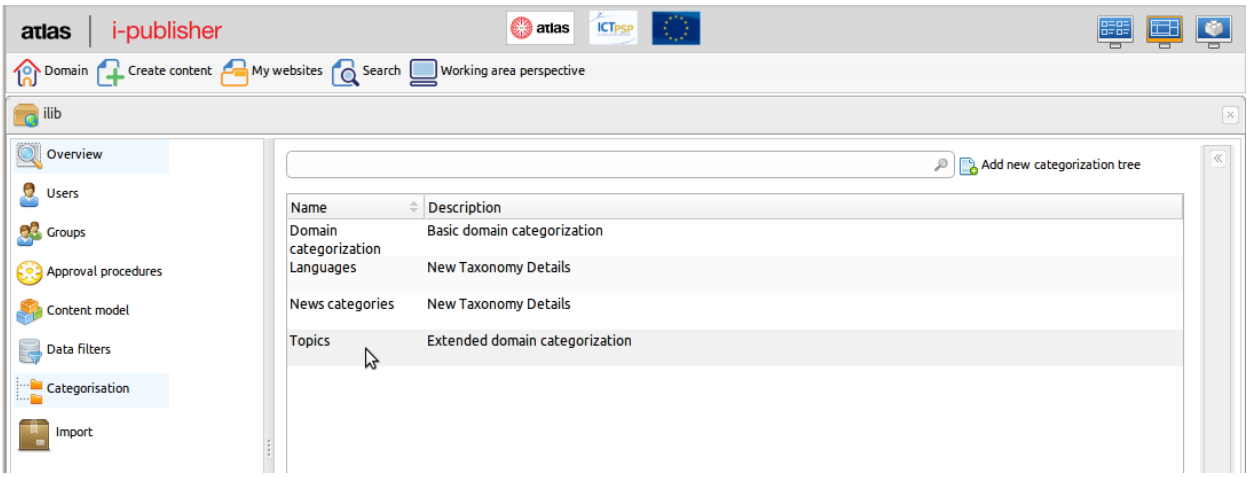
Appendix A: i-Publisher user interface

i-Publisher is the front-most ATLAS component providing the GUI (graphical user interface) between the ATLAS users and the lower functional layers of the system. This chapter describes how the ACT is integrated in i-Publisher and which are the key functionalities, related to the test categorization in ATLAS content management system.

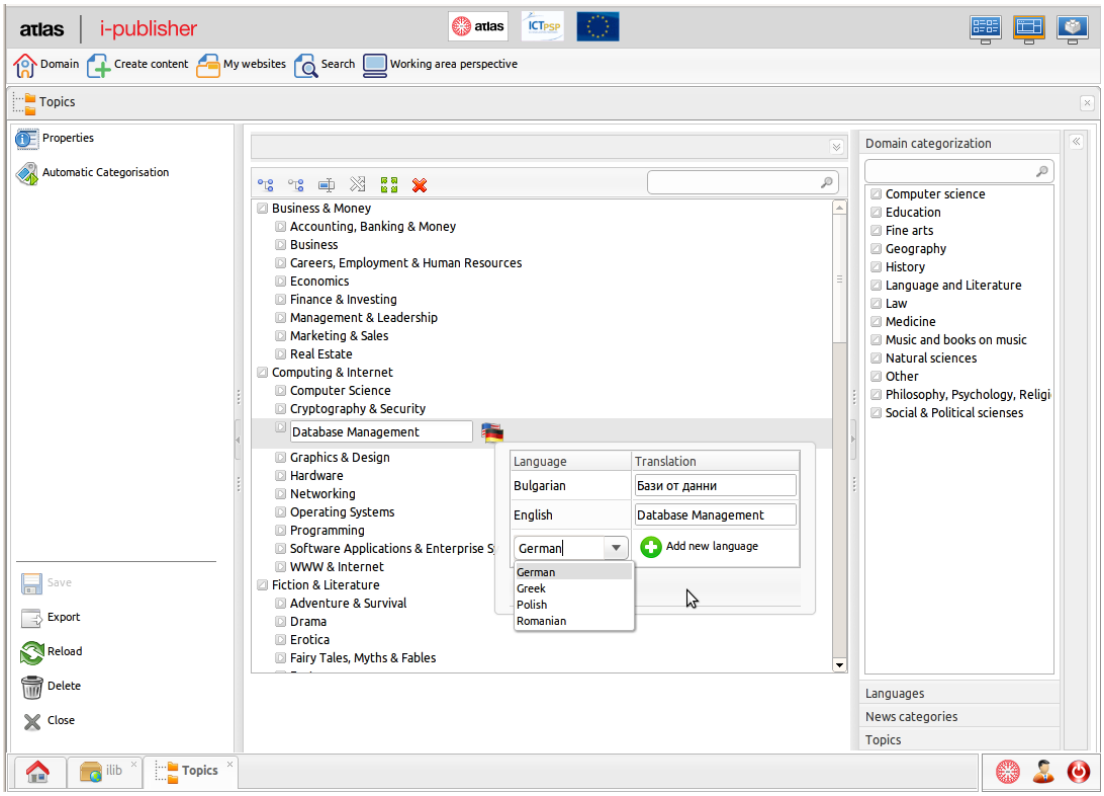
Taxonomy editor

As a modern CMS, ATLAS provides means for creating and further managing one or more taxonomies or flat list of keywords: the categorization labels are hierarchically organized; flat keywords lists are treated in the same way as the tree-like keywords; there is no limitations for the number of taxonomies, depth of their hierarchy or the number of keywords in each structure.

The categorization structures are common for all web sites in a given domain and are managed from the "Domain" editor. The user is provided with drag-and-drop UI for managing the hierarchy of keywords and with a simple translation dialog to translate the keywords in the multilingual domains. The screenshots below show the taxonomy editor.

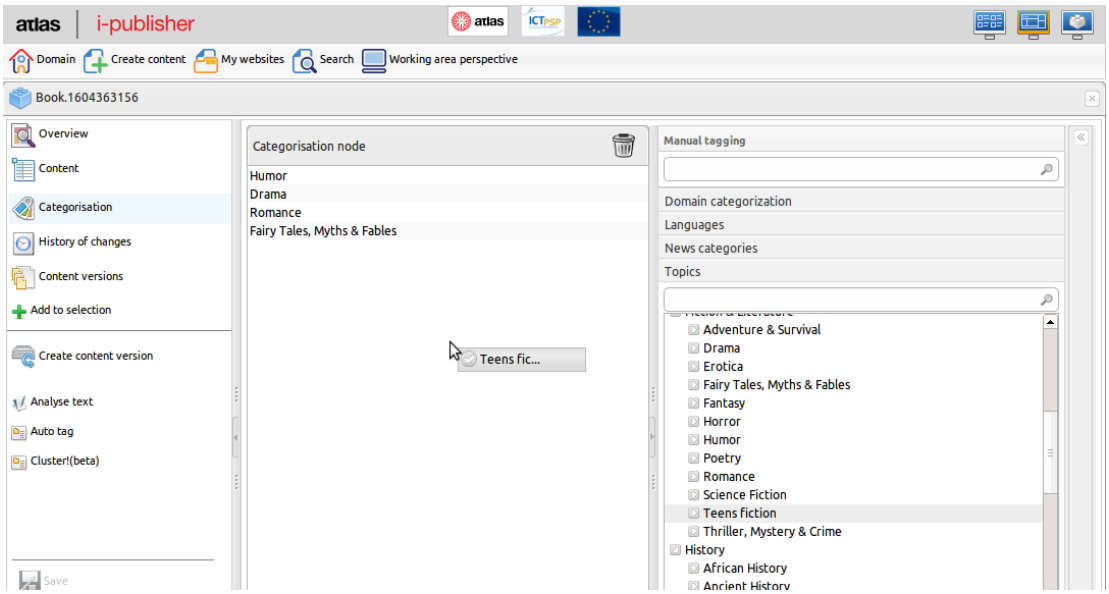


"Domain" editor » "Categorization" shortcut » **List of available taxonomies**

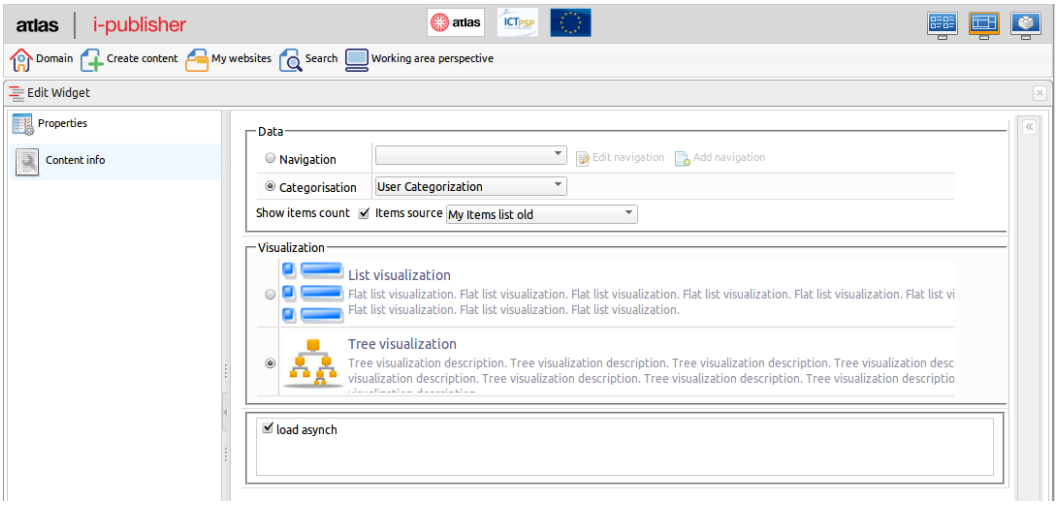


"Taxonomy" editor - toolbar buttons provide means to create new and reorder existing keywords; F2 renames/translates the selected keyword; use drag-and-drop to manage the hierarchy between the keywords.

At this stage, the taxonomies can be used for manually categorized content items to form keywords-based navigation widgets.



Manual categorization of a content item



Definition of **navigation widget** based on categorization taxonomy

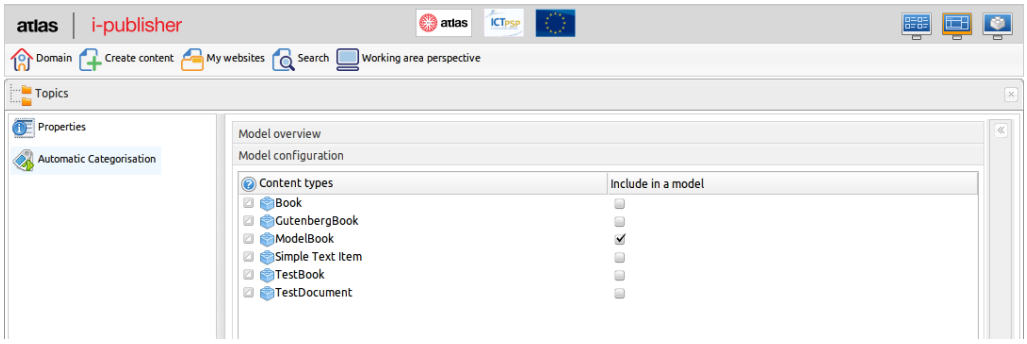
Training a model

Very few content management systems provide a simple user interface for building custom models for automatic text categorization. Most of the automatic categorization functionalities, integrated in the other CMS, use external services, such as AlchemyAPI[13] or OpenCalais[14]. ATLAS project does not replicate the already available services but provides the ATLAS ACT that can be easily tuned via simple GUI, to each individual client or entity. Using the ATLAS ACT tool, a user can easily build models for automatic text categorization, for example, for the domain of biomechanics, black holes or cookery.

The process of building such a tool includes:

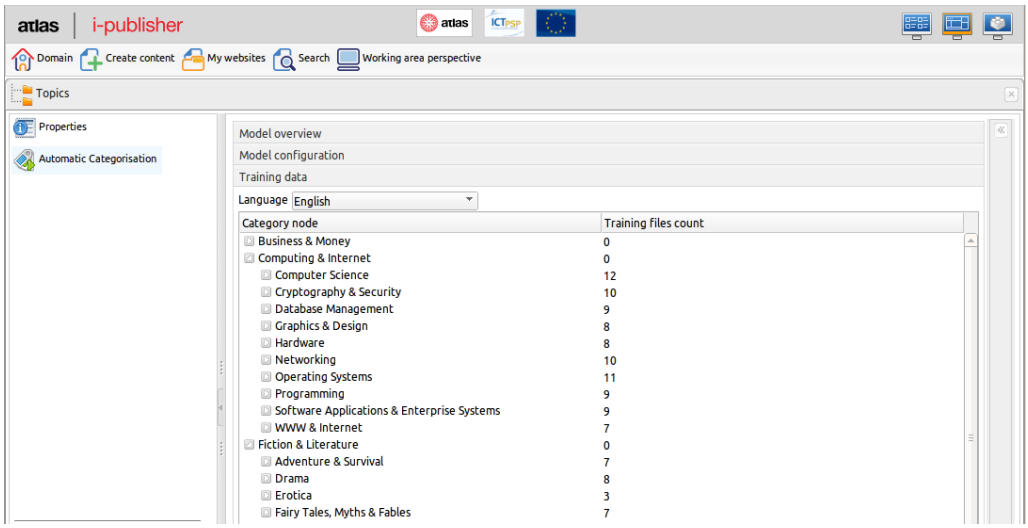
1. an existing taxonomy, built with the "Taxonomy editor";
2. a training/ testing content items. The training content items are first processed by the appropriate LPC and then the results are used to train a model;
3. customization/fine-tuning of the model. Here the user can decide on model-specific parameters, such as feature types, feature space reduction techniques, filtering on the initial set of categories (e.g. omit categories with less than 5 training content items).

The following screenshot shows how a model is firstly configured and then built.

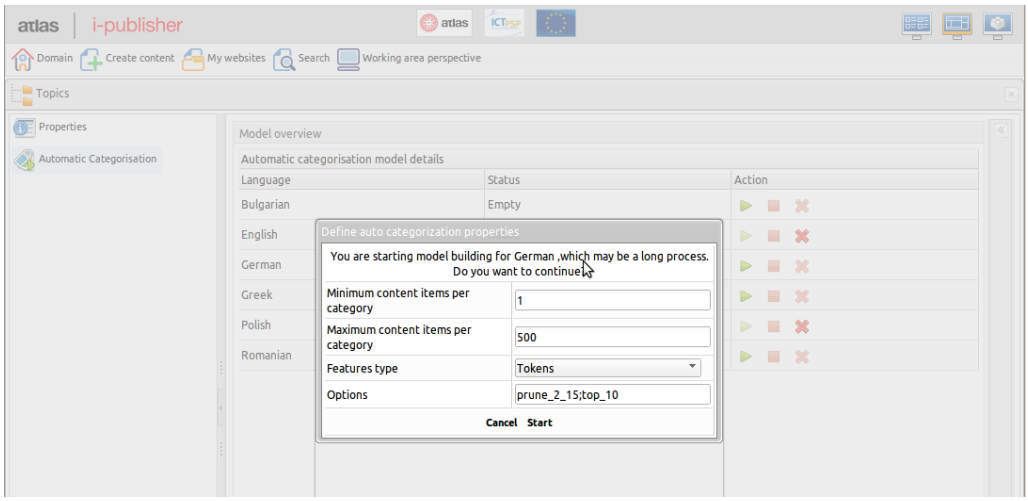


[13] http://www.alchemyapi.com/
[14] http://www.opencalais.com/

“Taxonomy” editor » “Automatic categorization” shortcut » Selecting the set of training items



“Taxonomy” editor » “Automatic categorization” shortcut » Overview of the training content

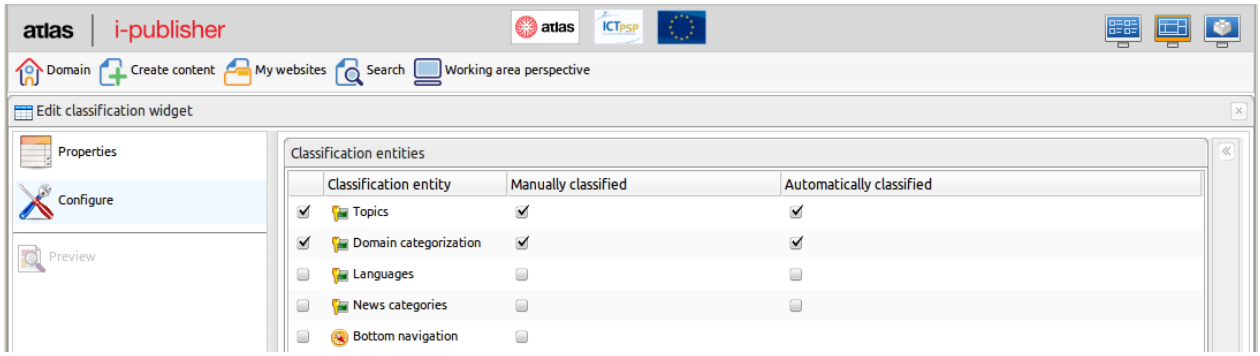


“Taxonomy” editor » “Automatic categorization” shortcut » Setting the model parameters

A separate model is built for every available classifier (algorithm) for each of the available languages. The automatic categorization functionality becomes available when the first model for a language is built. The categories production process is triggered by saving a content item or manually, through the i-Publisher GUI. The latest option is available for a single content item or a list of content items.

Categorization widget

The results of the automatic categorization are visible in i-Publisher, in the “Categorization” tab of the “Content item” editor. Furthermore, these result can be visualized in a website using a “Categorization” widget. This widget is created for each content item.



Clustering widget

The “Clustering” widget shows the results of the document clustering features of the ACT.

Backend evaluation tool

If the user/organization can provide a set of testing item,s the ACT can conduct a formal evaluation of the built models. The backend evaluation tool downloads the test content, processes it through the appropriate LPCs and runs the automatic categorization using the model and the processed data. Then the manual categorization is compared with the automatically suggested. The precision, recall and f1-measure are provided for each category in the model. The evaluation finishes with the micro and macro f1-measures for the whole model.