**SemEval 2016, Task 5**

**Aspect Based Sentiment Analysis (ABSA)**

**Baselines**

1. **Baselines Description**

We have implemented a baseline system for each slot (subtasks 1&2) of ABSA 2016.

**Subtask 1, Slot 1: Aspect Category (Entity and Attribute).** For category extraction (at sentence level) a Support Vector Machine (SVM) model with linear kernel is learnt. In particular, *n* bag-of-words (BOW) features[[1]](#footnote-1) are extracted from the respective sentence of each <category, target, polarity> tuple that is encountered in the training data. As a label for the feature vector the category value (e.g. SERVICE#GENERAL) of the tuple is used. Similarly, for each test sentence *s* one BOW feature vector is built and the trained SVM model is used to predict the probabilities of assigning each possible category to *s* (e.g. {SERVICE#GENERAL, 0.2}, {RESTAURANT#GENERAL, 0.4},…,{FOOD#STYLE\_OPTIONS, 0.4}). Then, a threshold t is used to decide which of them will be assigned to *s*[[2]](#footnote-2).

**Subtask 2, Slot 1: Aspect Category (Entity and Attribute).** The predicted sentence-level tuples (Subtask 1, Slot 1) are copied to text level and duplicates are removed.

**Subtask 1, Slot 2: Opinion Target Expression (OTE).** This baseline uses the training reviews to create for each category *c* (e.g. SERVICE#GENERAL) a list of targets to which it is linked to. Then, given a test sentence *s* and a category *c*, the baseline finds the first occurrence in *s* of each target encountered in *c*’s list. Finally, the target slot of *c* is filled with the first target occurrence in *s*. If no target occurrences are found, the slot is assigned the value NULL.

**Subtask 1, Slot 3: Sentiment Polarity.** For polarity prediction we train an SVM classifier with linear kernel. Again, as in Slot 1, *n* BOW features are extracted from the respective sentence of each <category, target, polarity> tuple of the training data. In addition, a feature that indicates the category of the tuple is used (*n*+1 features in total). As a label for the extracted feature vector the corresponding polarity value is used (positive, negative, neutral, conflict). Then, for each opinion tuple (category, target, -) of a test sentence *s* a feature vector is built and it is classified using the learnt SVM model.

**Subtask 2, Slot 3: Sentiment Polarity:** For each text-level category c the baseline traverses the predicted sentence-level tuples of the same category and counts the respective polarity labels {positive, negative neutral}. Finally, the polarity label with the highest frequency is assigned to the text-level category c. If there are not any sentence-level tuples of the same category the polarity label is determined based on all tuples regardless of the category.

1. **Baselines package**

The baseline systems and evaluation scripts are available for download from SemEval-2016 Task 5 website[[3]](#footnote-3). They are implemented in java[[4]](#footnote-4) and integrated into a Linux shell script (absa16.sh). The script uses the LibSVM software[[5]](#footnote-5) for SVM training and prediction.

**Installation and running**:

To install the package extract BaselinesRel.zip to a directory (e.g. BaselinesRel). Then, open a terminal and move to the top level directory (BaselinesRel) of the package. Move to libsvm-3.18 directory by typing “cd libsvm-3.18” and run “make” to build the “svm-train” and “svm-predict” programs. After this, return to the top level directory by typing “cd ..”. Before running them (e.g. bash absa16.sh “conf file”) you have to open the respective configuration file (e.g. cfgAR\_Hotels\_absa16.cfg) and set the required parameters as it is described below.

**SB1:** It should be assigned the name of the xml file that contains the training data of a domain (e.g. laptops or restaurants) for subtask1 (e.g. ABSA-16\_Rest\_Train\_Final.xml).

**SB1Test:** Test data for subtask 1.

**SB2**: Training data for subtask2

**SB2Test:** Test data for subtask 2.

**dom:** It should be assigned the domain that the **SB1** file corresponds to. e.g. dom=rest for restaurants and dom=lapt for laptops and “hote”, “phns”, “came”, “telco” for the remaining domains.

**thr:** It sets the threshold for assigning categories in a sentence. e.g. thr=0.2.

**sfl:** It indicates whether the reviews of the **SB1** file will be shuffled or not before splitting them into parts and generating the corresponding files. **slf** should be set to 0 for not shuffling or to 1 for shuffling. A constant seed is used so that the same review order is generated in every run.

**xva**: It indicates whether a cross validation will be performed on the input data (**SB1** file). For running cross validation **xva** should be set to 1. For just splitting the **SB1** file in a training and test part it should be set to 0.

**fld**: It specifies the number of chunks that the reviews of the **SB1** file will be split into. These chunks contain whole reviews. If **xva** is set to 0 then one of chunks will be used for testing and the rest fld-1 for training. The chunk that will be used for testing is specified by the **partIdx** parameter that is described below. Otherwise, if **xva** is set to 1 the script will run **fld** rounds (iterations), in each round one of generated chunks is used for testing and the other fld-1 for training.

**partIdx**: When **xva** is set to 0 it partIdx specifies the chunk index that will be used for testing. For example, if fld value is 10 then it can take a value from 0 to 9.

**ftr**: It specifies the number of BOW features that will be used to create the SVM training and testing vectors.

**ttd:** It specifies the name of the folder where the SVM training and testing files will be written. In this folder they also are also stored the input xml files as well the outputs of the baselines. This folder should be placed in the top level directory of the package.

**OTE:** Specifies if Slot2 annotations are required for the subtask 1 dataset.

**NOTE:** We have created the required configuration files (see below) for all datasets and a script that runs the baselines for all of them (scriptRunAll.sh)*.*

cfgAR\_Hotels\_absa16.conf, cfgCH\_Cam\_absa16.conf, cfgCH\_Phn\_absa16.conf

cfgDU\_Phn\_absa16.conf, cfgDU\_Rest\_absa16.conf, cfgEN\_Lapt\_absa16.conf

cfgEN\_Rest\_absa16.conf, cfgFR\_Rest\_absa16.conf, cfgRU\_Rest\_absa16.conf

cfgSP\_Rest\_absa16.conf, cfgTU\_Rest\_absa16.conf, cfgTU\_Telc\_absa16

**Inputs and outputs description:**

**Subtask 1**

When **xva** parameter is set to 0 the script operates as follows:

The **SB1** file reviews are split and the files that are listed below are generated. The contents of these files are determined from the values of **fld** and **partIdx** parameters**.**

* tr.xml: It contains the training reviews with the human annotations.
* teCln.xml: It contains the testing reviews without any human annotations.
* teCln.PrdAspTrg.xml: It contains the testing reviews along with the gold category and target annotations.
* teGld.xml. It contains the testing reviews and all the corresponding gold annotations (category, target, polarity).

Then, in a phase A the script predicts the category (Slot 1) and target (Slot 2) annotations for the sentences of **teCln.xml** and stores the result to **teCln.PrdAspTrg.xml**. Subsequently, in a phase B it predicts the polarity of each gold (category, target) tuple that is stored in **teGldAspTrg.xml** and generates **teGldAspTrg.PrdPol.xml[[6]](#footnote-6).** Finally, the outputs (**teCln.PrdAspTrg.xml, teGldAspTrg.PrdPol.xml**) are compared to the gold annotations of **teGld.xml** using the ABSA-16 evaluation measures/scripts.

**Subtask 2**

The baselines for Subtask 2 run after the ones for Subtask 1 and similar files are generated (e.g. **teCln.PrdAspTrgSB2.xml**).

1. **Evaluation scores for baselines**

For all domains we have set **sfl**=1, **xva**=0, **fld**=10, **partIdx**=9, **ftr=1000, thr=0.2** in the respective configuration files. Then, the scriptRunAll.sh script was executed. The script shuffles the reviews and splits the result into 10 parts. It uses the 10th part for testing and the remaining nine for training. The results are presented in **baselinesResults.ods**.

1. The n most frequent tokens of the training set are used. [↑](#footnote-ref-1)
2. We use the –b 1 option of LibSVM package to obtain probability estimates. [↑](#footnote-ref-2)
3. http://alt.qcri.org/semeval2016/task5/ [↑](#footnote-ref-3)
4. Java 1.8 is required. [↑](#footnote-ref-4)
5. http://www.csie.ntu.edu.tw/~cjlin/libsvm/ [↑](#footnote-ref-5)
6. Similar inputs and outputs are produced in cross validation mode, for example, for round 1 the script generates tr.xml.0, teGld.xml.0, teGldAspTrg.xml.0 etc. [↑](#footnote-ref-6)