

Text Classification API



Text Classification API

DigitalPebble

julien@digitalpebble.com

Introduction

Text Classification (or *Text Categorization*) is the automatic assignment of a label to a span of text, be it an entire document, a paragraph or a sentence.

This is usually achieved by using **Machine Learning** techniques, which require a collection of labelled documents and no human intervention for coding rules or heuristics. Once a Machine Learning algorithm has generated a model of the training data, this model can be used to classify new un-labelled documents automatically.

DigitalPebble has developed an efficient solution for **Text Classification** which is based on **Open Source** Machine Learning implementations. This resource can be easily combined with **Natural Language Processing** modules to improve the quality of the results.

Example

A popular trend in research deals with the identification of **sentiments** in textual documents, for instance by classifying whole documents or sentences as being either *neutral (objective)*, *positive* or *negative*. This is usually achieved by using Machine Learning techniques, such as those provided by our TextClassification API. Corpora such as the movie review corpus¹ are publicly available and are often used to train statistical models for sentiment identification.

The **basic idea** behind text classification is that some span of text (whole document, paragraph, sentence, etc...) is defined by a number of **attributes**, e.g. the words contain in that span. Each span is represented as a multidimensional vector, where each dimension corresponds to an attribute. The set of all attributes are often described as a **lexicon**.

The sentence below, contains 11 distinct words

“All human beings are born free and equal in dignity and rights”

which can be represented in a lexicon such as this one, where each attribute has an index and a form:

<i>Index</i>	<i>Form</i>
1	a
2	all

¹ <http://www.cs.cornell.edu/People/pabo/movie-review-data/>

Text Classification API

<i>Index</i>	<i>Form</i>
3	and
4	are
5	at
...	...
60	dignity
...	...

Representing the example as a vector can be done by setting a numerical value for each attribute defined in the lexicon using a given weighting scheme. The length of the vector is equal to the total number of attributes defined in the lexicon. With a **boolean** scheme, the value of an attribute is 0 or 1 depending on the presence or absence of a feature. The example above would then be represented by the vector:

0	1	1	1	0	...
---	---	---	---	---	-----

The first column has a value of 0 as the corresponding attribute (*a*) does not appear in the sentence, unlike the second one (*all*).

An **occurrence** scheme uses the number of occurrences of an attribute as value, in which case the value of the third dimension in the example above would be 2, as the corresponding word (*and*) appears twice in the sentence. In practice, a **frequency** based scheme is more often used, as it takes into account the size of the textual unit to weigh the values of the attributes. The **tf.idf** scheme also take into account the frequency of an attribute in the entire collection.

A Machine Learning system such as the one used in the Text Classification API is said to be **supervised**, as it requires a collection of labelled documents for the training. The label is a String that the TextClassification API will then assign to new unclassified examples, e.g. *objective*, *positive* or *negative*.

The TextClassification API has a simple method for building documents, which simply takes as argument an array of Strings, and possibly a label (depending on whether this document is used for training or classifying).

In the example given above, the attributes were just the surface forms of the words found in the sentence, however any type of information can be used, as

Text Classification API

long as it can be represented as a String. This can be for instance the lemma of the words, or a mix of the lemma and the part of speech tag of the word. The TextClassificationAPI is available as a **GATE** plugin to simplify the integration of this information.

Text Classification can also be used in a lot of different contexts, such as **filtering** a collection of documents. Imagine the case of an Information Retrieval system based for instance on Lucene, Solr or Nutch. A filtering functionality would allow automatic detection of documents you do not want to present to your users, for example because their content is not suitable. Filtering the data also improves the quality of the result by removing documents from the index which might have a high ranking but be irrelevant, such as junk pages.

Text Classification API

The TextClassification API is implemented in **Java**. Being based on Machine Learning, the API requires a set of documents for **training**, i.e. an array of Strings with a label. The value of the **label** is dependent on the use case and is not restricted by the API, it can also have any number of values.

Once the training is finished, any document sent to the **Classifier** will get a label with possibly a list of scores for all the possible values of the label.

The API implements the standard methods of converting a document into a vector, such as boolean, frequency or tf.idf.

TextClassification API is a **lightweight** component which does not rely on any existing Machine Learning framework. This guarantees a **low memory consumption** and **high CPU performance**, as the API is tailored for Text Classification. Integrating the API in an existing application is straightforward as it does not rely on external libraries, thus avoiding any possible conflict of versions.

One of the major features of the **TextClassification API** is the fact that it is not limited to particular implementation of a **learning algorithm**. It is possible to plug a new implementation if necessary. The API comes with a pre-integrated classifier based on the **libSVM library**², which is both efficient and relatively fast. Kernel methods such as SVM provide state-of-the-art performance and are widely used in academia and industry. The license of **libSVM** allows its use in commercial applications. Another popular algorithm is provided and is based on **SVMLight**³, however its usage is limited to research only.

The Text Classification API is **independent** from a particular storage mechanism and could be used directly on top of any type of document storage (search engine index, database, etc...).

Integration with GATE

GATE (<http://gate.ac.uk>) is an open source framework for Text Analysis implemented in JAVA. The **TextClassification API** is embedded in a GATE Processing Resource, which allows a preprocessing of the documents with different **Natural Language Processing** modules, such as a lemmatizer or a part of speech tagger. Having extra features for representing a text usually improves the quality of the classification. In addition, by integrating the API into GATE, one can benefit from all the **GATE features**, such as document storage, annotation, search, evaluation, etc...

2 <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

3 <http://svmlight.joachims.org/>

Text Classification API

You should see two new PR's in the list of available resources: **TextClassification_Creator** and **TextClassification_Applier**.

The **TextClassification_Creator** takes a number of runtime parameters:

- componentAnnotationType: (e.g. *Token*) Type of annotations to use as attributes of the unit to classify
- componentAnnotationValue: (e.g. *string*) Name of the feature containing the label of the attributes
- inputAnnotationSet: explicit
- minDocThreshold: removes the attributes which occur in less than x documents
- parameters: (e.g. *-s 0 -t 0*) parameters to pass to the libSVM classifier
- textAnnotationType: (e.g. *Sentence*) Type of annotations to classify
- textAnnotationValue: (e.g. *label*) Name of the feature containing the label of the class, for instance "spam" vs "not_spam" etc...
- weighting scheme: [*boolean|occurrences|frequency|tfidf*] method to use for building the vectors

This component has also two

- name: name to use for the PR
- directory: location of the directory used for storing the statistical models. Just create a new empty directory and point to it from the PR.

The **TextClassification_Applier** uses a subset of the parameters defined above.

In a nutshell

- lightweight
- easy to embed in an existing application
- source code available
- plug-in mechanism for new classifiers
- efficient SVM implementation provided
- GATE plug-in

Price

The TextClassification API is **free for research and evaluation**. Any commercial use of the library is subject a licensing agreement.

License

Copyright (c) 2007 DigitalPebble. All rights reserved.

1. Redistribution of the software or its parts is not allowed in any form without prior agreement from DigitalPebble
2. Restrictions apply on the resources used internally by the software, see the licence files of these resources for more information. DigitalPebble is not responsible for any use of the these resources by the licensee.
3. Disclaimer/Limitation of Liability:

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS

Text Classification API

SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.