TREC-IS 2019-A Results

Richard Mccreadie

Released June 2019. Updated October 2019

1 University of Paderborn

Submitted Runs

- **UPB_BERT**: Used transfer learning from BERT model to classify tweets as a multi-label classification problem.
- **UPB_FastText**: Used fasttext embeddings to represent tweets and classifiy as a multi-label classification problem.
- **UPB-BILSTM**: Used an BiLSTM deep model with BERT embeddings to classify tweets as a multi-label classification problem.
- **UPB_DICE**: Used an ensemble techniques which combines outputs of four different deep learning models. We used state-of-the-art architectures in deep learning (e.g Bi-LSTM) and transfer learning.

Organiser Notes: These runs were very conservative in terms of assigning high priority scores, resulting in low performance under the alerting metrics (there were lots of missed alerts). Information type categorization performance was one of the best submitted overall. Prioritization performance was around the median. Formatting note: some of these runs had strange formatting issues, with quotes were in the wrong places.

2 Beijing University of Technology

Submitted Runs

- **BJUTDMS-run1**: Uses pre-trained word vectors , generated a 50-dimensional word vector, and used the word vector as the input to xgboost, divided into 24 two-class models, and then integrated.
- **BJUTDMS-run2**: Uses pre-trained word vectors and augmented the training data set, generated a 50-dimensional word vector, and used the word vector as the input to xgboost, divided into 24 two-class models, and then integrated.
- **BJUTDMS-run3**: Uses pre-trained crisis word vectors , generated a 300-dimensional word vector, and used the word vector as the input to xgboost, divided into 24 two-class models, and then integrated.

Organiser Notes: These runs tended to return a very high number of tweets above the prioritization threshold that would generate an alert. This results in a higher than normal score under the alerting metrics, as we provide a flat 0.3 score for true positive alerts, even if the categories are incorrect. The issue with these runs are that there are too many false positive alerts returned, so the end-user would quickly loose trust in the system. So while the alerting scores are high here, actual usability would be limited. Information Type Categorization performance is very low for these runs. Prioritization error is better (lower) than most systems however.

3 Institut de Recherche en Informatique de Toulouse

Submitted Runs

- **IRIT_run_rf_gb**: Pre-processing (stopwords removing,...) Combination of rule-based, Gradient Boosting and Random Forest classifiers. 5 highest probabilities for dealing with multi-label classification.
- **IRIT_run_rf_gb_binary**: Pre-processing (stopwords removing,...), binary features Combination of rule-based, Gradient Boosting and Random Forest classifiers. Binary relevance for dealing with multi-label classification.
- **IRIT_run_rf_gb_binary_chain**: Pre-processing (stopwords removing,...), binary features Combination of rule-based, Gradient Boosting and Random Forest classifiers. Binary relevance and chain classifier for dealing with multi-label classification.
- **IRIT_run_rf_gb_threshold**: Pre-processing (stopwords removing,...) Combination of rule-based, Gradient Boosting and Random Forest classifiers. Threshold for dealing with multi-label classification.

Organiser Notes: These runs have a very low performance under the alerting metrics, as the priority scores produced are almost always below the threshold to generate an alert (0.7). Overall information type categorization performance is about median, but performance on the actionable categories is poor. Information prioritization overall is better than median however.

4 Indian Statistical Institute

Submitted Runs

- **irlabISIBase**: We follow Binary Relevance method for Multi-Label Learning with Logistic Regression Classifier with some feature enginnering techniques. We predict the priority of each tweet with its textual features along with the predicted Labels, used as a combined feature. We use tfidf vectorization of the tweets after preprocessing and cleaning.

- **irlabISIBase2**: In this run, we follow Binary Relevance method for Multi-Label Learning with Logistic Regression Classifier with some feature enginnering techniques. We predict the priority of each tweet with its textual features (tf-idf term weights), some additional text numerical features and tf-idf term weights of the POS tags. We use parameter tuned Logistic Regression.

- **irlabISIBase3**: In this run, we follow Binary Relevance method for Multi-Label Learning with Logistic Regression Classifier with some feature enginnering techniques. We predict the priority of each tweet with its textual features (tf-idf term weights), some additional text numerical features and tf-idf term weights of the POS tags. We use parameter tuned Logistic Regression. To make the model more generalized to the unseen samples we use spacy NER tagger for locations since events under test data might contain OOV words due to locations, organizations etc. Binary relevance and chain classifier for dealing with multi-label classification.

- **irlabISIDeep**: In this run, we build an LSTM Architecure to predict the classes (information types). We train the Network with Binary Cross Entropy Loss Function on each labels separately. We predict the priority of each tweet with its textual features (tf-idf term weights), some additional text numerical features and tf-idf term weights of the POS tags. We use parameter tuned Logistic Regression.

Organiser Notes: These runs have some of the better alerting scores, particularly irlabISIBase. That run also has by far the highest information type categorization performance for actionable information of the submitted runs (0.1969). Information prioritization scores are poorer, however that appears to be because the system only uses a sub-set of the score range (0.5-1, rather than 0-1). Indeed, irlabISIBase is likely the best 2019-A run if we consider performance alerting and information feed performance.

5 New York University

Submitted Runs

- **nyu-smap_run_baseline**: This method uses a tfi-idf-based vectorizer essentially equivalent to the umd_hcil-baseline version from 2018
- **nyu-smapp_run_baseline_multi**: This method uses a tfi-idf-based vectorizer with an one-vs-rest classifier to provide multiple class labels per tweet.
- **nyu-smapp_run_fasttext**: This method uses a fasttext-based embedding essentially equivalent to the umd_hcil-fasttext version from 2018.
- **nyu-smapp_run_fasttext_multi**: This method uses a fasttext-based embedding with an one-vs-rest classifier to provide multiple class labels per tweet.

Organiser Notes: Last-year's system (umd_hcil-baseline) is quite good relative to the other runs under the alerting metrics, both on actionable and on all tweets (best performance if we disregard the BJUT runs that returned over 80% of the dataset as alerts). However, it is not as good when focusing on information type categorization. The fasttext models seems to have a heavy bias towards categories with lots of training data. Format note: 2019-A runs use full category names, unlike 2018.

6 Carnegie Mellon University

Submitted Runs

- **cmu-rf**: Extract features from some hand-crafted features and some word embedding features (Glove, fasttext and so on), and use SkipThought as well as BERT to extract features. Use Random Forest as the model to do the classification on the features. Finally use top2 scores as prediction results.
- **cmu-rrf-autothre**: Extract features from some hand-crafted features and some word embedding features (Glove, fasttext and so on), and use SkipThought as well as BERT to extract features. Use Random Forest as the model to do the classification on the features. Finally use thresholds for different classes to generate prediction results.
- **cmu-rf-extra**: Extract features from some hand-crafted features and some word embedding features (Glove, fasttext and so on), and use SkipThought as well as BERT to extract features. Use Random Forest as the model to do the classification on the features. Finally use thresholds for different classes to generate prediction results.
- **cmu-xgboost**: Extract features from some hand-crafted features and some word embedding features (Glove, fasttext and so on), and use SkipThought as well as BERT to extract features. Use XGBoost as the model to do the classification on the features. Finally use thresholds (searched from 0.2 to 0.9) to generate prediction results.
- **cmu-xgboost-event**: Extract features from some hand-crafted features and some word embedding features (Glove, fasttext and so on), and use SkipThought as well as BERT to extract features. Use XGBoost as the model to do the classification on the features. Finally use event-wise model for different types of general events to generate prediction results.
- **cmu-xgboost-extra**: Extract features from some hand-crafted features and some word embedding features (Glove, fasttext and so on), and use SkipThought as well as BERT to extract features. Use XGBoost as the model to do the classification on the features. Finally use thresholds for different classes to generate prediction results.

Organiser Notes: Lower than median runs overall. The resultant learned models appear to only be effective on a limited number of information types, where Random Forest is better than XGBoost. Mixing some heuristics with the learned model might help for the actionable information types where the learned model is failing.

7 Mystery Group

Submitted Runs

- **SC-KRun28482low**: All my runs are using Keras Sequential model using different parameters, with features of using the text ngram, tweets users metadata, hashtags, user mentions.
- **SC-KRun68484low**: All my runs are using Keras Sequential model using different parameters, with features of using the text ngram, tweets users metadata, hashtags, user mentions.
- **SC-KRun2624435**: All my runs are using Keras Sequential model using different parameters, with features of using the text ngram, tweets users metadata, hashtags, user mentions.
- **SC-KRun-60002002410001**: All my runs are using Keras Sequential model using different parameters, with features of using the text ngram, tweets users metadata, hashtags, user mentions.

Organiser Notes: Lower than median runs overall for the learned model used in these runs. Looks like it does not find any information types for many tweets (in these cases it is probably better to include a fall-back type, e.g. Other-Irrelevant). The resultant learned model appears to only be effective on a small number of information types. It might be worth trying an SVM or decision tree classifier, mixed with some heuristics to bring performance up. Also, given how low the performance is for some classes, up-sampling may also help.

8 University at Buffalo

Submitted Runs

- **ubIS**: We implemented a basic linear SVM based classifier. There are 15 events files, each file with approximately 16 class labels. we fit a binary classifier on each of the class label i.e., the tweet with the current label under consideration is given '1' as label and the rest all are given '-1'. likewise, we fit for all the events and for each label thus having 15x16 approximate number of classifiers. During test case, for an event, if there are multiple models from our training that fall under the current test event, we estimate the prediction and basically take the ensemble prediction of the models. We use GADGET SVM - a distributed computing technique to run SVM in parallel.

Organiser Notes: Lower than median performance for the submitted run. The run does not return any tweets as alerts, as no importance scores exceed 0.7, hence the low performance on those metrics. Categorization performance is low, as the system appears to return the same categories again and again. Also too many categories are returned. Formating note: runs should be submitted in one file, not one file per event, and 2019-A runs use full category names.

9 University College Dublin

Submitted Runs

- **UCDrunEL1**: Feature matrix is constructed by a pre-trained word2vec model(2016, Muhammad) in domain (300 word2vec features). Emsemble technique combining Logistic Regression with Naive Bayes is trained on the 2018trecis-train and -test dataset. SMOTE is applied to leverage the imbalanced classes in training set. Priority is estimated by a linear combination of quantitative analysis and a priority classifier.

- **UCDrunEL2**: Feature matrix is constructed by a pre-trained word2vec model(2016, Muhammad) in domain (300 word2vec features). Emsemble technique: a tweet is assigned labels first classified by a Logistic Regression model combining the labels classified by an actionable-specific model afterwards. SMOTE is applied to leverage the imbalanced classes in training set. Priority is estimated by a linear combination of quantitative analysis and a priority classifier.

- **UCDrunELFB3**: Feature matrix is constructed by a pre-trained word2vec model(2016, Muhammad) in domain (300 word2vec features) and 21 handcrafted features for performance boosting in actionable types classification. Emsemble technique combining Logistic Regression with Naive Bayes is trained on the 2018trecis-train and -test dataset. SMOTE is applied to leverage the imbalanced classes in training set. Priority is estimated by a linear combination of quantitative analysis and a priority classifier.

- **UCDrunELFB4**: Feature matrix is constructed by a pre-trained word2vec model(2016, Muhammad) in domain (300 dimensions) and 21 hand-crafted features for performance boosting in actionable types classification. Emsemble technique: a tweet is assigned labels first classified by a Logistic Regression model combining the labels classified by an actionable-specific model afterwards. SMOTE is applied to leverage the imbalanced classes in training set. Priority is estimated by a linear combination of quantitative analysis and a priority classifier.

Organiser Notes: These runs are some of the better performing ones in terms of information type categorization, but are somewhat too conservative in terms of alerting. In contrast to a lot of the runs, the runs have a better distribution across classes (there are no complete class failures), likely due to the application of SMOTE. Evaluation note: For some reason UCDrunELFB3 breaks the evaluation script when calculating RMSE on the priority scores, the input format looks correct, so this may be an edge-case issue with scikitlearn.

10 Result Table

Run	Norm	Alerting		Information Feed			Prioritization	
		Accumulated Alert Worth		Info. Type Positive F1		Info. Type Accuracy	Priority RMSE	
		High Priority	All	Actionable	All	All	Actionable	All
UPB_BERT	N	-0.9680	-0.4882	0.0868	0.2312	0.8809	0.1514	0.0717
UPB_FastText	N	-0.9793	-0.4903	0.0922	0.1809	0.8501	0.1752	0.0737
UPB-BILSTM	N	-0.9643	-0.4855	0.0814	0.2027	0.8699	0.1744	0.0757
UPB_DICE	N	-0.9680	-0.4855	0.0501	0.1955	0.8740	0.1532	0.0664
BJUTDMS-run1	Y	-0.9942	-0.4971	0.0071	0.0958	0.6740	0.1343	0.0623
BJUTDMS-run2	Y	-0.9942	-0.4971	0.0237	0.1100	0.5565	0.1150	0.0563
BJUTDMS-run3	-	NA	NA	0.0014	0.0100	0.6642	NA	NA
ICTNET2019_IS	-	NA	NA	0.0046	0.0797	0.6844	NA	NA
IRIT_run_rf_gb	N	-0.9867	-0.4935	0.0185	0.1735	0.8052	0.1134	0.0557
IRIT_run_rf_gb_binary	Y	-0.6983	-0.3659	0.0202	0.1716	0.8834	0.0751	0.0694
IRIT_run_rf_gb_binary_chain	Y	-0.6983	-0.3659	0.0202	0.1716	0.8834	0.0751	0.0694
IRIT_run_rf_gb_threshold	N	-0.9867	-0.4935	0.0398	0.1774	0.7615	0.1134	0.0557
irlabISIBase	N	-0.2337	-0.1839	0.1695	0.2512	0.8521	0.1559	0.1552
irlabISIBase2	N	-0.4173	-0.2470	0.1284	0.2215	0.8812	0.1145	0.0823
irlabISIBase3	N	-0.3798	-0.2291	0.1487	0.2267	0.8775	0.1132	0.0833
irlabISIDeep	N	-0.3798	-0.2337	0.0856	0.1464	0.8765	0.1132	0.0833
nyu-smapp_run_baseline	N	-0.1213	-0.1973	0.0567	0.1326	0.8930	0.1223	0.1036
nyu-smapp_run_baseline_multi	N	-0.1213	-0.1973	0.0858	0.2087	0.8357	0.1223	0.1036
nyu-smapp_run_fasttext	N	-0.5409	-0.2957	0.0440	0.1210	0.8959	0.1406	0.0722
nyu-smapp_run_fasttext_multi	N	-0.2637	-0.2207	0.0320	0.1582	0.8982	0.1306	0.0911
cmu-rf	N	-0.9867	-0.4936	0.0409	0.1419	0.8734	0.1571	0.0743
cmu-rf-autothre	N	-0.8481	-0.4456	0.0442	0.1136	0.8911	0.1890	0.0942
cmu-rf-extra	N	-0.9942	-0.4975	0.0047	0.1489	0.8982	0.1886	0.0760
cmu-xgboost	N	-0.9942	-0.4972	0.0049	0.1639	0.8997	0.1864	0.0754
cmu-xgboost-event	N	-0.9942	-0.4971	0.0205	0.1302	0.8718	0.1751	0.0724
cmu-xgboost-extra	N	-0.9793	-0.4900	0.0106	0.1751	0.9017	0.1817	0.0732
SC-KRun28482low	Y	-0.9905	-0.4955	0.0000	0.0623	0.8962	0.1723	0.0756
SC-KRun68484low	Y	-0.9755	-0.4893	0.0000	0.0384	0.9003	0.1756	0.0747
SC-KRun2624435	Y	-0.9605	-0.4831	0.0000	0.0386	0.9039	0.1752	0.0743
SC-KRun-60002002410001	Y	-0.9568	-0.4827	0.0000	0.0483	0.8909	0.1731	0.0753
ubIS	-	NA	NA	0.0312	0.0979	0.3790	NA	NA
UCDrunEL1	N	-0.8744	-0.4442	0.0970	0.1627	0.7784	0.1149	0.0617
UCDrunEL2	N	-0.8556	-0.4382	0.0884	0.1467	0.8324	0.1144	0.0633
UCDrunELFB3	N	-0.9343	-0.4700	0.1180	0.1807	0.7853	0.1171	0.0603
UCDrunELFB4	N	-0.9268	-0.4676	0.0918	0.1617	0.8519	0.1207	0.0623

Table 1: 2019-A Submitted runs under the v2.3 evaluation script. Alerting metrics range from -1 to 1, higher is better. Information Feed metrics range from 0 to 1, higher is better. Prioritization metrics range from 0 to 1, lower is better. Norm indicates whether the run priority scores were subject to post-hoc max-min normalization by the organizers. Norm '-' indicates that the priority scores provided were not meaningful with or without normalization.