Maria Skeppstedt, Carita Paradis, Andreas Kerren

PAL, a tool for Pre-annotation and Active Learning

Abstract

Many natural language processing systems rely on machine learning models that are trained on large amounts of manually annotated text data. The lack of sufficient amounts of annotated data is, however, a common obstacle for such systems, since manual annotation of text is often expensive and time-consuming.

The aim of "PAL, a tool for Pre-annotation and Active Learning" is to provide a ready-made package that can be used to simplify annotation and to reduce the amount of annotated data required to train a machine learning classifier. The package provides support for two techniques that have been shown to be successful in previous studies, namely active learning and pre-annotation.

The output of the pre-annotation is provided in the annotation format of the annotation tool BRAT, but PAL is a stand-alone package that can be adapted to other formats.

1 Introduction

Some *artificial* intelligence systems rely heavily on *human* intelligence in the form of training data that is collected by manual annotation. Within the field of natural language processing and computational linguistics, this data collection is typically performed through different kinds of manual annotations of text. Manual text annotation is, however, often an expensive and time-consuming task. The lack of sufficient amounts of manually annotated data that can be used for training machine learning classifiers is therefore a common obstacle.

There are a number of techniques that can be used to reduce the amount of annotated data required for training a machine learning classifier and to simplify the annotation process. Two examples of such techniques are i) the selection of training samples that are informative to the classifier by the use of active learning, and ii) pre-annotation of the data. These techniques are not included as a standard procedure in text annotation projects, presumably because annotation tools typically do not include this functionality. The aim of the development of the PAL package presented here is therefore to take the first step towards changing this standard. The package uses active learning and pre-annotation to facilitate annotation of text chunks. The package is tailored towards the annotation format of text chunks in the annotation tool BRAT (Stenetorp et al., 2012), but it is written as a stand-alone package that can be adapted to other formats.

2 Background

There is a large amount of previous studies on active learning, as well as a number of studies in which pre-annotation has been used, but to the best of our knowledge there are no freely available code packages that can be directly used to apply both of these techniques.

The objective of this paper is to present "PAL, a tool for Pre-annotation and Active Learning", which provides this functionality. Since the tool is built on established techniques, an evaluation of the techniques incorporated in the tool is not within the scope of this paper. Instead, references to previous studies are given.

The main inspiration for PAL was provided by Olsson (2008), who shows the usefulness of active learning for annotation of text chunks and who recommends the use of pre-annotation. The usefulness of active learning was however shown purely by simulation, and no tools for performing active learning or pre-annotation were provided.

2.1 Other approaches for obtaining manually annotated data

Apart from the two techniques mentioned, there are also other possible approaches for obtaining large resources of manually annotated data in a low-resource project. Examples of such techniques are crowd sourcing, community annotation and gamification. However, we argue that the use of these approaches is neither possible nor desirable in all cases.

Manually annotated training data is often collected by the use of crowd sourcing platforms such as Amazon Mechanical Turk. There are several problems associated with this approach (Fort et al., 2011; Archambault et al., 2016). Most important is the ethical issue, i.e., criticism against annotation projects that are carried out by very low-paid annotators. Another concern is the skills required, as it might be difficult to find crowd sourcing annotators with specialised skills, for instance in linguistics, the domain of the text, or annotators that speak the specific language required. This means that the use of crowd sourcing is not a universal solution to the problem of creating enough annotated textual data in a low-resource project.

Another approach is to use community annotation (Uzuner et al., 2010), i.e., to share the annotation task among many annotators by distributing it to a number of researchers in a community. There are disadvantages associated with this approach as well, for instance that it entails a large portion of administrative work and that it is not efficient for difficult annotation tasks, for which initial training phases are required. Another possibility is to gamify the annotation, i.e., to apply game design principles to the task. By making the annotation more fun, there is a potential to gain voluntary annotators (Hanbury et al., 2015). This approach has, for instance, been applied to word sense labelling (Venhuizen et al., 2013) and to identify important expressions in medical case reports (Dumitrache et al., 2013). Yet not all annotation tasks are possible to gamify, at least not effortlessly.

2.2 The type of annotations targeted

The PAL package is meant to be used for annotations of named entities or other types of shorter chunks of text.

The aim could be to create annotated corpora, which in turn can be used for training a classifier to detect the types of text chunks annotated. That is, the machine learning classifier should i) automatically detect interesting tokens (or chunks of interesting tokens) in a text, and ii) automatically categorise these tokens into pre-defined classes. Annotation and detection of named entities such as names of people and places (Nadeau and Sekine, 2007), or of tokens that signal specific functions in a language such as marker words for negation and speculation (Konstantinova et al., 2012) are examples from previous studies.

Another aim could be to create annotated corpora, on which to perform corpus linguistic studies of language phenomena that are expressed through shorter text chunks. Examples of such annotation tasks from previous corpus linguistic studies are annotations of spans of tokens that express attitude (Taboada and Carretero, 2012), evaluation (Fuoli and Hommerberg, 2015), or sensory perceptions (Paradis and Eeg-Olofsson, 2013).

It should be noted that the use cases for annotating these two types of corpora are different. When the aim is to create a corpus for training a machine learning classifier, it is enough to annotate an actively selected subset of the corpus, whereas the entire corpus must be annotated in a corpus study. Otherwise it will not be possible to draw statistically valid conclusions from the annotations. As suggested by Olsson (2008), the methodology when annotating the entire corpus is i) to annotate an actively selected subset of the corpus to achieve a model that can perform pre-annotation with a high accuracy, and ii) to use pre-annotation to annotate the remaining part of the corpus.

To create a corpus for training a classifier, only the first step has to be carried out.

2.3 Pre-annotation

Pre-annotation, or pre-tagging, refers to the procedure to automatically annotate a text corpus by using an existing automatic system and to present these annotations to the human annotator. The human annotator then typically corrects mistakes or omissions made by the automatic system (Chou et al., 2006; Henriksson et al., 2015), or alternatively makes a choice between different options given by the automatic system (Brants and Plaehn, 2000).

The pre-annotation could be built on a rule-based system, such as a system that performs rule-based matching against an existing lexicon (Albright et al., 2013). This approach does not require annotated data. It could also use an existing machine learning system, which may be trained on data from another text domain (Henriksson et al., 2015). A third option is to use pre-annotation based on a machine learning model and to iteratively improve the pre-annotation by retraining it on the new data that is created in the annotation process (Tomanek et al., 2012).

For the task of named entity annotations within the medical domain, lexicon-based pre-annotation has led to an increase in annotation speed, ranging from 14% to 22% per entity for different experiments (Lingren et al., 2014). The same study also included an investigation of whether the pre-annotation functionality biased the annotators, but no bias that stemmed from the pre-annotations could be detected.

2.4 Active learning

Active learning is a technique that is used to reduce the number of training samples that are required to successfully train a machine learning model. The standard method used for manual text annotation is to randomly select which data samples to annotate. For active learning, the training data samples that are estimated to be most useful for the machine learning classifier are instead actively selected from a pool of unlabelled data (Tomanek, 2010).

The estimation of which data samples in the pool that are most useful can, for instance, be based on the level of disagreement among a number of different classifiers (query by committee). That is, the more different classifiers disagree, the more informative is the sample likely to be (Olsson, 2008, pp. 25–29). The estimation can also be based on properties specific to the type of model that is used. When using support vector machines, the unlabelled sample closest to the separating hyperplane of the classifier can be selected, which is the sample that is expected to result in the largest model change when added to the training set (Tomanek, 2010; Tong and Koller, 2002).

Another frequently used method for active selection is *uncertainty sampling*. This technique is built on active selection of the unlabelled training samples that the machine learning model is least certain of how to classify. The approach for a binary classification task is then to select samples for which the classifier has no clear classification preference.

One option for a multi-class classification task is to use the confidence for the most probable class as the measure of uncertainty. However, this option only uses the certainty level of the most probable classification. Thereby, some of the information available is discarded, i.e., the information regarding the certainty levels of the less probable classifications (Settles, 2009). An alternative approach is to base the sample selection for a multi-class classifier on the difference in certainty level between the two most probable classifications. Given c_{p1} as the most probable classification and c_{p2} as the second most probable classification for the observation $\boldsymbol{x}_n$, the margin for measuring the uncertainty of that sample would then be:

$$M_n = P(c_{p1}|\boldsymbol{x}_n) - P(c_{p2}|\boldsymbol{x}_n) \tag{1}$$

Samples with a large margin (M_n) are easy to classify since the classifier is much more certain of the most probable classification than of the second most probable. Samples with a small margin, on the other hand, are difficult to classify. Therefore, in the process of uncertainty-based active selection of training samples, samples with a small margin are preferred (Schein and Ungar, 2007).

This kind of uncertainty selection, based on the confidence difference between the two most probable classifications, is the sampling method that is implemented in the current version of the PAL package.

2.5 Functionality of previous annotation tools

Among 13 annotation tools included in an annotation tool survey from 2012 (Neves and Leser, 2012), only two (JANE and WordFreak) provide functionality for active learning. JANE does not provide a functionality for pre-annotations (Tomanek et al., 2007), and WordFreak only provides pre-annotation performed by pre-defined NLP tools that are independent of the annotation task, e.g., part-of-speech tagging (Morton and LaCivita, 2003).

Among the tools included in the survey, there are others that provide a pre-annotation functionality, typically based on lexicon-matching. There are also more recent annotation tools that either provide the functionality of lexicon-based pre-annotation (Campos et al., 2013) or the functionality of active learning (Kucher et al., 2016).

A disadvantage with pre-annotation based on lexicon-matching is that this approach relies on the existence of a lexicon in the domain of the annotation task. The approach is therefore limited to domains in which such a lexicon exists. There are also tools that pre-annotate without a lexicon, e.g., WebAnno (Yimam et al., 2014). For this tool, the pre-annotations are performed by a machine learning model that is trained on the labelled data provided by the annotator, and the model is iteratively re-trained with more data as more text is manually annotated.

PAL combines the functionality of i) a tool such as WebAnno, in which the pre-annotations are provided by models trained on data that is annotated when using the tool, with ii) the functionality of tools such as JANE and WordFreak, in which data selection through active learning is carried out. We are not aware of any previously constructed, freely available, chunk annotation tools in which these two functionalities are combined.

3 Method

The general functionality of the PAL package consists of training a classifier to select suitable data to annotate from a pool of unlabelled data and to carry out pre-annotation of this data. The pre-annotated data can then be loaded into the BRAT tool, to be revised by the human annotator.

3.1 Active learning and pre-annotation

The PAL package is structured around three different data folders:

1. The *labelled* folder, which contains the data that has been manually labelled.
2. The *unlabelled* folder, which contains the pool of unlabelled data.

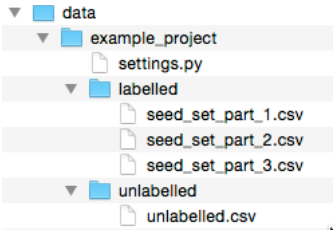

Figure 1: Files and folders that are to be available when a project starts.

3. The *tolabel* folder, to which the pre-annotated, actively selected data is written.

When a project starts, it must contain the data and the folders that are shown in Figure 1. In the *labelled* folder, there must be at least one file containing the seed set of annotated data, which is required to start the active learning process. In the example in Figure 1, the seed set is split into three different files of annotated data. The file that contains the pool of unlabelled data must be called *unlabelled.csv* and must be positioned in the *unlabelled* folder.

When the process of active learning and pre-annotation is run, all .csv-files located in the *labelled* folder are used to train a machine learning model. This model is then employed to perform the active selection of training samples and the pre-annotation. Data samples to be labelled are selected from the file *unlabelled.csv*, pre-annotated and written to the three files that are created in the *tolabel* folder, as shown in Figure 2. These three new files receive a name containing their creation timestamp. The two files named *brat_tolabel_20160928_174414.ann* and *brat_tolabel_201609_28174414.txt* contain the pre-annotated data in BRAT format. That is, those two files are the ones that can be directly imported into BRAT for manual annotation. The file *unlabelled.csv* is also updated and does not any longer contain the data samples that have been selected for annotation. That is, the selected data samples have been removed from the pool of unlabelled data. A copy of the original version of the unlabelled data is also created, the file *unlabelled_20160928_174414.csv* in the *unlabelled* folder.

The data in the .csv-files must be available in tab-separated format in which the data has been tokenised with each token on a separate line. The data must also be segmented into sentences with an empty line signalling a sentence break. See Figure 5 for an example of the data format. The tokens are expected to be labelled according to the BIO-format (Jurafsky and Martin, 2008, pp. 763–764), i.e., a token could be the *B*eginning of, *I*nside or *O*utside of a named entity (or of another type of text chunk). The data samples in the active selection process consist of the sentences. This means that text units in the form of a number of tokens that are separated by an empty line are the units which are selected in the active sampling process.

In the current version of PAL, there are two types of machine learning classifiers to be chosen from, one structured and one non-structured. The following is a step-by-step

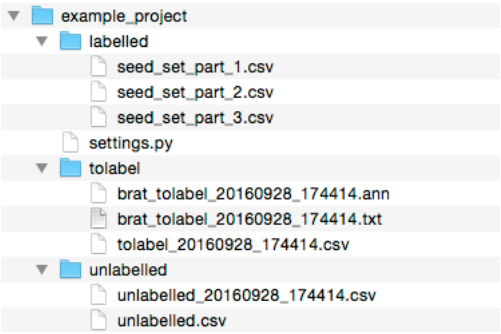

Figure 2: Files and folders created by running active learning and pre-annotation.

description of how the active learning and pre-annotation process is carried out for the non-structured classifier:

1. A machine learning model is trained using the data in the .csv-files that are located in the *labelled* folder.

2. The data located in the *unlabelled* folder is classified with this model. For each of the N tokens in the unlabelled data $(t_n)_{n=1}^N$, there will be an observation in the form of the features for this token, i.e., $(\boldsymbol{x}_n)_{n=1}^N$. For each of these observations, the model provides a probability score for every category in the data.

3. The sentences in the unlabelled data are then ranked according to the uncertainty sampling criterion described in Equation 1, i.e., $M_n = P(c_{p1}|\boldsymbol{x_n}) - P(c_{p2}|\boldsymbol{x_n})$. This is carried out by measuring the difference in probability score between the most likely classification and the second most likely classification for a token. A sentence is represented by the lowest M-value among the tokens it includes, and the sentences are ranked according to this M-value.

4. The k sentences with the k lowest M-values are selected. The value of k (the number of sentences to select in each iteration) is determined by the user in a settings file (see Section 3.2).

5. To achieve a variation among the k samples selected, the same word predicted as belonging to another class than the Outside class is only allowed to occur once among the sentences selected. If such a re-occurring word is present among the sentences selected, the lowest ranked sentence containing this word is removed from the selected set. The sentence that is removed is replaced by the sentence next in rank, i.e., the sentence ranked at position $k + 1$.

6. The k selected sentences are pre-annotated according to the classification made by the machine learning model.

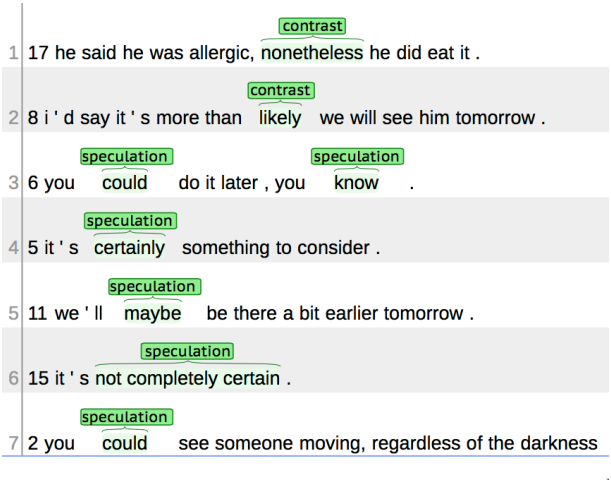

Figure 3: An example of pre-annotated text for the two entity categories consisting of marker words signalling *speculation* and *contrast*, which is imported and displayed in BRAT.

7. The pre-annotated sentences are written to the *tolabel* folder, in a tab separated format and in the BRAT-format.

The same selection procedure is carried out for the structured model, except that the classifications are not made on token level, but on sentence level. M is thereby not computed for the difference in probability score between different token classifications, but between different sentence level classifications. In addition, to reduce the processing-time of the system, only a subset of the alternative sentence classifications are taken into account when computing M.

The pre-annotated sentences can then be opened in BRAT, as shown in Figure 3. This allows the human annotator to carry out the annotation, which then consists of correcting and supplementing the pre-annotations. Figure 4 shows how an incorrect pre-annotation is deleted.[1]

3.2 Configuring and running PAL

To run the package, a directory for the project needs to be created. This directory needs to contain the directories with *labelled* and *unlabelled* data (as shown above), as well

[1] In the BRAT options menu, different annotation modes can be selected. The recommendation is to not use the *Careful* mode, since this mode prompts the user for a confirmation every time an annotation is removed.

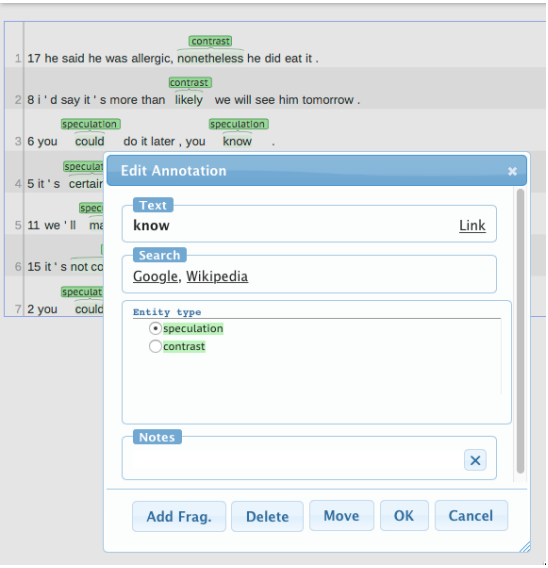

Figure 4: The annotator uses BRAT to delete the incorrect pre-annotations.

as a *settings.py* file with configuration information. The most important configuration parameters are described here.[2]

```
# Classes to include with their prefix (B or I).
# Classes, apart from "O" the outside-class, not in this list, will be ignored
minority_classes = ["B-speculation", "I-speculation", "B-contrast", "I-contrast"]

# Number of sentences to be actively selected and pre-annotated in each round
# (referred to as k above)
nr_of_samples = 20

# Type of model to use
model_type = NonStructuredLogisticRegression

# The context around the current word to
# include when training the classifiers
number_of_previous_words = 2
number_of_following_words = 1

# A cut-off for the number of occurrences of a token in
# the data for it to be included as a feature (current and context-token)
min_df_current = 2
min_df_context = 2
```

[2] See the readme-file of the package, for a full description of additional configuration parameters.

The *settings.py* file needs to include at least 1) the classes that represent named entities or other types of relevant text chunks that are to be included in the active learning and pre-annotation process, 2) the number of samples to select in each iteration of the active learning process (referred to as k above), 3) the type of model (structured or non-structured), 4) the number of context tokens to include when constructing the feature vector for a token, and 5) the minimum number of occurrences of a token (and a context token) for it to be included in the vector model constructed for the tokens occurring in the data (see Section 3.3).

The functionality for selecting and pre-annotating samples can then be run with the following command, in which the location of the data and the settings file is specified:

```
python active_learning_preannotation.py --project=data.example_project
```

The data files that are created by this command (the .txt and the .ann files) can then be moved to the BRAT *data* folder to make it available for the annotator to select for annotation.[3]

When the annotator has finished the annotation/correction of the pre-annotated files, they can be transformed back to a tab separated format and positioned in the folder with annotated data with the following command (on one line, without line breaks):

```
python transform_from_brat_format.py
 --project=data.example_project
 --annotated=my_brat_path/data/example_project/brat_tolabel_20160928_174414
```

The name *brat_tolabel_20160928_174414* is the path to the .txt-file and to the human-annotated version of the .ann-file, but without the suffices. The two files must be positioned in the same folder.

3.3 Models and feature representations

The PAL package is written in Python 3. It is developed to be used in a Unix environment, and it has been tested on Ubuntu 12.04.5.

There are currently two types of machine learning models that are supported by PAL, a logistic regression model (Bishop, 2006) and a conditional random fields model (Sutton and McCallum, 2006). The logistic regression model is built on the LogisticRegression class of the Scikit-learn library. This library includes, among many other components, Python classes for performing non-structured prediction (Pedregosa et al., 2011). The conditional random fields model is built on the ChainCRF class that is available in the PyStruct library, which is a library that contains a number of Python classes for performing structured prediction. A structured prediction model takes the structure of the output labels into account for training the classifiers and performing the predictions (Müller, 2013). The ChainCRF model is an implementation of a linear-chain conditional random fields model, in which an output variable is only directly dependent on its immediate neighbours (Sutton and McCallum, 2006, p. 9).

[3] See the BRAT user documentation for more information about this.

n (Token number)	Word	Label	
..	..	..	
21	it	O	
22	’	O	
23	**s**	O	← previous context token
24	**not**	B-speculation	← previous context token
25	**completely**	I-speculation	← current token
26	**sure**	I-speculation	← following context token
27	.	O	
28	Another	O	
29	sentence	O	
..	..	..	

$\boldsymbol{x}_{25} =$ [Current token: 0 1 0 0 0 0 | Context -2: 0 0 1 0 0 | Context -1: 0 1 0 0 0 | Context +1: 0 0 0 1]

Figure 5: An example that shows how the feature vector for token number 25 is constructed, i.e., $\boldsymbol{x}_{25}$. In this example, the user has configured PAL to include a context in the form of two preceding tokens and one following token. That is, *number_of_previous_words*=2 and *number_of_following_words*=1. The words used for constructing the feature vector are marked with bold and colours. The vector representations of these words are concatenated to form the feature vector $\boldsymbol{x}_{25}$. See Figure 6 for an example of how vector representations for words are constructed.

What features to use for training and predicting the label of a token, i.e., the observation vector $\boldsymbol{x}_n$, is decided by the parameters *number_of_previous_words*, *number_of_following_words*, *min_df_current*, and *min_df_context* in the settings-file. Two different matrices that represent the words in the labelled data are created, using the CountVectorizer class of the Scikit-learn library. One of the matrices contains the vector representation of the tokens for which a label is to be predicted and the other matrix contains the vector representation of the context tokens (Figure 5). Each word that occurs at least *min_df_current* times in the labelled corpus receives its unique vector representation in the matrix for the current tokens and each word that occurs at least *min_df_context* times in the labelled data receives a unique vector representation in the matrix for context tokens. Such a vector representation is exemplified in Figure 6.

The feature vector to use for training and prediction (the observation vector $\boldsymbol{x}_n$) is constructed by concatenating the vector representation of the current token with the vector representations of the tokens surrounding it. How many surrounding tokens to include is decided by the setting parameters *number_of_previous_words* and *number_of_following_words* (See Figure 5).

Word	From labelled corpus
be	[1 0 0 0 0]
completely	[0 1 0 0 0]
not	[0 0 1 0 0]
s	[0 0 0 1 0]
sure	[0 0 0 0 1]

Figure 6: Matrix of vector representations of words in a very small, hypothetical corpus. In this corpus, only five words occur frequent enough to be included in the matrix. The value of "frequent enough" is decided by the user configuration, i.e., by the variables *min_df_current* (if the matrix is a representation of words when they occur as the current token) and *min_df_context* (if the matrix is a representation of words when they occur as the context tokens).

3.4 Incorporation of unsupervised approaches

Information harvested in an unsupervised fashion from unlabelled corpora has for a long time (Miller et al., 2004; Freitag, 2004) been used as classifier features for improving the performance of named entity recognition and other types of chunk classification tasks. For instance, information from Brown clustering has been used for biomedical entity recognition (de Bruijn et al., 2011), predictive class bigram model clustering for general named entity recognition in several languages (Täckström et al., 2012), Random Indexing for named entity recognition in clinical text (Henriksson, 2015), and different kinds of word embeddings have been used for recognising opinion targets (Liu et al., 2015).

The PAL package has support for adding unsupervised features through the Gensim library (Řehůřek and Sojka, 2010). PAL can be configured to append external vector representations from Gensim to the feature vectors. This functionality has so far only been tested with vectors from an out-of-the-box word2vec model for English, which has been trained on Google news (Mikolov, 2013; Mikolov et al., 2013).

3.5 Availability

The tool is freely available to clone or download from GitHub at:

`https://github.com/mariask2/PAL-A-tool-for-Pre-annotation-and-Active-Learning`

PAL makes use of three external libraries, which all are freely available:[4]

1. For training the machine learning models for performing structured prediction, the PyStruct library is used (Müller and Behnke, 2014).

2. For performing non-structured prediction and for vectorising the data, the LogisticRegression and the CountVectorizer classes of the Scikit-learn library are used (Pedregosa et al., 2011).

[4] See the readme-file of PAL for a description of how to install them.

3. For incorporating unsupervised features, the Gensim library is used (Řehůřek and Sojka, 2010).

These libraries are in turn dependent on cvxopt, numpy and scipy.

4 Discussion and future work

Although PAL incorporates a number of functions to facilitate annotation, the tool can still be improved and extended. There are, for instance, possible limitations of the design decisions applied and potential problems with the general approach of pre-annotation.

4.1 Design decisions

One potential issue is the level of technical knowledge that is required to use PAL. No programming skills are required to configure and run PAL, but knowledge of how to use a Unix command line and how to modify a settings file is required. These two skills are, however, also required for configuring the BRAT annotation tool. Therefore, since the target users of PAL are those that typically would configure BRAT, or a similar annotation tool, the technical skills required for using PAL should not be a problem for its target users. There are other annotation tools that incorporate active learning, which provide a graphical user interface by which settings can be altered and by which actively selected annotation batches can be generated (Kucher et al., 2016). To add such a functionality to PAL would extend its group of target users.

Another design decision is that PAL is constructed as a stand-alone tool for pre-annotation and active learning and that it is not fully integrated with an annotation tool. This design decision has the advantage of making the package more flexible to adapt to other annotation tools. The only modification that has to be carried out is the transformation of the output files that are generated by PAL to the format of the other annotation tool.

4.2 Potential problems with the use of pre-annotation

As stated above, there are studies in which pre-annotation has been shown to increase the speed of annotation without introducing a bias (Lingren et al., 2014) and annotation studies in which pre-annotation has been applied successfully (Albright et al., 2013). In contrast, other studies have shown that pre-annotation has had no effect on the time taken (South et al., 2014) or that pre-annotation has slowed down the annotators due to the low quality of the pre-annotations provided (Ogren et al., 2008). There are also cases where high-quality pre-annotations have been shown to be successful in general, with an increase in annotation speed and general annotation quality, but where the pre-annotations have reduced the annotator attention and resulted in a bias (Fort and Sagot, 2010).

The first of these two potential problems, i.e., that low quality pre-annotations are slowing down the annotator, is not a limitation for the functionality provided by PAL. If

the annotator does not consider the pre-annotations to be beneficial, the pre-annotation functionality can be switched off by removing the content of the .ann-file created by PAL. The quality of the pre-annotations might, for instance, be too low in the early stages of the annotation process when there is only a limited amount of annotated data available for training the machine learning model. The methodology proposed by Olsson (2008) is to start the process of annotating for active learning without providing pre-annotation and to introduce the functionality of pre-annotation when the model has reached a certain level of performance. The risk of providing pre-annotations that are not useful would then be minimised.

The usefulness of the pre-annotation functionality provided by PAL has so far been tested in an initial annotation project. A corpus of 6,000 sentences was annotated for text chunks signalling topic-independent expressions of agreement and disagreement, according to a previously defined annotation scheme (Skeppstedt et al., 2016). A seed set that contained 1,500 sentences was first annotated with BRAT without the use of the pre-annotation functionality. The remaining sentences were then annotated with the help of pre-annotation. The quality of the pre-annotation was in this case assessed by the annotator as high enough to be useful.

The issue that pre-annotation might introduce a bias is a more serious problem. This problem is, however, not unique to PAL, but is a problem shared by all tools that provide pre-annotations. We have previously published an outline for an annotation interface for text chunk annotation that would alleviate this problem (Skeppstedt, 2013). The idea is to always provide two alternative pre-annotations to the annotator, i.e., the two pre-annotations that are assessed as most probable by the pre-annotation functionality. No information will however be given to the annotator regarding which of these two possible pre-annotation choices is considered the most probable one. The annotator will thereby always be forced to make an active choice, and will not be biased towards an annotation decision made by the pre-annotation functionality.

The next step for the PAL tool will be to implement and evaluate this functionality.

4.3 Additional future work

Future work also includes the addition of functionality for creating the initial seed set that is required for starting the active learning process. In the current version of PAL, it is assumed that random selection of training data is applied to create this initial seed set. While this procedure is the standard technique used when applying active learning, it should be mentioned that there is research on other methods for creating a seed set. Knowledge of existing vocabularies can be leveraged, for example, by selecting seed set samples that contain instances of these vocabulary terms (Tomanek et al., 2007). There are also techniques that use unsupervised machine learning approaches to create the seed set. For instance, by clustering the unlabelled data and constructing the seed set by selecting data samples from different clusters, a more diverse seed set can be achieved (Debarr and Wechsler, 2009). These types of techniques can also be used in

the process of active selection of data samples (Symons et al., 2006; Settles and Craven, 2008)

Another branch of active learning research is concerned with techniques for estimating when the addition of more training data samples no longer contributes to an increase in classifier performance. There are a number of techniques for performing such an estimation (Olsson, 2008), but none of them are included in the functionality currently provided by PAL.

Apart from uncertainty sampling, there are also many other methods for performing active learning, as mentioned in the background. Although these methods are not included in the current version of PAL, its code structure makes it easy to add other PyStruct and Scikit-learn classifiers, as well as other methods for performing active learning. We hope that this will inspire others to further develop the package with classifiers and active learning methods that suit their annotation and classification tasks.

Yet another topic for possible future work includes a practical evaluation of the benefit for the user of the functionality provided by the PAL package. As previously mentioned, the package has been used for annotating text chunks signalling agreement and disagreement in a corpus of 6,000 sentences. The functionality provided by PAL was found useful in this context, but the annotation was performed by the developer of PAL, which makes the assessment of PAL's usefulness likely to be biased (Munzner, 2008). Instead, usability studies of PAL should be performed with independent annotators and for different types of annotation tasks.

5 Conclusion

Although there are a number of techniques for simplifying text annotation and for reducing the amount of data required for training a machine learning classifier, these techniques are not included as a standard procedure in text annotation projects. The reason may be that they are typically not included by default in annotation tools.

The aim of "PAL, a tool for Pre-annotation and Active Learning" is to take the first step towards changing this standard. The package provides a functionality that includes pre-annotation and active selection of training data.

The output of the pre-annotation is provided in the annotation format of the annotation tool BRAT, but it is a stand-alone package that can be adapted to other formats. The tool is freely available to clone or download from GitHub at:
https://github.com/mariask2/PAL-A-tool-for-Pre-annotation-and-Active-Learning.

Acknowledgements

This work was funded by the StaViCTA project, framework grant "the Digitized Society – Past, Present, and Future" with No. 2012-5659 from the Swedish Research Council (Vetenskapsrådet).

References

Albright, D., Lanfranchi, A., Fredriksen, A., Styler, 4th, W. F., Warner, C., Hwang, J. D., Choi, J. D., Dligach, D., Nielsen, R. D., Martin, J., Ward, W., Palmer, M., and Savova, G. K. (2013). Towards comprehensive syntactic and semantic annotations of the clinical narrative. *J Am Med Inform Assoc*.

Archambault, D., Hoßfeld, T., and Purchase, H. C. (2016). Crowdsourcing and Human-Centred Experiments (Dagstuhl Seminar 15481). *Dagstuhl Reports*, 5(11):103–126.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer, New York, NY.

Brants, T. and Plaehn, O. (2000). Interactive corpus annotation. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, Paris, France. European Language Resources Association (ELRA).

Campos, D., Lourencço, J., Nunes, T., Vitorino, R., Domingues, P., Matos, S., and Oliveira, J. L. (2013). Egas – collaborative biomedical annotation as a service. In *Proceedings of the Fourth BioCreative Challenge Evaluation Workshop*, volume 1, pages 254–259. http://www.biocreative.org/media/store/files/2013/ProceedingsBioCreativeIV_vol1_.pdf.

Chou, W.-c., Tzong-han Tsai, R., and Su, Y.-s. (2006). A semi-automatic method for annotating a biomedical proposition bank. In *ACL Workshop on Frontiers in Linguistically Annotated Corpora*, pages 5–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

de Bruijn, B., Cherry, C., Kiritchenko, S., Martin, J. D., and Zhu, X. (2011). Machine-learned solutions for three stages of clinical information extraction: the state of the art at i2b2 2010. *J Am Med Inform Assoc*, 18(5):557–562.

Debarr, D. and Wechsler, H. (2009). Spam detection using clustering, random forests and active learning. In *CEAS 2009 – Sixth Conference on Email and Anti-Spam, Mountain View*, pages 16–17, Mountain View, California USA.

Dumitrache, A., Aroyo, L., Welty, C., Sips, R., and Levas, A. (2013). "Dr. Detective": combining gamification techniques and crowdsourcing to create a gold standard in medical text. In *Proceedings of the 1st International Workshop on Crowdsourcing the Semantic Web, Sydney, Australia, October 19, 2013*, pages 16–31, Aachen, Germany. CEUR-WS.org.

Fort, K., Adda, G., Sagot, B., and Mariani, J. (2011). Crowdsourcing for language resource development: Critical analysis of Amazon Mechanical Turk overpowering use. In *LTC, 5th Language and Technology Conference*, Poznan, Poland. https://hal.archives-ouvertes.fr/hal-00648187.

Fort, K. and Sagot, B. (2010). Influence of pre-annotation on pos-tagged corpus development. In *Proceedings of the Fourth Linguistic Annotation Workshop*, LAW IV '10, pages 56–63, Stroudsburg, PA, USA. Association for Computational Linguistics.

Freitag, D. (2004). Trained named entity recognition using distributional clusters. In *Conference on Empirical Methods in Natural Language Processing*, pages 262–269, Association for Computational Linguistics. Stroudsburg, PA, USA.

Fuoli, M. and Hommerberg, C. (2015). Optimising transparency, reliability and replicability: annotation principles and inter-coder agreement in the quantification of evaluative expressions. *Corpora*, 10(3):315–349.

Hanbury, A., Kazai, G., Rauber, A., and Fuhr, N., editors (2015). *Second International Workshop on Gamification for Information Retrieval*, Berlin/Heidelberg, Germany. Springer Verlag. Lecture Notes in Computer Science vol. 9022.

Henriksson, A. (2015). Learning multiple distributed prototypes of semantic categories for named entity recognition. *Int. J. Data Min. Bioinformatics*, 13(4):395–411.

Henriksson, A., Kvist, M., Dalianis, H., and Duneld, M. (2015). Identifying adverse drug event information in clinical notes with distributional semantic representations of context. *J Biomed Inform*, 57:333–49.

Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall, Saddle River, NJ, USA, second edition.

Konstantinova, N., de Sousa, S. C., Cruz, N. P., Maña, M. J., Taboada, M., and Mitkov, R. (2012). A review corpus annotated for negation, speculation and their scope. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 3190–3195, Paris, France. European Language Resources Association (ELRA).

Kucher, K., Kerren, A., Paradis, C., and Sahlgren, M. (2016). Visual Analysis of Text Annotations for Stance Classification with ALVA. In Isenberg, T. and Sadlo, F., editors, *EuroVis 2016 - Posters*. The Eurographics Association.

Lingren, T., Deleger, L., Molnar, K., Zhai, H., Meinzen-Derr, J., Kaiser, M., Stoutenborough, L., Li, Q., and Solti, I. (2014). Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *J Am Med Inform Assoc*, 21(3):406–13.

Liu, P., Joty, S., and Meng, H. (2015). Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mikolov, T. (2013). https://code.google.com/archive/p/word2vec/.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Miller, S., Guinness, J., and Zamanian, A. (2004). Name tagging with word clusters and discriminative training. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL HLT)*, pages 337–342, Stroudsburg, PA, USA. Association for Computational Linguistics.

Morton, T. and LaCivita, J. (2003). Wordfreak: An open tool for linguistic annotation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Demonstrations - Volume 4 (NAACL HLT)*, pages 17–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Müller, A. C. (2013). What is structured learning? https://pystruct.github.io/intro.html (Accessed 2016-09-29).

Müller, A. C. and Behnke, S. (2014). PyStruct - learning structured prediction in Python. *Journal of Machine Learning Research*, 15:2055–2060.

Munzner, T. (2008). Process and pitfalls in writing information visualization research papers. In Kerren, A., Stasko, J. T., Fekete, J.-D., and North, C., editors, *Information Visualization: Human-Centered Issues and Perspectives*, pages 134–153. Springer, Berlin/Heidelberg, Germany.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Neves, M. and Leser, U. (2012). A survey on annotation tools for the biomedical literature. *Briefings in Bioinformatics*, 15(2):327–340.

Ogren, P., Savova, G., and Chute, C. (2008). Constructing evaluation corpora for automated clinical named entity recognition. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 3143–3149, Paris, France. European Language Resources Association (ELRA).

Olsson, F. (2008). *Bootstrapping Named Entity Annotation by Means of Active Machine Learning*. PhD thesis, University of Gothenburg. Faculty of Arts.

Paradis, C. and Eeg-Olofsson, M. (2013). Describing sensory experience: The genre of wine reviews. *Metaphor and Symbol*, 28(1):22–40.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Paris, France. European Language Resources Association (ELRA).

Schein, A. I. and Ungar, L. H. (2007). Active learning for logistic regression: an evaluation. *Mach. Learn.*, 68(3):235–265.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report #1648, University of Wisconsin–Madison, http://research.cs.wisc.edu/techreports/2009/TR1648.pdf.

Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (EMNLP), pages 1070–1079, Stroudsburg, PA, USA. Association for Computational Linguistics.

Skeppstedt, M. (2013). Annotating named entities in clinical text by combining pre-annotation and active learning. In *Proceedings of the Student Research Workshop at ACL*, pages 74–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Skeppstedt, M., Sahlgren, M., Paradis, C., and Kerren, A. (2016). Unshared task: (Dis)agreement in online debates. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, Stroudsburg, PA, USA. Association for Computational Linguistics.

South, B. R., Mowery, D., Suo, Y., Leng, J., Ferrández, Ó., Meystre, S. M., and Chapman, W. W. (2014). Evaluating the effects of machine pre-annotation and an interactive annotation interface on manual de-identification of clinical text. *J Biomed Inform*, 50:162–72.

Stenetorp, P., Pyysalo, S., Topic, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: a web-based tool for NLP-assisted text annotation. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sutton, C. and McCallum, A. (2006). An introduction to conditional random fields for relational learning. In Getoor, L. and Taskar, B., editors, *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA, USA.

Symons, C. T., Samatova, N. F., Krishnamurthy, R., Park, B. H., Umar, T., Buttler, D., Critchlow, T., and Hysom, D. (2006). Multi-criterion active learning in conditional random fields. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI '06, pages 323–331, Washington, DC, USA. IEEE Computer Society.

Taboada, M. and Carretero, M. (2012). Contrastive analyses of evaluation in text: Key issues in the design of an annotation system for attitude applicable to consumer reviews in English and Spanish. *Linguistics and the Human Sciences*, 6(1-3):275–295.

Täckström, O., McDonald, R., and Uszkoreit, J. (2012). Cross-lingual Word Clusters for Direct Transfer of Linguistic Structure. In *Proceedings of the North American Chapter of the Associ- ation for Computational Linguistics on Human Language Technology (NAACL HLT)*, pages 477–487, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomanek, K. (2010). *Resource-Aware Annotation through Active Learning*. PhD thesis, Technical University of Dortmund.

Tomanek, K., Daumke, P., Enders, F., Huber, J., Theres, K., and Müller, M. (2012). An interactive de-identification-system. In *Proceedings of SMBM 2012 - The 5th International Symposium on Semantic Mining in Biomedicine*, pages 82–86, Zürich, Switzerland. Institute of Computational Linguistics, University of Zurich.

Tomanek, K., Wermter, J., and Hahn, U. (2007). Efficient annotation with the Jena ANnotation Environment (JANE). In *Proceedings of the Linguistic Annotation Workshop*, pages 9–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tong, S. and Koller, D. (2002). Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66.

Uzuner, Ö., Solti, I., Xia, F., and Cadag, E. (2010). Community annotation experiment for ground truth generation for the i2b2 medication challenge. *J Am Med Inform Assoc*, 17(5):519–523.

Venhuizen, N., Basile, V., Evang, K., and Bos, J. (2013). Gamification for word sense labeling. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*, pages 397–403, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yimam, M. S., Biemann, C., Eckart de Castilho, R., and Gurevych, I. (2014). Automatic annotation suggestions and custom annotation layers in WebAnno. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 91–96, Stroudsburg, PA, USA. Association for Computational Linguistics.