ZHAW-CAI at CheckThat! 2023: Ensembling using Kernel Averaging

Notebook for the CheckThat! Lab at CLEF 2023

Pius von Däniken¹, Jan Deriu¹ and Mark Cieliebak¹

¹Zurich University of Applied Sciences, Centre for Artificial Intelligence, Winterthur, Switzerland

Abstract
We describe our approaches to sub-task 1A on multi-modal check-worthiness classification of the CheckThat! Lab 2023 in English. The goal was to determine whether a tweet is worth fact-checking based on its text and image content. Our submission was based on a kernel ensemble of different uni-modal and multi-modal classifiers. It achieved second place out of 7 teams with an F1 score of 0.708.

Keywords
multi-modal, claim check-worthiness, multiple kernel learning, CheckThat!

1. Introduction

The CheckThat! Lab 2023 [1] included five tasks targeting various aspects of misinformation. We describe our approach to Task 1 Check-Worthiness in Multimodal and Unimodal Contents, which contained two sub-tasks. Of the two sub-tasks, we participated specifically in sub-task 1A targeting multi-modal content. The goal was to classify a tweet consisting of both text and an image as check-worthy or not. The sub-task was offered both in Arabic and English. We only developed methods for the English data.

Check-worthiness classification represents an important first triage step in a fact-checking pipeline. Successfully removing claims that are not worth checking reduces the work-load of human fact-checkers. It has been part of all CheckThat! Lab iterations so far [2, 3, 4, 5, 6]. Where previously the focus was on text content, this year’s sub-task 1A is a multi-modal task, involving both text and image data. This represents an important next step since much of the current content on social media is multi-modal in nature.

In this work, we will describe the approaches of our team, ZHAW-CAI, for sub-task 1A in English. We developed several different classifiers based on text only, ranging from traditional word frequency, to deep learning, and LLM solutions. We also developed a multi-modal classification model, as well as a kernel-method based ensemble model. When discussing our results, we will in particular highlight the importance of threshold selection.
2. Related Work

The general problem of misinformation in social media has received a lot of interest from the community in recent years. Apart from the CheckThat! Lab tasks there have been tasks focusing on identifying the veracity of a claim or rumour, such as RumourEval [7, 8] and FEVEROUS [9].

The first modern systems for check-worthiness detection include ClaimBuster [10] and ClaimRank [11]. Their main focus is on identifying check-worthy claims in political debates. The various CheckThat! Lab check-worthiness tasks have targeted different text genres, including social media and tweets in particular. While TF-IDF features are a staple of any text classification task and have been included in systems such as ClaimBuster, many successful previous participants [12, 13] used fine-tuned masked language models such as BERT [14] and RoBERTa [15] in their solutions. We include both approaches in our solution. In terms of analysis of multi-modal social media content, the Hateful Memes challenge [16] has sparked a lot of interest in the community. For the challenge of multi-modality for disinformation in particular, we refer the reader to a recent survey [17]. The MM-Claims dataset [18] is a recent multi-modal claim detection dataset, on which this shared task is based. Our multi-modal sub-component is most similar to systems such as [19] that use cross-attention between modalities. However, we use a full transformer [20] encoder to fuse the modalities. Of course, an important recent development involves the use of large language models such as the GPT family [21] and LLaMa [22] that exhibit astonishing zero-shot classification capabilities. We include this approach in our solutions as well. Finally, we use a multiple kernel learning [23] approach to combine these disparate classifiers into a unified ensemble model.

3. Method

3.1. Data

The multi-modal check-worthiness sub-task is a binary classification task where a tweet consisting of a short text and an image has to be classified as check-worthy or not. During the development phase of the shared task, the organizers released training data (\(D_{train}\)), validation data (\(D_{dev}\)) and a dev-test set to be used for evaluation during development (\(D_{dev-test}\)). The test data \(D_{test}\) was released shortly before the submission deadline and its labels were only released after the submission deadline. For all our experiments, we combine the \(D_{dev}\) and \(D_{dev-test}\) sets into a single validation set \(D_{valid}\). The individual systems are trained on \(D_{train}\) and evaluated on \(D_{test}\). The sizes of these sets and their label distributions are shown in Table 1. We note that each sample contained both text and image data. The training and development data came from the MM-Claims dataset [18] and for the full description of the task data, we refer the reader to the task overview [24].

3.2. Systems

We will now describe the different uni-modal and multi-modal systems we trained and our method to combine them using a kernel-based ensemble.
Table 1
Information for the English Data

<table>
<thead>
<tr>
<th></th>
<th>Number of Samples</th>
<th>Number of Check-worthy</th>
<th>Number non-check-worthy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{train}}$</td>
<td>2356</td>
<td>820 (34.8%)</td>
<td>1536 (65.2%)</td>
</tr>
<tr>
<td>$D_{\text{dev}}$</td>
<td>271</td>
<td>87 (32.1%)</td>
<td>184 (67.9%)</td>
</tr>
<tr>
<td>$D_{\text{dev-test}}$</td>
<td>548</td>
<td>174 (32.8%)</td>
<td>374 (68.2%)</td>
</tr>
<tr>
<td>$D_{\text{valid}}$</td>
<td>819 (= 271 + 548)</td>
<td>261 (31.9%)</td>
<td>558 (68.1%)</td>
</tr>
<tr>
<td>$D_{\text{test}}$</td>
<td>736</td>
<td>277 (37.6%)</td>
<td>459 (62.4%)</td>
</tr>
</tbody>
</table>

3.2.1. Text N-gram Classifier

Our first uni-modal system is based on the tweet text only. We first pre-process the texts by replacing URLs \(^1\), user handles, and sequences of emoji \(^2\) by placeholder tokens. The text was then lower-cased and tokenized by splitting on white-space. Tokens shorter than 2 characters were discarded. Based on this we computed TF-IDF \(^3\) vectors for each text. This means counting the uni-grams and bi-grams of tokens for each sample. We count only one occurrence for each n-gram, meaning we ignore repetitions. We also ignore n-grams that appear in fewer than 3 samples in $D_{\text{train}}$. Based on these counts one can compute the inverse document frequency (IDF) for each token. The resulting feature vectors are normalized to have unit euclidean length. We used the TfidfVectorizer implementation provided by scikit-learn \(^4\). We call the resulting feature vectors $x_{\text{text-ngram}}$.

We then use these feature vectors to train a linear Support Vector Machine (SVM) \(^5\) with regularization strength of 1. We again rely on the implementation provided by scikit-learn. In particular we also employ their implementation of reweighing the classes based on their frequency in the training data which was inspired by \(^6\). We will call this model text-ngram.

3.2.2. MLM Classifier

Next, we trained another text-only system. For this we fine-tuned an electra-base-discriminator \(^7\) model on the training data. Electra models have the same architecture as BERT \(^8\) but follow a different pre-training setup. During masked language modelling (MLM) pre-training there is both a generator network $G$ and a discriminator network $D$. During pre-training a certain number of input tokens are masked and $G$ has to predict the original token. The masked tokens are then replaced by those predicted by $G$ and $D$ has to determine whether a token was the original or has been replaced.

For our experiments we use the provided discriminator model checkpoint from Huggingface \(^9\). We show the training hyper-parameters in Table 2. We will call the resulting model electra-clf.

In section 3.2.5 we will need access to a feature vector extracted from electra-clf. For this we remove the final dense layer of electra-clf and use the model activations as feature vectors and

---

\(^1\)For this we use the urlextract package: https://github.com/lipoja/URLExtract.

\(^2\)For this we use the emoji package: https://github.com/carpedm20/emoji/.

\(^3\)https://huggingface.co/google/electra-base-discriminator
Table 2  
Training Hyper-parameters for electra-clf

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epochs</td>
<td>10</td>
</tr>
<tr>
<td>Batch Size</td>
<td>16</td>
</tr>
<tr>
<td>Optimizer</td>
<td>AdamW [31]</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>5e−5</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0.01</td>
</tr>
</tbody>
</table>

scale them to unit length. We will refer to these feature vectors as \(x_{\text{electra}}\).

3.2.3. Multi-Modal Classifier

Our multi-modal model relies on pre-trained encoder models for each modality. For text, we use the twitter-roberta-base\(^4\) checkpoint from Huggingface. This is a RoBERTa\(^15\) model that has been pre-trained on 58M tweets\([32]\). The output of this text encoder has dimensions \(L_{\text{text}} \times d_{\text{text}}\) where \(L_{\text{text}}\) is the number of tokens and \(d_{\text{text}}\) the dimension of the token embedding.

For images, we use a Vision Transformer (ViT)\([33]\) that has been pre-trained on ImageNet21k\([34]\). We again use a checkpoint provided by Huggingface\(^5\). The model takes images at a \(224 \times 224\) pixel resolution as input and processes them as a sequence of \(16 \times 16\) pixel patches. This results in an output representation of size \(L_{\text{img}} \times d_{\text{img}}\) where \(L_{\text{img}}\) is the number of patches and \(d_{\text{img}}\) the patch embedding dimension.

We first project both representations into a shared space of dimension \(d_{\text{shared}}\) using a dense layer and \texttt{relu} activation for each representation. This results in representations of sizes \(L_{\text{text}} \times d_{\text{shared}}\) and \(L_{\text{img}} \times d_{\text{shared}}\). We then concatenate them to get a new representation of size \(L \times d_{\text{shared}}\) where \(L = L_{\text{text}} + L_{\text{img}}\). We then feed this representation through a transformer encoder\([20]\) and a \texttt{relu} activation. The transformer encoder preserves the size of the representation and we use mean pooling across the sequence length to get an embedding \(x_{\text{multi-modal}}\) of size \(d_{\text{shared}}\). Finally, we normalize \(x_{\text{multi-modal}}\) to unit length and feed it through a final dense layer for classification.

We fine-tune this model on \(D_{\text{train}}\) but keep the weights of both the RoBERTa and the ViT encoders frozen. We call the resulting model \textit{multi-modal-clf} and show its hyper-parameters in Table 3.

3.2.4. LLM Classifier

Recent Large Language Models (LLMs) such as the GPT family\([21]\) have shown impressive few-shot and even zero-shot classification capabilities. In particular, chain-of-thought prompting\([35]\), where the model is asked to generate a step-by-step explanation how it arrives at a certain prediction, has shown much promise.

\(^4\)https://huggingface.co/cardiffnlp/twitter-roberta-base  
\(^5\)https://huggingface.co/google/vit-base-patch16-224-in21k
Table 3
Training Hyper-parameters for multi-modal-clf

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{ext}$</td>
<td>768</td>
</tr>
<tr>
<td>$d_{img}$</td>
<td>768</td>
</tr>
<tr>
<td>$d_{shared}$</td>
<td>256</td>
</tr>
<tr>
<td>Transformer Encoder Layers</td>
<td>1</td>
</tr>
<tr>
<td>Attention Heads</td>
<td>4</td>
</tr>
<tr>
<td>Transformer Feedforward Dimension</td>
<td>1024</td>
</tr>
<tr>
<td>Epochs</td>
<td>10</td>
</tr>
<tr>
<td>Batch Size</td>
<td>16</td>
</tr>
<tr>
<td>Optimizer</td>
<td>AdamW [31]</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>$5 \times 10^{-5}$</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Based on these observations, we constructed a very simple zero-shot classification prompt. We use the Language Model Query Language (LMQL) [36] to formulate the prompt and constrain the answers. We show the prompt written in LMQL in Listing 1.

Listing 1: LMQL Prompt

```
argmax
  Consider the following Tweet:
  {claim}
  Do you think this Tweet contains a claim that is worth fact-checking?
  Answer: [ANSWER]
  Reasoning: [REASON]
from openai/text-davinci-003
where
  STOPS_AT(Reason, ".")
  and ANSWER in [ 'Yes', 'No' ]
```

The placeholder `claim` is where we insert the tweet text. The placeholders `ANSWER` and `REASON` are filled in by the model. In our case we use OpenAI’s `text-davinci-003` model. The answer is constrained to the words `Yes` and `No` which we can directly use as predictions, which we will call `gpt-answer`. The reasoning is constrained to be one sentence, since it should stop generating when it produces the first full stop. We apply a similar feature extraction procedure as for `text-ngram` in Section 3.2.1 to these reasoning sentences. We forgo any special token replacements and use n-grams up to length 3 but keep the other parameters the same. The resulting feature vectors will be called $x_{gpt-ngram}$. We then train a linear SVM on $x_{gpt-ngram}$ and call it $gpt-ngram$.

[^6]: https://platform.openai.com/docs/models/gpt-3-5
3.2.5. Kernel Ensemble

We have seen that all our base models have an associated feature vector: \( x_{\text{text-ngram}} \), \( x_{\text{electra}} \), \( x_{\text{multi-modal}} \), and \( x_{\text{gpt-ngram}} \). For each of these we can define a linear kernel. The kernel value for two samples \( i \) and \( j \) for a given system \( s \) is then defined as \( k_s(i, j) = x_s^T(i)x_s(j) \), where \( x_s(i) \) is the feature vector of system \( s \) for sample \( i \). Given such a kernel \( k_s \), we can then train an SVM. For \( x_{\text{text-ngram}} \) and \( x_{\text{gpt-ngram}} \) this is equivalent to their associated classifiers \( \text{text-ngram} \) and \( \text{gpt-ngram} \). On the other hand, for \( x_{\text{electra}} \) and \( x_{\text{multi-modal}} \) we will call the resulting SVM classifiers \( \text{electra-kernel} \) and \( \text{multi-modal-kernel} \) respectively.

We will include an additional ViT encoder based feature vector \( x_{\text{img-untrained}} \). It is based on the same ViT encoder as \( \text{multi-modal-clf} \), which also provides a pooled representation for classification, which we will use as \( x_{\text{img-untrained}} \). We will call the resulting kernel-based SVM classifier \( \text{img-untrained-kernel} \).

Next, we show how we combine these kernels into an ensemble. Given a set of systems \( S \), we can define their average kernel as:

\[
k_{\text{avg}}(i, j) = \frac{1}{|S|} \sum_{s \in S} k_s(i, j)
\]

This is known as a fixed rule multiple kernel learning method [23]. We can then use \( k_{\text{avg}} \) to train an SVM. Our main submission was based on this method and used an average kernel using \( \text{text-ngram} \), \( \text{gpt-ngram} \), \( \text{electra-kernel} \), and \( \text{multi-modal-kernel} \) as components. We will also show results for \( \text{all-kernels} \) which additionally includes \( \text{img-untrained-kernel} \) in the average.\(^7\) All kernel-based SVMs were trained using a regularization strength of 1 and frequency based class weights.

4. Results

In Table 4 we show our main results. Our submission achieved an F1 score of 0.708 on the test set. We note that if we use the default classification threshold \(^8\) \( \text{electra-kernel} \) and \( \text{all-kernels} \) achieve that exact same score. This could indicate that our ensemble method is redundant. In practice, F1 scores can be sensitive to the decision threshold. In Figure 1 we show the Precision and Recall Curves for each system. They show the Precision and Recall of a system for all potential thresholds. In the plot we include lines of constant F1 in light gray. We can see that the default thresholds (black cross marks) tend to select sub-optimal operating points.

We could therefore try to find a better classification threshold. For this we can use the validation set \( D_{\text{valid}} \) and use the threshold which maximizes the F1 score on \( D_{\text{valid}} \). The results are shown as red cross marks in Figure 1 and in the column called Tuned Threshold in Table 4. Since \( \text{gpt-answer} \) provides only binary outputs we can not change its threshold. The values for \( \text{electra-clf} \) and \( \text{multi-modal-clf} \) are missing since we did not compute their output on \( D_{\text{valid}} \)\(^9\).

We can see that for most systems this method selects an even worse threshold. We had already

\(^7\)The difference between submission and \( \text{all-kernels} \) was due to time constraints.

\(^8\)For SVM-based systems the default threshold is 0, for classifiers trained using cross-entropy to produce class probabilities, the default threshold is 0.5.

\(^9\)This was due to time constraints.
Table 4
Performance of our Systems on $D_{test}$ based on different decision thresholds. Best values in each column in bold. See text for more details.

<table>
<thead>
<tr>
<th>System</th>
<th>Default Threshold</th>
<th>Tuned Threshold</th>
<th>Optimal Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td>gpt-answer</td>
<td>0.536</td>
<td>0.903</td>
<td>0.673</td>
</tr>
<tr>
<td>img-untrained-kernel</td>
<td>0.498</td>
<td>0.556</td>
<td>0.526</td>
</tr>
<tr>
<td>text-ngram</td>
<td>0.676</td>
<td>0.458</td>
<td>0.546</td>
</tr>
<tr>
<td>gpt-ngram</td>
<td>0.630</td>
<td>0.664</td>
<td>0.647</td>
</tr>
<tr>
<td>electra-kernel</td>
<td>0.768</td>
<td>0.657</td>
<td><strong>0.708</strong></td>
</tr>
<tr>
<td>multi-modal-kernel</td>
<td>0.812</td>
<td>0.545</td>
<td>0.652</td>
</tr>
<tr>
<td>submission</td>
<td>0.768</td>
<td>0.657</td>
<td><strong>0.708</strong></td>
</tr>
<tr>
<td>all-kernels</td>
<td>0.768</td>
<td>0.657</td>
<td><strong>0.708</strong></td>
</tr>
<tr>
<td>electra-clf</td>
<td>0.765</td>
<td>0.657</td>
<td>0.707</td>
</tr>
<tr>
<td>multi-modal-clf</td>
<td><strong>0.830</strong></td>
<td>0.527</td>
<td>0.645</td>
</tr>
</tbody>
</table>

Figure 1: Precision Recall Curves for all Systems computed on $D_{test}$

noticed this during development, where system performance varied greatly between $D_{dev}$ and $D_{dev-test}$, and therefore we chose the default classification threshold.

Finally, in Table 4 we also include the scores that could be achieved if we had access to the ideal threshold. We computed it by selecting the threshold which maximizes the F1 score on $D_{test}$. Of course, in reality one never has access to this knowledge, but we include it here to show how much influence threshold selection can have on the system comparison.

In Figure 1 we can also see that the curve for submission lies above the individual kernel based systems over the most recall values. Meaning that for most fixed recalls it achieves higher precision. This indicates that our ensembling method indeed yields an improved classifier. On the other hand, we can also see that electra-clf and multi-modal-clf perform even better.
In Figure 2 we show the Receiver Operating Characteristic (ROC) curves for all our systems. We can again see that electra-clf and multi-modal-clf have the highest area under the curve (AUC), meaning that for most fixed false positive rates they have a higher true positive rate than other systems. We can also see that our ensembling method outperforms individual kernel methods.

5. Conclusion

We have laid out our solution to the CheckThat! Lab 2023 sub-task 1A on multi-modal check-worthiness classification. Our solution includes diverse components that we combine using a multiple kernel learning approach. Our submission achieved second place out of 7 teams with an F1 score of 0.708. While analysing our results, we noted that the performance measure can vary drastically based on the selected decision threshold. When considering threshold-free methods such as ROC and PR curves, we find that our ensemble indeed seems to perform better than its individual components. Nevertheless, we note that the directly fine-tuned models outperform our submission under this lens. The performance gap between electra-clf and electra-kernel as well as multi-modal-clf and multi-modal-kernel is an open question requiring further study.

Acknowledgments

This work has been funded by the Hamison project supported by the EU ERA-Net CHIST-ERA; the Swiss National Science Foundation [20CH21_209672].
References


