TopBraid Tagger - User Guide

Page Contents

Introduction

This document describes *TopBraid Tagger*, a web-based tool for linking controlled vocabulary terms to content. Content resources can be tagged, or annotated, through a visual user interface that displays the context for both the content and the vocabulary. Tagger's AutoClassifier capability automatically assigns relevant tags to content. The result is a set of metadata properties that establish a named relationship between the content and vocabulary, and vice-versa. For example, a resource representing a news story can be linked to vocabulary topics of Election and Weather through a property named "has subject," stating that a given news story has topics of Election and Weather.

These relationships—tags—can be used to enrich search, browsing, and other applications by managing metadata on concept-to-vocabulary relationships. The role of Tagger is to make it easy to manage and create these relationships.

Tagger can also be used to create mappings between two vocabularies. In order to do this, simply select one vocabulary as the Content Graph and then another vocabulary as the Concept Vocabulary. (If you do need to build a mapping between two vocabularies, also see if the EDG Crosswalks feature is appropriate.)

Tagger lets you assign terms from a controlled vocabulary (typically, a taxonomy or a thesaurus; but it could also be an ontology) to content resources with a specific relationship. This is referred to as tagging, or annotation, and a collection of such assignments is a tag set. For example, you might need to tag a news story about a sports team being sold so it can be found by a search engine or appear in a list. The news story is represented by an RDF resource that references the story. By creating a relationship named mainSubject between the story and the Business concept and a relationship secondarySubject with the Sports concept, the story is tagged by these relationships and can be used to find other data. For example, a query for all items tagged with a secondarySubject of Sports will find the story.

In Tagger terms, Business and Sports are the controlled vocabulary terms used to tag the news story, and mainSubject and secondarySubject are the relationships, or tag properties, of the news story. These relationships are saved in the Tag Graph, which consists of metadata about the graphs used as content and vocabulary and triples representing the tags. For example, the triples in the tag graph from the example above would have the general form:

```
{  <story> <mainSubject> <Business> .
   <story> <secondarySubject> <Sports> .
}
```

Tag triples are saved in a separate graph for flexibility. Tags can be maintained with EDG working copy change management. Tags can be used in a variety of contexts, including federated SPARQL queries, using owl:imports to attach tags to the content or vocabulary graph, or other contexts.

The choice of terms and properties to use when tagging is up to the administrator of the Tagger installation. The section Creating and Managing Tag Sets below describes how to identify the content to tag, the tag properties, and the controlled vocabulary to use for a given tag set, and the section Tagging documents shows how to assign tags to content with your choice of properties. The section Using AutoClassifier describes how to auto-generate tag assignments.

Creating and Managing Tag Sets

Content Tag Sets provide similar creation and utility functions as asset collection types. The Manage view also has two special features for working with tag sets:

- Add Tag Property Graph lets you add additional sets of properties to use when tagging. You will select from the choices made available by your system administrator as described at Configuring content and property graphs, as well as any viewable ontologies.
- Select Tag Properties lets you modify, by checking or unchecking checkboxes, the list of which properties from the property graphs should be available on the Tagger drop-down property list.

Tagging Documents

See Content Tag Set View or Edit.

Using AutoClassifier

Tagger's AutoClassifier feature can tag a set of documents to a specified taxonomy after you train it with an appropriate set of tagged sample documents. It stores the automatically added tags in a working copy of a tag set where you can review the tags before committing them to production.

This section assumes that AutoClassifier's indexing server is installed. If a *Content Tag Set's* **> Manage > Configure AutoClassifier** is missing, then contact your EDG administrator regarding both (1) the licensing and installation of the Maui Indexer server and (2) the EDG administrative configuration of the AutoClassifier parameters for Maui Server (and Tagger Content Graphs and Property Graphs).

Setting up a tag set

The first few steps of setting up an AutoClassifier session are the same as the setup of a manual tagging session: create a content tag set, specify the content graph, tag property graph, and concept vocabulary, and then select a default tag property and a root content type as described in Creating and Managing Tag Sets.The default tag property is the one that AutoClassifier will use when tagging content items with vocabulary terms.

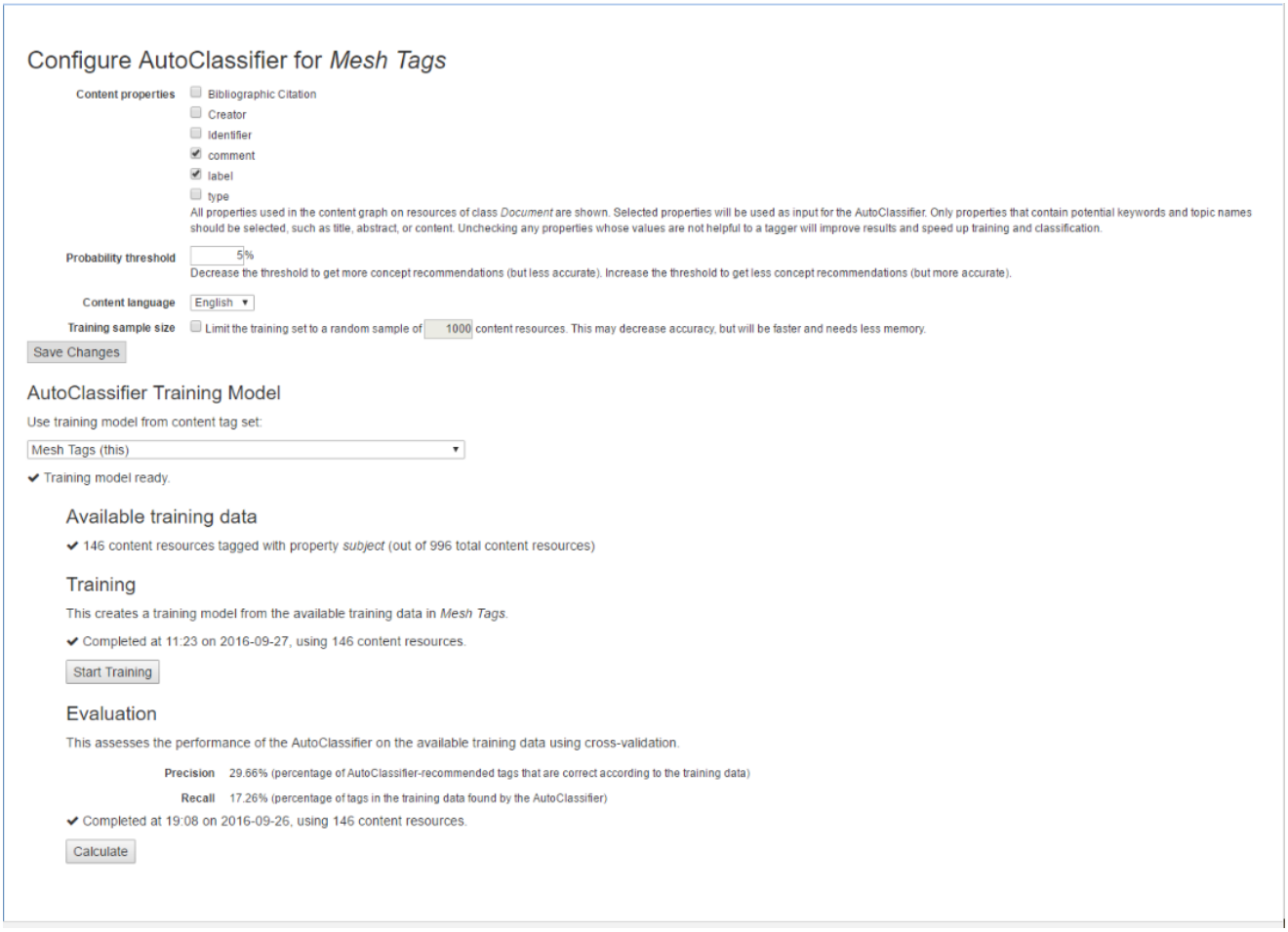
If instances in the content graph include `dc:source` values with URLs pointing to actual documents, Tagger will use these URLs to turn content titles in reports on AutoClassifier activity into hypertext links.

There are a few things to keep in mind about the vocabulary that you use for automated tagging:

- Large taxonomies that cover the domain of interest in detail are good, as they give AutoClassifier more terms to choose from.
- Smaller, shallow taxonomies with general concepts work less well, as the concepts are less likely to directly appear as keywords in the text.
- Taxonomies that have rich skos:altLabel and skos:hiddenLabel values will work best.
- Taxonomies can be improved to work better with AutoClassifier by adding more specific sub-concepts and by adding alternate labels to existing concepts. The goal is to have keywords or phrases that occur in the document corpus appear as labels in the taxonomy.
- Some special considerations apply when using EDG-managed *ontologies* (as opposed to EDG-managed *taxonomies*) as concept vocabularies.

Training the AutoClassifier

After either manually tagging some of the content with the configured default tag property and concept vocabulary, or after importing a set of tags, the next step is to have the AutoClassifier analyze this set of tags to identify patterns that it can use when tagging additional content. To configure this, first ensure (per above) that the AutoClassifier server (Maui) is both installed and configured, then select **Configure AutoClassifier** from the content tag set's **Manage** tab.



As shown above, the configuration screen includes descriptions of the properties that you can set.

The **Content language** offers a choice of English, French, German, or Spanish parsing of the content. This setting also affects the handling of *language-tagged literals* in the content graph and concept vocabulary graph. If a language is chosen here, non-matching language-tagged literals will be ignored during auto-classification. Leaving the setting on *(default)* will use the system language (as configured in Maui Server) and is not recommended for multi-lingual content graphs.

Amount of training data: As a guideline, we recommend having at least 100 tagged content resources as training data. AutoClassifier will work with lower numbers, but the quality of recommended tags will be lower. Adding more training data is good, but there are diminishing returns. If 1000 tagged content resources are available as training data, adding more may make little difference to the quality of results, but will still increase the amount of computing resources required for training. If you wish to limit the training set to a random sample, you may specify a **Training sample size** and enter the desired number of content resources.

Only taggings that use the default tag property will be used as training data.

After choosing appropriate options, click **Save Changes** and then the **Start Training** button. Training may take a while. After training is complete, the AutoClassifier is ready to recommend tags, as described in the next chapter.

Re-training: The training process can be repeated at any time. Re-training is recommended in the following situations:

1. After the taxonomy has changed.
2. When the nature of the content resources changes significantly, e.g., many new content resources that differ in length or in used properties are being added since the last training.
3. If the amount of available training data has increased significantly since the last training, but is still below the numbers where we see diminishing returns.

Note that re-training is not necessary when only a few content resources were added, or when incremental changes were made to content resources.

Evaluation: AutoClassifier works by using the training data to train a machine learning model. AutoClassifier analyses each document in the training data and its associated training tags. If the learning process works well, AutoClassifier is then able to predict similar tags for any other input document. AutoClassifier has an evaluation function that can be used to quantify how well this training process works, by computing precision and recall scores. These scores can be helpful to understand the effect of activities such as adding more training data, adding more skos:altLabels, or adding more concepts to the taxonomy.

Click **Calculate** to compute precision and recall. This calculation is done using 10-fold cross-validation, in other words, the collection of training documents is split into 10 parts, and then 9 parts are used for training, and AutoClassifier is run with the resulting training model on the documents in the 10th part (the test set). This is repeated ten times, with a different part as test set each time. The results are averaged. This process means that the training process never sees the documents that are used for calculating the scores.

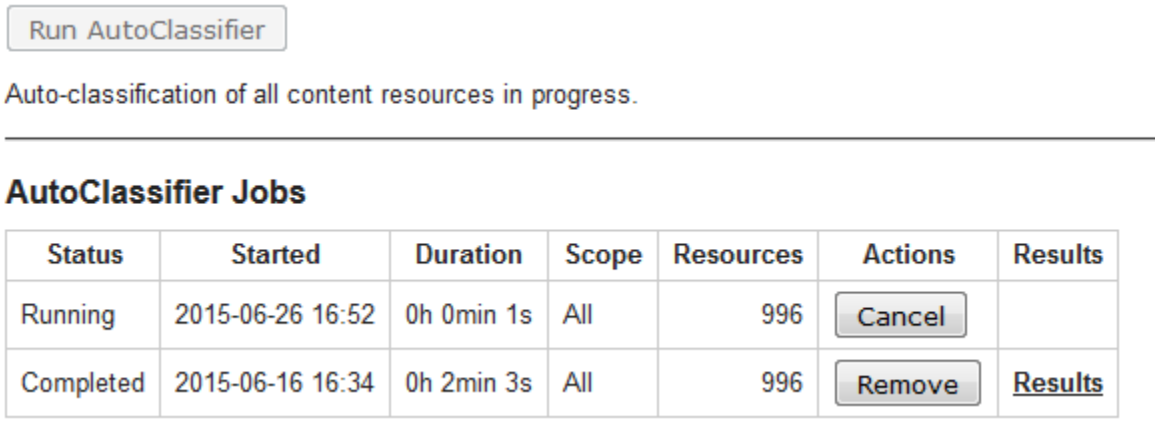
Keeping training data separate from AutoClassifier results: The recommended workflow for AutoClassifier is to keep training data in one content tag set (the *training tag set*), and AutoClassifier results in a different content tag set (the *target tag set*). This ensures that manually created training tags are not mixed with lower-quality auto-generated tags. This setup also makes it possible to do training on one set of documents, and auto-classification on a different (larger) set of documents.

In this scenario, the AutoClassifier configuration and training described above would only be done within the training tag set. After this has been done, navigate to the AutoClassifier configuration screen of the target tag set, and select the training tag set from the *AutoClassifier Training Model* dropdown. This will enable AutoClassifier on the target tag set, using the training result from the training tag set.

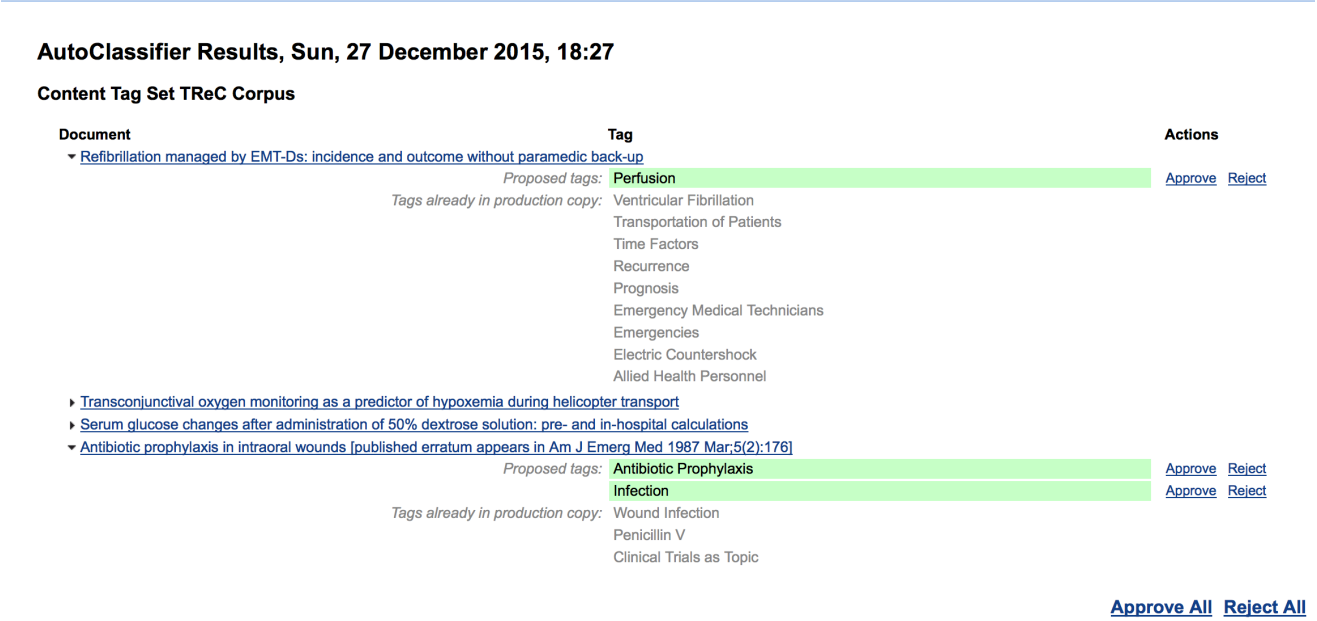
It is recommended that training tag set and target tag set should share the same concept vocabulary. If they use different concept vocabularies, then only terms that are shared between the two vocabularies will be found by the AutoClassifier. If they don't share any terms, AutoClassifier will be unable to generate any recommendations.

Running the AutoClassifier on the entire content set

Once you've done your training, you can return to the **AutoClassifier** tab and click the **Run AutoClassifier** button. You will see a new row added to the table on that page, showing that your job is running and giving you the opportunity to cancel the job if you wish, as shown in the "Running" row of the table below:



If you refresh your browser before the job is completed, you will see the **Duration** value for the running job updated in the table. The next time that you refresh your browser after the job is finished, you will see that the status has changed from "Running" to "Completed," and a "Results" link will appear in the final column. This link leads to a report similar to the following:



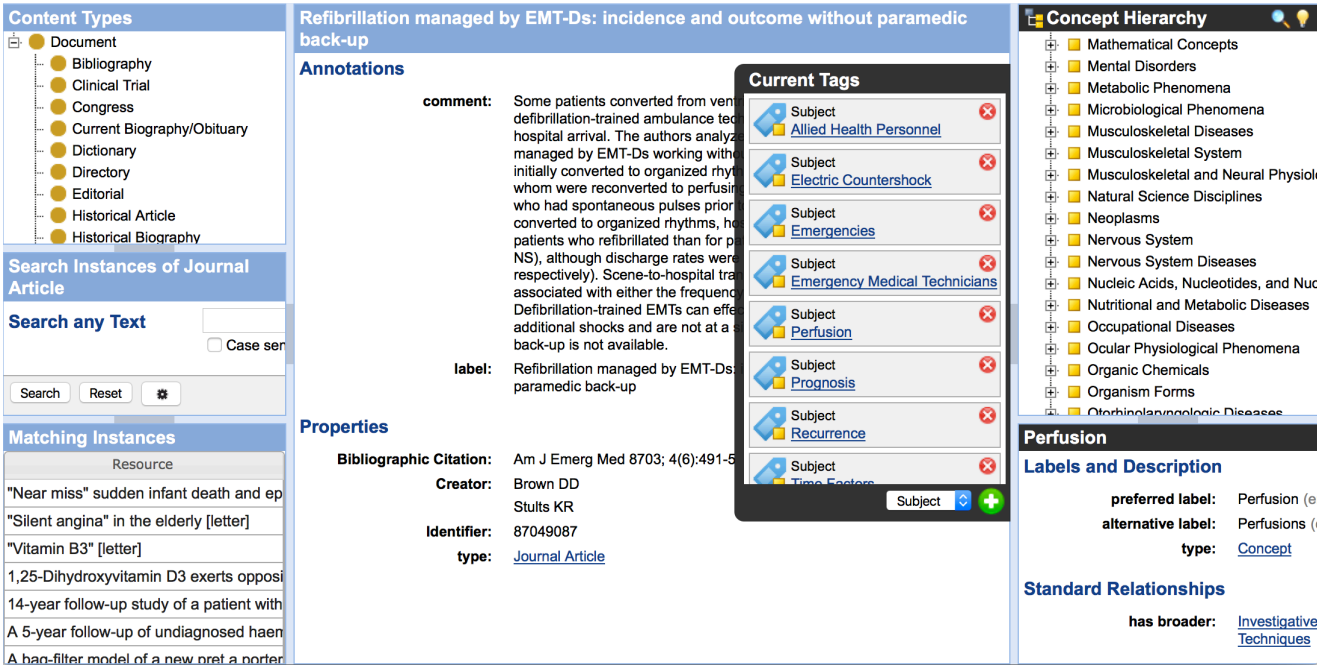
The first column of the report lists tagged content items, clicking the little triangle to the left of any of them (as someone has done with the "Refibrillation managed by" and "Antibiotic prophylaxis" ones above) expands in the second column the list of tags proposed for assignment to those documents, along with tags already in the production copy (if this is the first time AutoClassifier runs on this Content Tag Set, it is likely that tags already in production copy are part of the training data).

Note from the header of this report that the new tags have not been applied to the production tag set, but are instead stored in a working copy named "AutoClassifier Results" plus a date and time stamp.

Clicking the triangle to the left of any content item name in the report shows the tags that have already been or are about to be assigned to it. If the information about a document in the content graph includes a `dc:source` value, its row in AutoClassifier reports will link to the given address, otherwise it will open a new Tagger editing window focused on this specific document.

In the last two columns of the report, each proposed tag can be approved, which commits it to the production tag set, or rejected, which removes it from the working copy. Links in the lower-right do these for all the generated tags.

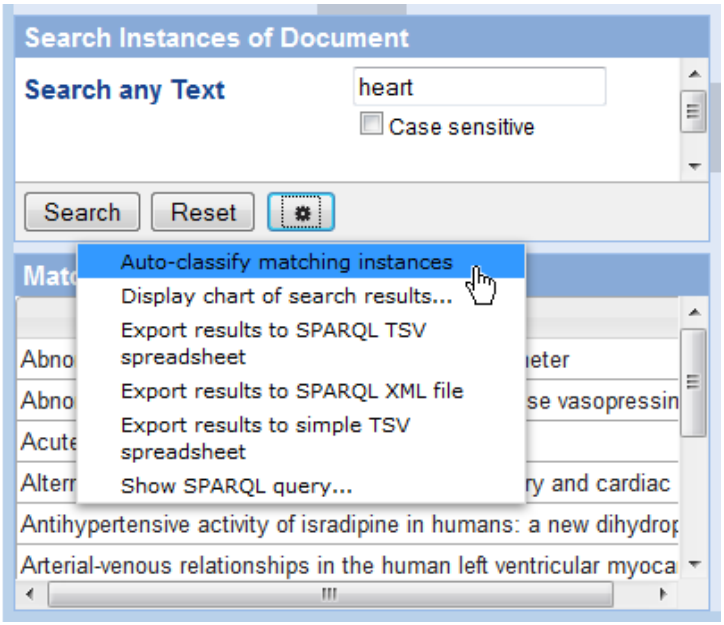
You can also edit the tags added by the AutoClassifier in the regular Tagger interface. In the production copy of the "TReC (1000 Documents)" tag set used for the example above, the content item with the title "Refibrillation managed by EMT-Ds: incidence and outcome without paramedic back-up" has nine tags. Editing this tag set's working copy with the title "AutoClassifier Results, Fri, 27 December 2015, 18:27" shows this content item to have ten tags—the nine from the production copy and the one proposed in the working copy by the AutoClassifier, as shown in the top of the report above:



You can then delete, add, and otherwise edit these tags as you normally would in a tag set working copy.

AutoClassifying a subset of a content set

After performing a search in the lower-left of Tagger's main editing screen, the search results menu includes the option **Auto-classify matching instances**, which lets you can send only the documents that match the search to the AutoClassifier:



This makes it possible to do more focused, faster tagging of documents. After you select this menu option, a message box will inform you that the job is being run and that the results will be available as a new working copy.

You can see a job's progress on the **AutoClassifier** tab, where it will have a "Scope" value of "Search":

Run AutoClassifier

This runs automatic classification on all content resources. Results will be stored for review in a new working copy.

AutoClassifier Jobs

Status	Started	Duration	Scope	Resources	Actions	Results
Completed	2015-12-29 12:45	0h 0min 2s	Search	26	Remove	Results
Completed	2015-12-29 12:44	0h 0min 3s	Search	78	Remove	Results
Completed	2015-12-27 18:27	0h 1min 2s	All	996	Remove	Results

The results report will work the same as if you had run AutoClassifier on all the documents.

Recommended concepts for a single document

After AutoClassifier has been trained, editors can request its suggestions for candidate tags (concepts) for any edited concept instance. On the current item, users can see a list of AC's candidate tags, along with each candidate's confidence score. See the view/edit page for usage details.

AutoClassifier Integration Points

While several entry points are provided to make use of AutoClassifier with different scopes across Tagger, its further integration with external systems extending EDG is possible with two options: a web service and a SPARQL property function.

AutoClassifier Web Services

AutoClassifier can be controlled and invoked through a collection of web APIs. This includes an API function that recommends tags for a submitted text snippet and returns JSON results. Documentation for the AutoClassifier web services is available in the EDG Server Administration area, under *Available Web Services*.

Autoclassifying SPARQL property function

Auto-classification can also be triggered from within a SPARQL query with the property function autotagger:autoClassify, which enables further customization of its behaviour. The example below does showcase it:

```
PREFIX autotagger: <http://evn.topbraidlive.org/autotagger#>
SELECT * WHERE {
    GRAPH ?contentGraph {
        ?cts autotagger:autoClassify (?doc ?concept ?probability)
    }
    GRAPH ?taxon {
        OPTIONAL { ?concept skos:prefLabel ?label }
    }
}
```

Values shall be bound (either hardcoded or in other parts of the query) to the following variables:

- `?contentGraph` is the content graph where unstructured text resides in documents;
- `?cts` is the EDG Content Tag Set project URI, e.g. <urn:x-evn-master:DocsSet>
- `?doc` is an instance of document, e.g. <http://example.org/trec/doc/87049087>. These instances **must** have triples for properties configured as content properties in the Content Tag Set's AutoClassifier configuration
- `?taxon` is the EDG Taxonomy project URI, e.g. <urn:x-evn-master:MyTaxonomy>

Preparing ontologies for use as concept vocabularies with AutoClassifier

AutoClassifier automatically assigns tags to documents. The tags are chosen from the pool of concepts in a concept vocabulary. The concept vocabulary can be a taxonomy expressed in SKOS. In this case, instances of skos:Concept form the pool of possible concepts. Or alternatively, the concept vocabulary can be some other, non-SKOS ontology. In this case, the modeller has some control over the pool of concepts.

The rules used by AutoClassifier to determine the pool of concepts are as follows:

1. If the ontology contains any instances of skos:Concept or any of its subclasses, then these *concept instances* become the pool of concepts.
2. Otherwise, if the ontology's root class is owl:Thing (this is the default), then the *classes* defined in the ontology will become the pool of concepts.
3. Otherwise, any subclasses of the root class will become the pool of concepts.

Some ontologies may use neither skos:Concept nor a class hierarchy to model the concepts to be used as tags by AutoClassifier. Instead, it may use some other class, such as Topic or Location. In this case, it is recommended to make this other class as a subclass of skos:Concept. This will tell AutoClassifier to include the instances of that class in the pool of concepts.

AutoClassifier makes use of preferred and alternate labels as well as concept relationships in the concept vocabulary. It will consider the following properties:

- **Preferred label:** skos:prefLabel and sub-properties, or if that property is not defined, rdfs:label and sub properties
- **Alternate label:** skos:altLabel and skos:hiddenLabel
- **Hierarchy:** skos:broader, skos:narrower, rdfs:subClassOf
- **General relationships:** skos:related, skos:hasTopConcept, owl:sameAs, owl:equivalentClass

Other properties in the ontology can be added to these lists of considered properties by declaring them as subproperties of skos:prefLabel, skos:altLabel, skos:broader and skos:related.

Content tag sets provide a Normalized Concepts exporter that provides a view a simplified SKOS view of the concept taxonomy or ontology. This can be useful for troubleshooting when preparing an ontology. See Normalized Concepts (Troubleshooting) for further information.

Permission Profiles and Workflow

Per standard EDG permission-profile management, the creator of a content tag set has *manager* privileges. Content tag sets can also use working copies. See Workflow Overview for background and details. Also see, **Content Tag Set (Utilities) > Users** and **... > Workflows** for usage.

Managing Tag Set Data

You can import and export tag set RDF data in standard serialization formats. When you use Tagger to tag content with a concept, it is stored in the tag set as actual triples, which are statements expressed using the W3C standard RDF. RDF uses URIs to represent resources such as content resources, tag properties, and the concepts.

For example, if the URI associated with the news story "'Gangnam Style' becomes most watched YouTube video ever" is http://en.wikinews.org/w/index.php?&oldid=1711859, and you use Tagger to tag it as having a Dublin Core subject of "dance" from the IPTC set of news codes, the triple created by Tagger is:

```
{ <http://en.wikinews.org/w/index.php?&oldid=1711859>
  <http://purl.org/dc/elements/1.1/subject>
  <http://cv.iptc.org/newscodes/subjectcode/01006000> }
```

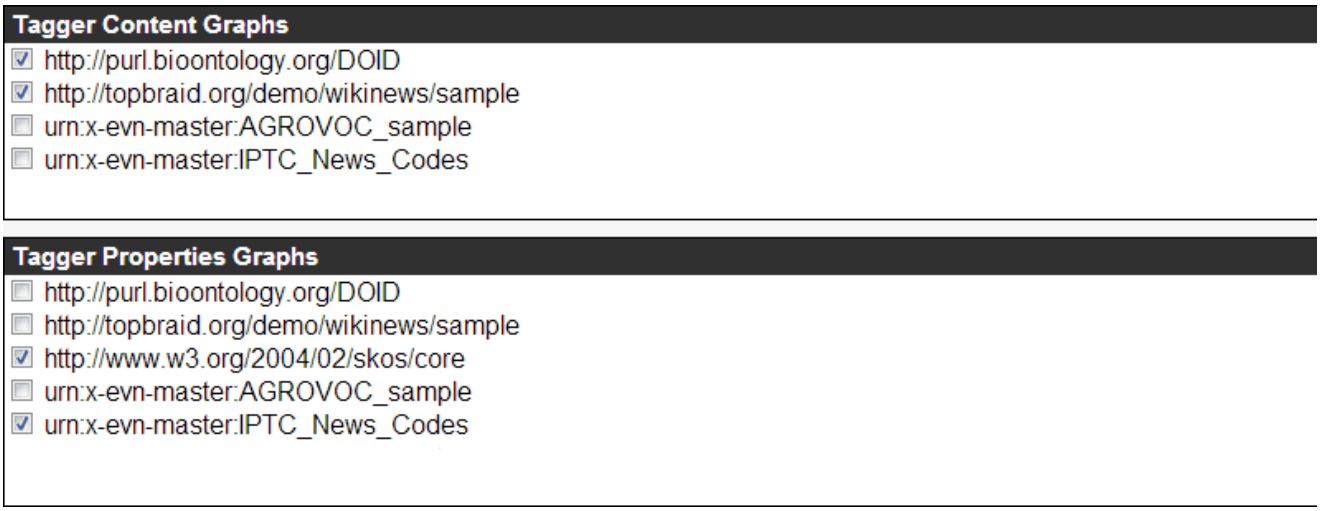
Having the data stored using this standard lets you use it in a variety of applications such as TopBraid EDG and other applications that support the RDF standard. To access a given tag set labeled "my tag set" for use in applications, the URI identifying the tag set itself will be urn:x-evn-master:my_tag_set.

The triples can also be exported in the JSON-LD, Turtle, N-Triple, and RDF/XML serializations of the RDF data model.

Configuring content and property graphs

This section describes how an EDG administrator adds choices to the Content Graph and Tag Property Graph lists that appear when a Tagger user creates a new tag set. (Note that users may also select from viewable ontologies.) It also provides advice on modeling of the graphs that makes the tagging of content resources easier.

To configure the graph choices, first pick Server Administration from the main EDG page. On the "TopBraid Enterprise Data Governance — Server Administration" page that appears, select EDG Configuration Parameters. This leads to the configuration screen:



The configuration screen has three sections:

1. **EDG parameters** is the section where an administrator stores information about back-end storage for EDG. These include the Tagger license number provided when you installed Tagger. See Configuring the persistence technology for new vocabularies and assets for details on the remaining parameters on this section.
2. **Tagger Content Graphs** lists graphs available to display as content graphs in Tagger. Check the ones that you want to appear on the Co ntent Graph drop-down list that Tagger displays when you create a new Content Tag Set.
3. **Tagger Properties Graphs** lists graphs available to display on the Tag Property Graph drop-down list when a Tagger user creates a new Content Tag Set. Check the ones that you want to appear there. Only properties with an *rdfs:range* of *skos:Concept*—or a subclass thereof—will appear in the list of properties in the wizard. For Tagger Properties to display in the Current Tags widget of the Tagger interface, they need an rdfs:range of skos:Concept (or subclass), because the process of tagging is assigning concepts stored using the SKOS standard to content resources.

There are a few notes to keep in mind when setting up content graphs for use by Tagger:

- For the Content Types hierarchy to display properly, the root classes must include assertions that they are subclasses of rdfs:Resource. When a Tagger user selects a class from this hierarchy in the upper-left of the main Tagger screen, as shown in step 1 of Tagging Your Documents, the search panel shown in step 2 of that section will search instances of that class of content to determine which instance titles to display in the lower-left of the screen, as shown in step 3.
- The Tagger interface displays titles of content resources in the content graph using the rdfs:label property, or any subproperty of rdfs: label. If the title (label) for the resources uses a property other than rdfs:label, such as dc:title, define dc:title as a subproperty of rdfs: label. The title will then display properly in Tagger. Additional properties about each content resource will be displayed on the central form when the resource is selected. The form (and AutoClassifier result reports) will display any `dc:source` properties of those graphs as hypertext links to the URLs provided as values, so these are useful for providing easy access to such a resource for the users tagging them.

- The documents should all be typed with some class (or one if its subclasses), and that class should then later be used as the **Root Content Type** when creating a Content Tag Set.
- All properties that occur on any document resource (for example, date or author) should have an rdfs:label, or a subclass of rdfs:label, so that they can be displayed better. This can be achieved by importing an ontology that defines such labels into the content graph.
- Content graphs and tag property graphs should have rdfs:labels for the graph URI so that they are displayed nicely in the dropdowns when creating new Content Tag Sets.

Similarly, a property defined in a Tagger Properties Graph for use by Tagger shall fulfil the conditions below, to be offered for selection when users create new Content Tag Sets:

1. It must be typed as an *rdf:Property* or *owl:ObjectProperty.*
2. Either it must have a declared *rdfs:domain* and *rdfs:range*, or there must be a SHACL property constraint attaching it to a class and constraining its values to another class.
3. Its range, as declared using RDFS or SHACL in 2., must not be a datatype.
4. For use with Content Tag Sets whose concept vocabulary are Taxonomies, its range must be *skos:Concept* or one of its subclasses.
5. Any sub-properties of eligible properties will also be eligible.

Ensuring these conditions may require slight customization of the graphs that you're using; for standard graphs obtained from third parties, this is usually achieved most easily by creating a new graph, importing the standard one, adding customizations to this new one, and then selecting that one on this configuration screen.

For content graphs that will be used by Tagger's AutoClassifier feature, also keep that the actual text content of the documents should be kept in the content graph, in a property of the document resource such as "fullText" or "content". The text in this property, possibly along with other text-containing properties such as title or abstract, will be analyzed by the AutoClassifier, both in training and when generating tag recommendations for documents.

The TopBraid platform used to develop EDG provides tools such as SPARQLMotion for automating conversion of the documents. However, this can also be done using a programming language of the user’s choice. Typically, a conversion script runs on a scheduled basis (for example, nightly) or on demand. After the initial run, a conversion script can process only newly added or changed documents. Once the documents are tagged, information about tags can be provided to the content management and enterprise search systems through export or, in real time, through web services and queries.