Predicting Emoji Exploiting Multimodal Data: FBK Participation in ITAmoji Task

Andrei Catalin Coman
Fondazione Bruno Kessler
coman@fbk.eu

Yaroslav Nechaev
Fondazione Bruno Kessler
nechaev@fbk.eu

Giacomo Zara
Fondazione Bruno Kessler
gzara@fbk.eu

Abstract

English. In this paper, we present our approach that has won the ITAmoji task of the 2018 edition of the EVALITA evaluation campaign[1]. ITAmoji is a classification task for predicting the most probable emoji (a total of 25 classes) to go along with the target tweet written by a given person in Italian. We demonstrate that using only textual features is insufficient to achieve reasonable performance levels on this task and propose a system that is able to benefit from the multimodal information contained in the training set, enabling significant F_1 gains and earning us the first place in the final ranking.

Italiano. *In questo articolo presentiamo l'approccio con cui abbiamo vinto la competizione ITAmoji dell'edizione 2018 di EVALITA[1]. ITAmoji è un task di classificazione per predire l'emoji più probabile (tra un totale di 25 classi) che possa essere associato ad un dato tweet scritto in italiano da uno specifico utente. Dimostriamo che utilizzare esclusivamente dati testuali non è sufficiente per ottenere un ragionevole livello di performance su questo task, e proponiamo un sistema in grado di beneficiare dalle informazioni multimodali contenute nel training set, aumentando significativamente lo score F_1 e guadagnando la prima posizione nella classifica finale.*

1 Introduction

Particularly over the last few years, with the increasing presence of social networks and instant messaging services in our lives, we have been witnessing how common it has become for average users to enrich natural language by means of emojis. An emoji is essentially a symbol placed directly into the text, which is meant to convey a simple concept or more specifically, as the name says, an emotion.

The emoji phenomenon has attracted considerable research interest. In particular, recent works have studied the connection between the natural language and the emojis used in a specific piece of text. The 2018 edition of EVALITA ITAmoji competition (Ronzano et al., 2018) is a prime example of such interest. In this competition, participants were asked to predict one of the 25 emojis to be used in a given Italian tweet based on a text, the date and the user that has written it. Differently from the similar SemEval (Barbieri et al., 2018) challenge, the addition of the user information significantly expanded the scope of potential solutions that could be devised.

In this paper, we describe our neural network-based system that exhibited the best performance among the submitted approaches in this task. Our approach is able to successfully exploit user information, such as the prior emoji usage history of a user, in conjunction with the textual features that are customary for this task. In our experiments, we have found that the usage of just the textual information from the tweet provides limited results: none of our text-based models were able to outperform a simple rule-based baseline based on prior emoji history of a target user. However, by considering all the modalities of the input data that were made available to us, we were able to improve our results significantly. Specifically, we combine into a single efficient neural network the typical Bi-LSTM-based recurrent architecture, that has shown excellent performance previously in this task, with the multilayer perceptron applied to user-based features.

[1] **EVALITA**: http://evalita.it/2018

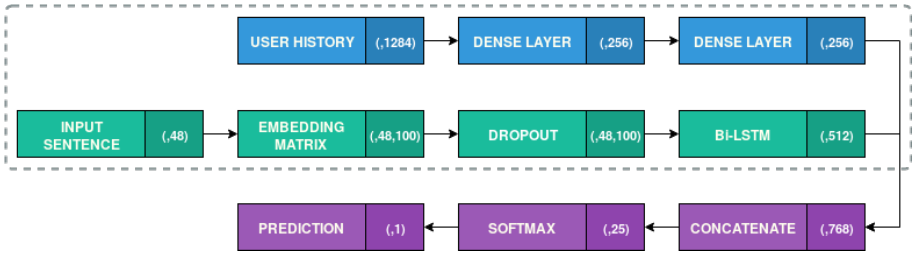

Figure 1: A diagram of the approach.

2 Description of the System

ITAmoji task is a classification task of predicting one of the 25 emojis to go along with the tweet. The training set provided by the organizers of the competition consists of 250 000 Italian tweets, including for each tweet the text (without the target emoji), the user ID and the timestamp as features. Participants were explicitly forbidden to expand the training set. Figure 1 provides an overview of our approach. In this section, we provide detailed descriptions of the methods we employed to solve the proposed task.

2.1 Textual features

In order to embed the textual content of the tweet, we have decided to apply a vectorization based on `fastText` (Bojanowski et al., 2017), a recent approach for learning unsupervised low-dimensional word representations. `fastText` sees the words as a collection of character n-grams, learning a representation for each n-gram. `fastText` follows the famous distributional semantics hypothesis utilized in other approaches, such as LSA, word2vec and GloVe. In this work, we exploit the Italian embeddings trained on text from Wikipedia and Common Crawl[2] and made available by the `fastText` authors[3]. Such embeddings include 300-dimensional vectors for each of 2M words in the vocabulary. Additionally, we have trained our own[4] embeddings using the corpus of 48M Italian tweets that were acquired from Twitter Streaming API. This yielded 1.1M 100-dimensional word vectors. Finally, we have also conducted experiments with word vectors suggested by the task organizers (Barbieri et al., 2016).

[2] `http://commoncrawl.org/`

[3] `https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md`

[4] `https://doi.org/10.5281/zenodo.1467220`

2.2 User-based features

Rather than relying solely on a text of the target tweet, we exploit additional user-based features to improve performance. The task features many variations of the smiling face and three different heart emojis, making it impossible even for a human to determine the most suitable one just based on a tweet. One of the features we considered was the prior emoji distribution for a target author. The hypothesis was that the choice of a particular emoji is driven mainly by the personal user preferences exemplified by the previous emoji choices.

To this end, we have collected two different types of emoji history for each user. Firstly, we use labels in the training set to compute emoji distributions for each user yielding vectors of size 25. Users from the test set that were not present in the training set were initialized with zeroes. Secondly, we have gathered the last 200 tweets for each user using Twitter API[5], and then extracted and counted all emojis that were present in those tweets. This yielded a sparse vector of size 1284. At this step we took extra care to prevent data leaks: if a tweet from the test set ended up in the collected 200 tweets, it wasn't considered in the user history. The runs that used the former, training set-based approach had a "`_tr`" suffix in its name. The ones that used the full user history-based approach had a "`_ud`" suffix.

In addition to prior emoji distribution, we did preliminary experiments with user's social graph. Social graph, which is a graph of connections between the users, is shown to be an important feature for many tasks on social media, for example, user profiling. We followed the recently proposed approach (Nechaev et al., 2018a) to acquire 300-dimensional dense user representations based on a social graph. This feature, however, did not improve the performance of our approach and was excluded.

[5] `https://developer.twitter.com`

Layer	Parameter	Value
Textual Input	*seq. length*	48
Embedding	*input*	256
	output	100
	trainable	true
	l2 regularization	10^{-6}
Dropout	*probability*	0.4
Bi-LSTM	*output*	512

(a) Bi-LSTM model hyperparameters.

History Input	*input*	1284
Dense	*output*	256
	l2 regularization	10^{-5}
	activation	tanh
Dense	*output*	256
	l2 regularization	10^{-5}
	activation	tanh

(b) User history model hyperparameters.

Concatenate	*output*	768
Dense	*output*	25
	l2 regularization	–
	activation	softmax
Optimizer	*method*	*Adam*
	learning rate	0.001
	β_1	0.9
	β_2	0.999
	decay	0.001

(c) Joint model and optimizer parameters.

Table 1: Model hyperparameters.

2.3 RNN exploiting textual features

The Recurrent Neural Networks have turned out to be a powerful architecture when it comes to analyzing and performing prediction on sequential data. In particular, over the last few years, different variations of the RNN has shown to be the top performing approaches for a wide variety of tasks, including tasks in Natural Language Processing (NLP). RNN consumes the input sequence one element at the time, modifying the internal state along the way to capture relevant information from the sequence. When used for NLP tasks, RNN is able to consider the entirety of the target sentence, capturing even the longest dependencies within the text. In our system, we use the bi-directional long short-term memory (Bi-LSTM) variation of the RNN. This variation uses two separate RNNs to traverse the input sequence in both directions (hence bi-directional) and employs LSTM cells.

Input text provided by the organizers is split into tokens using a modified version of the Keras tokenizer (can be found in our repository). Then, the input tokens are turned into word vectors of fixed dimensionality using the embedding matrix of one of the approaches listed in Section 2.1. The resulting sequence is padded with zeroes to a constant length, in our case 48, and fed into the neural network.

2.4 Overall implementation

In order to accommodate both textual and user-based features, we devise a joint architecture that takes both types of features as input and produces probability distribution for the target 25 classes. The general logic of our approach is shown in Figure 1. The core consists of two main components:

- **Bi-LSTM.** The recurrent unit consumes the input sequence one vector at a time, modifying the hidden state (i.e., memory). After the whole sequence is consumed in both directions, the internal states of the two RNNs are concatenated and used as a tweet embedding. Additionally, we perform $l2$-regularization of the input embedding matrix and the dropout to prevent overfitting and fine-tune the performance. Table 1a details the hyperparameters we used for a textual part of our approach.

- **User-based features.** The emoji distribution we collected (as described in Section 2.2) was fed as input to a multilayer perceptron: two densely-connected layers with $tanh$ as activation and $l2$-regularization to prevent overfitting. Table 1b showcases the chosen hyperparameters for this component using the full user history as input.

The outputs of the two components are then concatenated and a final layer with the $softmax$ activation is applied to acquire the probability distribution of the 25 emoji labels. The network is then optimized jointly with cross entropy as the objective function using `Adam` optimizer. Table 1c includes all relevant hyperparameters we used for this step.

Since the runs are evaluated based on macro-F_1, in order to optimize our approach for this metric, we have introduced class weights into the objective function. Each class i is associated with the weight equal to:

$$w_i = \left(\frac{\max_i(N)}{N_i} \right)^{\alpha} \quad (1)$$

where N_i is the amount of samples in a particular class and $\alpha = 1.1$ is a hyperparameter we tuned

for this task. This way the optimizer is assigning a greater penalty for mistakes in rare classes, thus optimising for the target metric.

During the training of our approach, we employ an early stopping criteria to halt the training once the performance on the validation set stops improving. In order to properly evaluate our system, we employ 10-fold cross-validation, additionally extracting a small validation set from the training set for that fold to perform the early stopping. For the final submission we use a simple ensemble mechanism, where predictions are acquired independently from each fold and then averaged out to produce the final submission. Additionally, one of the runs was submitted using predictions from the random fold. Runs exploiting the ensemble approach have the "_10f" suffix, while runs using just one fold have the "_1f" suffix.

The code used to preprocess data, train and evaluate our approach is available on GitHub[6].

3 Evaluation setting

In this section, we provide details on some of the approaches we have tested during the development of our system, as well as the models we submitted for the official evaluation. In this paper, we report results for the following models:

- MF_HISTORY. A rule-based baseline that always outputs the most frequent emoji from the user history based on a training set.
- BASE_CNN. A basic Convolutional Neural Network (CNN) taking word embeddings as input without any user-based features.
- BASE_LSTM. A Bi-LSTM model described in Section 2.3 used with textual features only.
- BASE_LSTM_TR. The complete approach including both feature families with emoji distribution coming from the training set.
- BASE_LSTM_UD. The complete approach with emoji distribution coming from the most recent 200 tweets for each user.

For the other models tested during our local evaluation and complete experimental results, please refer to our GitHub repository.

Additionally, for the BASE_LSTM approach we report performance variations due to a choice of a particular word embedding approach. In particular, provided refers to the ones that were suggested by organisers, custom-100d indicates our fastText-based embeddings and common-300d refers to the ones available on fastText website. Table 2 details the performances of the mentioned models.

Finally, we submitted three of our best models for the official evaluation. All of the submitted runs use the Bi-LSTM approach with our custom-100d word embeddings along with some variation of user emoji distribution as detailed in Section 2.2. Two of the runs use the ensembling trick using all available cross-validation folds, while the remaining one we submitted ("_1f") uses predictions from just one fold.

4 Results

Here we report performances of the models benchmarked both during our local evaluation (Table 2) and the official results (Table 3). We started experiments with just the textual models testing different architectures and embedding combinations. Among those, the Bi-LSTM architecture was a clear choice, providing 1-2% F_1 over CNN, which led to us abandoning the CNN-based models. Among the three word embedding models we evaluated, our custom-100d embedding exhibited the best performance on Bi-LSTM, while common-300d showed the best performance using the CNN architecture.

After we have acquired the user emoji distributions, we have devised a simple baseline (MF_HISTORY), which, to our surprise, outperformed all the text-based models we've tested so far: 3% F_1 improvement compared to the best Bi-LSTM model. When we introduced the user emoji histories in our approach, we have gained a significant performance gain: 4% when using the scarce training set data and 12% when using the complete user history of 1284 emojis from recent tweets. During the final days of the competition, we have tried to exploit other user-based features to further bolster our results, for example, the social graph of a user. Unfortunately, such experiments did not yield performance gains before the deadline.

During the official evaluation, complete user history-based runs exhibited top performance with ensembling trick actually decreasing the final F_1. As we expected from our experiments, training set-based emoji distribution was much less per-

[6] **GitHub repository**: https://github.com/Remper/emojinet

Approach	Embedding	Accuracy	Precision	Recall	F1 macro
MF_HISTORY	–	0.4396	0.4076	0.2774	0.3133
BASE_CNN	common-300d	0.4351	0.3489	0.2464	0.2673
BASE_LSTM	common-300d	0.4053	0.3167	0.2534	0.2707
BASE_LSTM	provided	0.4415	0.3836	0.2408	0.2622
BASE_LSTM	custom-100d	0.4443	0.3666	0.2586	0.2809
BASE_LSTM_TR	custom-100d	0.4874	0.4343	0.3218	0.3565
BASE_LSTM_UD	custom-100d	0.5498	0.4872	0.4097	**0.4397**

Table 2: Performance of the approaches as tested locally by us.

Run	Accuracy@5	Accuracy@10	Accuracy@15	Accuracy@20	F1 macro
BASE_UD_1F	0.8167	0.9214	0.9685	0.9909	**0.3653**
BASE_UD_10F	0.8152	0.9194	0.9681	0.9917	0.3563
BASE_TR_10F	0.7453	0.8750	0.9434	0.9800	0.2920
gw2017_p.list	0.6718	0.8148	0.8941	0.9299	0.2329

Table 3: Official evaluation results for our three submitted runs and the runner-up model.

formant but still offered significant improvement over the runner-up team (gw2017_p.list) as shown in Table 3. Additionally, we detail the performance of our best submission (BASE_UD_1F) for each individual emoji in Table 4 and Figure 2.

5 Discussion and Conclusions

Our findings suggest that emojis are currently used mostly based on user preferences: the more prior user history we added, the more significant performance boost we have observed. Therefore, the emojis in a text cannot be considered independently from the person that has used them and textual features alone can not yield a sufficiently performant approach for predicting emojis. Additionally, we have shown that the task was sensitive to the choice of a particular neural architecture as well as to the choice of the word embeddings used to represent text.

An analogous task was proposed to the participants of the SemEval 2018 competition. The winners of that edition applied an SVM-based approach for the classification (Çöltekin and Rama, 2018). Instead, we have opted for a neural network-based architecture that allowed us greater flexibility to experiment with various features coming from different modalities: the text of the tweet represented using word embeddings and the sparse user-based history. During our experiments with the SemEval 2018 task as part of the NL4AI workshop (Coman et al., 2018), we have found the CNN-based architecture to perform better, while here the RNN was a clear winner. Such discrepancy might suggest that even within the emoji prediction task the effectiveness of different approaches may significantly vary based either on a language of the tweets or based on a way the dataset was constructed.

In the future, we would like to investigate this topic further by trying to study differences in

	Precision	Recall	F_1	Support
❤️	**0.7991**	0.6490	**0.7163**	5069
😂	0.4765	**0.7116**	0.5708	4966
💋	0.6402	0.4337	0.5171	279
😊	0.5493	0.4315	0.4834	387
🌹	0.4937	0.4453	0.4683	265
🌟	0.7254	0.3229	0.4469	319
😍	0.3576	0.5370	0.4293	2363
😘	0.4236	0.4089	0.4161	834
💙	0.4090	0.3775	0.3926	506
😄	0.4034	0.3354	0.3663	1282
😀	0.4250	0.3299	0.3715	885
😉	0.3743	0.3184	0.3441	1338
😁	0.3684	0.3239	0.3447	1028
✨	0.3854	0.2782	0.3231	266
💯	0.3844	0.2711	0.3179	546
👍	0.3899	0.2648	0.3154	642
😎	0.3536	0.2743	0.3089	700
💪	0.3835	0.2566	0.3075	417
🤔	0.3525	0.1922	0.2488	541
💕	0.2866	0.2639	0.2748	341
😭	0.2280	0.2922	0.2562	373
🔝	0.2751	0.2133	0.2403	347
😅	0.2845	0.1741	0.2160	379
😜	0.3154	0.1822	0.2310	483
😱	0.2956	0.1824	0.2256	444

Table 4: Precision, Recall, F_1 of our best submission and the number of samples in test set for each emoji.

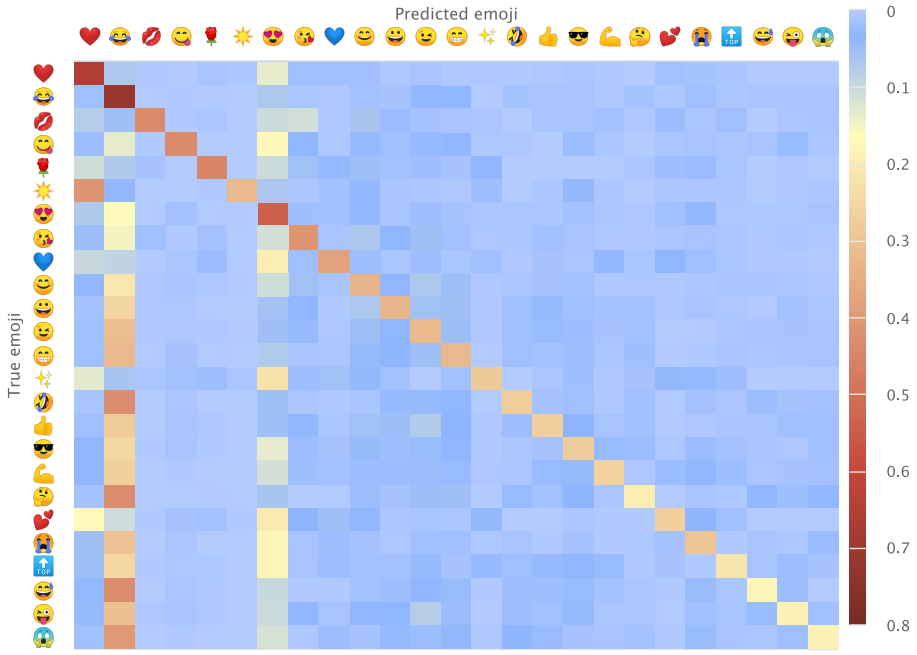

Figure 2: Confusion matrix for our best submission normalized by support size: each value in a row is divided by the row marginal. Diagonal values give recall for each individual class (see Table 4).

emoji usage between languages and communities. Additionally, we aim to further improve our approach by identifying more user-based features, for example, by taking into account the feature families suggested by Nechaev et al. (Nechaev et al., 2018b).

References

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proc. of the 12th Int. Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, United States. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Çagri Çöltekin and Taraka Rama. 2018. Tübingen-oslo at semeval-2018 task 2: Svms perform better than rnns in emoji prediction. In *Proc. of The 12th Int. Workshop on Semantic Evaluation, SemEval@NAACL-HLT, New Orleans, Louisiana*, pages 34–38.

Andrei Catalin Coman, Giacomo Zara, Yaroslav Nechaev, Gianni Barlacchi, and Alessandro Moschitti. 2018. Exploiting deep neural networks for tweet-based emoji prediction. In *Proc. of the 2nd Workshop on Natural Language for Artificial Intelligence co-located with 17th Int. Conf. of the Italian Association for Artificial Intelligence (AI*IA 2018), Trento, Italy*.

Yaroslav Nechaev, Francesco Corcoglioniti, and Claudio Giuliano. 2018a. Sociallink: Exploiting graph embeddings to link dbpedia entities to twitter profiles. *Progress in AI*, 7(4):251–272.

Yaroslav Nechaev, Francesco Corcoglioniti, and Claudio Giuliano. 2018b. Type prediction combining linked open data and social media. In *Proc. of the 27th ACM Int. Conf. on Information and Knowledge Management, CIKM 2018, Torino, Italy*, pages 1033–1042.

Francesco Ronzano, Francesco Barbieri, Endang Wahyu Pamungkas, Viviana Patti, and Francesca Chiusaroli. 2018. Overview of the evalita 2018 italian emoji prediction (itamoji) task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.