Deep Learning Project

Elettra Favazza, Fabio Pernisi, Alan Picucci, Lorenzo Tarricone

May 2023

1 Introduction

In today's data-driven society, one of the major challenges lies in finding efficient and accurate ways to bridge the gap between various forms of data, such as visual and textual information. The project at hand revolves around this very challenge. Given a potentially ambiguous word in a limited textual context, we aim to develop a deep learning model capable of selecting the most suitable image, from a set of candidate images, that corresponds to the intended meaning of the target word. This task extends beyond basic image recognition or text analysis and requires harmonising the two. The significance of this problem is reflected in several real-world applications, including improving search engine results, enhancing the user experience on digital platforms, and aiding in teaching machines to comprehend intricate associations between visual and textual data.

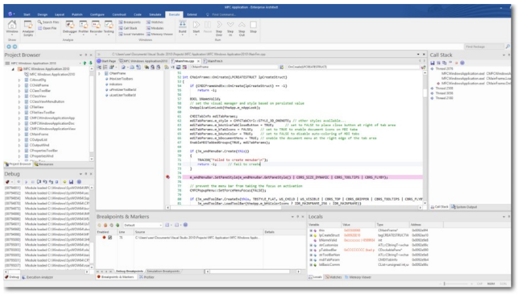
This project's task and related datasets are taken from Task 1 of the 17th international workshop about Semantic Evaluation (**SemEval-2023**) [1]

2 Problem Statement

Task 1 is centred around the disambiguation of target words in limited textual contexts, as well as their correlation with visual data. Specifically, we are given an ambiguous word and an additional word for context. The aim is to select among 10 candidate images the one representing the ambiguous word in the provided context.

2.1 Example

In this part, we want to give a flavour of the difficulty of the task that we require our system to perform.
For the ambiguous word "*bug*" and the full phrase "*programming bug*" the golden image is the one shown in figure 1.



```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CMDIFrameWndEx::OnCreate(lpCreateStruct) == -1)
        return -1;

    BOOL bNameValid;
    // set the visual manager and style based on persisted value
    OnApplicationLook(theApp.m_nAppLook);

    CMDITabInfo mdiTabParams;
    mdiTabParams.m_style = CMFCTabCtrl::STYLE_3D_ONENOTE; // other styles available...
    mdiTabParams.m_bActiveTabCloseButton = TRUE;      // set to FALSE to place close button at right of tab area
    mdiTabParams.m_bTabIcons = FALSE;    // set to TRUE to enable document icons on MDI taba
    mdiTabParams.m_bAutoColor = TRUE;    // set to FALSE to disable auto-coloring of MDI tabs
    mdiTabParams.m_bDocumentMenu = TRUE; // enable the document menu at the right edge of the tab area
    EnableMDITabbedGroups(TRUE, mdiTabParams);

    if (!m_wndMenuBar.Create(this))
    {
        TRACE0("Failed to create menubar\n");
        return -1;      // fail to create
    }

    m_wndMenuBar.SetPaneStyle(m_wndMenuBar.GetPaneStyle() | CBRS_SIZE_DYNAMIC | CBRS_TOOLTIPS | CBRS_FLYBY);

    // prevent the menu bar from taking the focus on activation
    CMFCPopupMenu::SetForceMenuFocus(FALSE);

    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYB
        !m_wndToolBar.LoadToolBar(theApp.m_bHiColorIcons ? IDR_MAINFRAME_256 : IDR_MAINFRAME))
```

Figure 1: Golden image for the target phrase "programming bug"

The other nine, instead, are shown in figure 2.
It's clear how the single instances were designed in order to trick any systems, adding both the other meaning of the ambiguous words (here the images of insects) and images related to the other word (in this case images related to the word of computers and informatics)

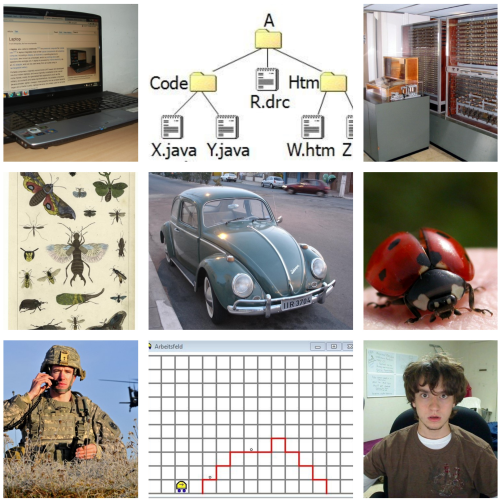


Figure 2: Wrong images for the target phrase ”programming bug”

3 Dataset Description

Our dataset is comprised of a folder for training and one for testing.
Each folder contains a collection of JPG images and two corresponding text files that detail the contextual data for each instance. The first text file, ’data.txt’, is tab-separated and contains the following information for each instance:

- a potentially ambiguous target word
- a full phrase providing limited context for the target word
- ten associated images labeled as ’image_1’ through ’image_10’. The images associated with each instance are represented by their filenames and can be located in the *image* folder.

The second text file, ’gold.txt’, contains the gold labels for each instance. These labels provide the correct image for the intended meaning of the target word in its specific context.

4 Methodology

The methodology of our project can be broken down into three main approaches: the Basic Zero-Shot Approach, the Baseline Approach and the CLIP Approach.

4.1 Basic zero-shot Approach

We tried to use ready-made Hugging Face multimodal models, such as ”Vision-and-Languge Transformer” (ViLT) [2] and ”Vision Text Dual Encoder” (nothing else than a double encoder that can project two separate encoding for vision and text in the same 512-dimensional vector space. We adopted it with ViT for the images and Uncased BERT for the text) [3].

4.2 Baseline Approach

Our first strategy, which we call *Baseline Approach*, is based on two pre-trained models: ResNet50 and DistilBert.

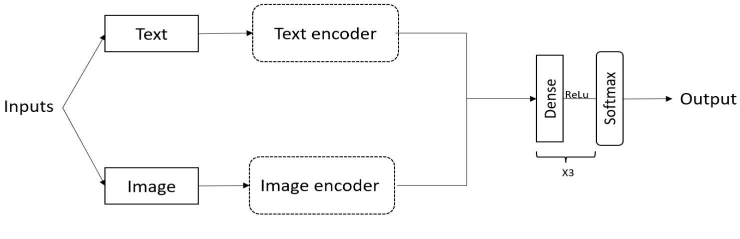
ResNet50 is employed to convert the input images into an embedded format, the image embeddings. The images in the dataset are first resized to 224×224; moreover, each color channel is normalized to the same scale used in the ImageNet dataset. This pre-processing ensures that input features are on a similar scale, providing effective lower-dimensional representations that preserve key visual information from the images.

Simultaneously, we use DistilBert to transform the textual inputs into a similar embedded format. DistilBert, a lighter version of BERT, seemed to provide a balanced and efficient solution to the trade-off between the results' quality and the computational cost of our model.

Upon obtaining these two types of embeddings, we concatenated them to create a unified representation capturing both visual and textual information. This combined representation is then fed into a Feed-Forward Neural Network for the classification task. The neural network comprised three Linear Layers, with the first two employing ReLU (Rectified Linear Unit) activation function to introduce non-linearity into the model.

We subsequently used an 80-20 split on the training dataset to evaluate our models before applying them to the test set.

We decided to adopt Adam as an optimizer for all the models cited in the ”Baseline” section. Moreover, we used a step scheduler for the learning rate with a stepsize of 2 and $\gamma = 0.8$



4.3 Adding regularisation to Baseline

Despite the seemingly sound methodology of the Baseline Approach, we noticed that our model was prone to overfitting (figure 4), indicating that it was capturing the noise in the training data. To alleviate this, we made several improvements. First, we incorporated regularization techniques, specifically Dropout (rate = 0.3) and Batch Normalization layers, into our Feed-Forward Neural Network. Dropout layers reduce overfitting by randomly dropping out (i.e., set to zero) a number (in our case 20% of the weights) of output features of the layer during training, while Batch Normalization helps in faster and more stable training. We chose to set the dropout parameter to 0.2 because of the shallowness of the network we implemented.

4.4 Finetuning ResNet and improving DistilBert in Baseline

A further improvement we implemented regards the fine-tuning of ResNet, together with the addition of a Dense layer applied to the output of DistilBert.

In particular, we unfroze the last Convolutional block and the last fully-connected layer in the ResNet50 model, updating the weights of these parts only. On the other hand, we added a dense layer with ReLU activation to the output of DistilBert, as proposed in [5]. Here, we maintained several regularization layers (dropout and batch normalization was implemented to the dense layer after Distilbert and in the final Network that performs classification).

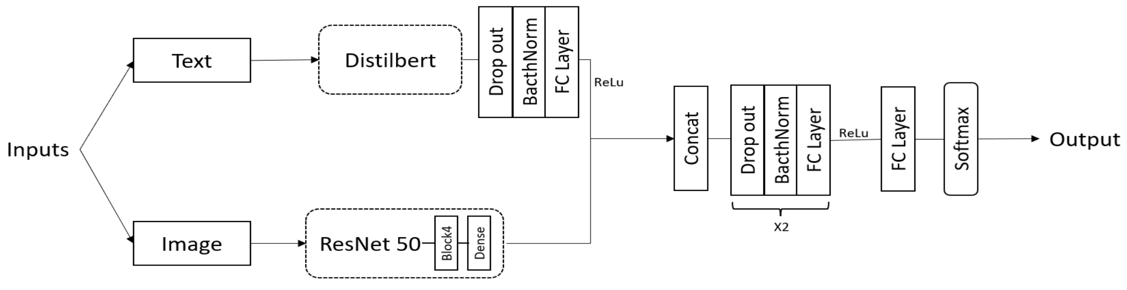


Figure 3:

4.5 CLIP Approach

Our second approach employed OpenAI's CLIP (Contrastive Language–Image Pretraining) model, which is designed to understand images and text in a joint embedding space. In particular, we employed clip-vit-base-patch32. Rather than treating images and text separately as in the Baseline Approach, CLIP has learned a unified representation that can compare images and text directly. It embeds them in the same space so that we can use a cosine similarity measure to understand how closely related some text input is to an image.

Initially, we applied CLIP directly to our data without any additional training (Vanilla CLIP), effectively treating it as a zero-shot learning task. CLIP was able to give us a much higher accuracy than the Baseline Approach.

Given that the full phrases in our dataset consisted only of the ambiguous word and another word which acted as context, we thought that obtaining additional textual context could further improve the model's performance. To this end, we used the Wikipedia API to expand our phrases, searching for articles related to the phrase or ambiguous word and returning a short summary. This is a common approach that we have found in the literature adopted by other groups taking part in this challenge, an example is the one of [4], which however searched for context in dictionaries. However, some target words or full phrases did not match any articles on Wikipedia. In those cases, we decided to query the GPT-3.5-turbo API to help us automatically define the ambiguous word or the sentence. The layout of the context-enhanced CLIP model can be seen if Figure 7

An example of full phrase expansion is the following: the full phrase "*bangalore torpedo*" becomes "*A Bangalore torpedo is an explosive charge placed within one or several connected tubes. It is used by combat engineers to clear obstacles that would otherwise require them to approach directly, possibly under fire*".

In conclusion, the methodologies we used in this project ranged from integrating pre-trained models for image and text processing to applying state-of-the-art models that understand both modalities in a joint manner. Through a series of experiments and improvements, we were able to achieve a decent model performance.

5 Experimental Results

The experimental results provided critical insights into the performance and potential of our methodologies. As the accuracy of the models increased, so did our understanding of how best to tackle the problem at hand.

5.1 Basic Zero-Shot Results

Concerning the pre-trained Hugging Face models, both approaches lead to a very bad result, with an accuracy of around 11% / 12%, just above the dummy random baseline of 10%. These results were obtained on just the first 100 samples and because of the simplicity of the model and its poor accuracy we decided not to investigate further, passing directly to more complex and promising models.

5.2 Baseline Approach Results

Our initial experimentation using the Baseline Approach yielded a modest accuracy of approximately 34% on the validation set. However, we noticed that our model demonstrated a significant level of overfitting, as observable in Figure 4.

5.3 Improved Baseline Approach Results

To combat overfitting, we introduced regularization techniques such as Dropout layers and Batch Normalization into our Feed-Forward Neural Network. As a result, the accuracy improved marginally to about 35.5% (figure 4). The dropout layers introduced randomness into our model by randomly dropping out a portion of neurons during training, preventing those neurons from co-adapting and making the model more robust. Meanwhile, Batch Normalization helped in standardizing the inputs to each layer, thus improving the overall stability and reducing running-time. These modifications did manage to mitigate overfitting to an extent, but not entirely, indicating the need for further optimization (figure 5).
The model that fine-tunes ResNet and adds a dense layer after DistilBert represents a significant improvement with respect to the regularized version of the base model, reaching an accuracy of 41.2% on the validation set (figure 6).

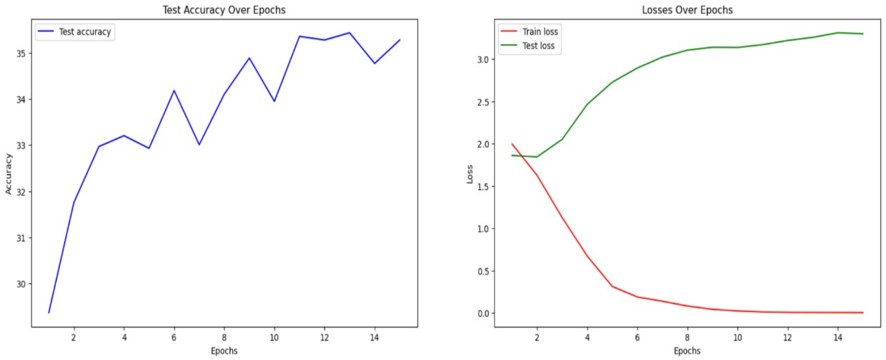


Figure 4: Baseline model presenting clear overfitting

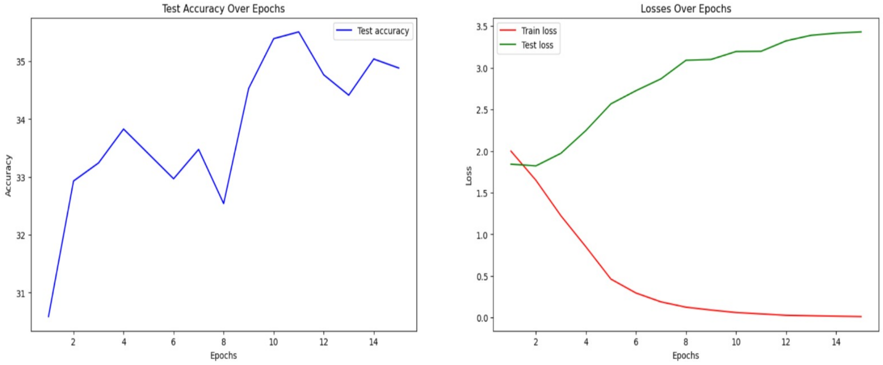


Figure 5: New regularized model

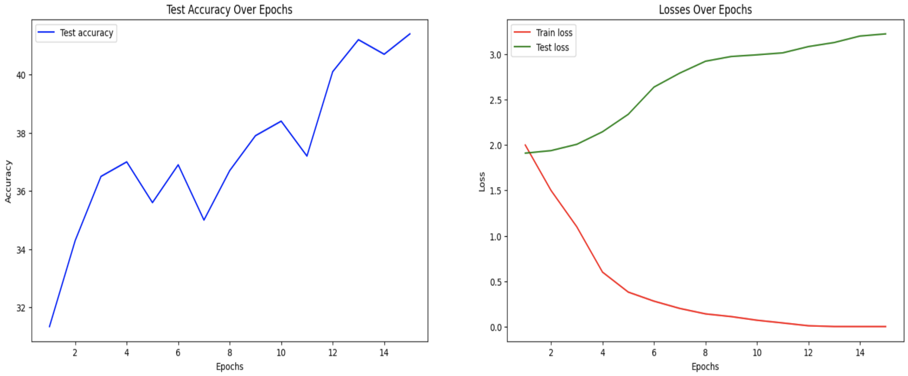


Figure 6: Accuracy and Losses of the finetuned-ResNet50 with a Dense Network applied to DistilBert's output

5.4 CLIP Approach Results

Now, we will present the results obtained using the CLIP model. The initial results were promising, with an accuracy jump to approximately 70% on the training set, suggesting that CLIP's joint understanding of image and text significantly contributed to better performance. The expanded context from Wikipedia and GPT-3.5 provided CLIP with a more detailed understanding of the intended meanings of the target words, further boosting our accuracy to an impressive 81% on the training set. However, it only resulted in an accuracy of 51% on the test set while, unexpectedly, Vanilla CLIP reached 55% accuracy.

We noticed that sometimes the presence of the extended context decreased the accuracy, especially on the test set provided by the challenge organizers (that we suspect was constructed to have the most difficult instances, as all our models showed a big decrease in performance). We suppose this is because sometimes the Wikipedia expansion is too rich, giving the model a way to "diverge its focus" from the ambiguous word itself.

The experimental results highlight the potential of applying state-of-the-art models like CLIP, which have been trained on a broad range of internet text and images, to complex problems such as ours. Further, they underline the importance of providing sufficient and meaningful context to our model, improving its ability to disambiguate and correlate information across different modalities.

6 Discussion and Conclusion

The experimentation with different methodologies offered intriguing insights. Our Baseline Approach, which combined pre-trained models for textual and visual embeddings, proved to not be very effective and highlighted the challenges of model overfitting. Even with regularisation techniques, we could not escape the model overfitting. We further experimented with regularization and noted that with an excessive level of regularization, the performance of our model decreased. The main bottleneck is surely the huge number of parameters (around 23.1

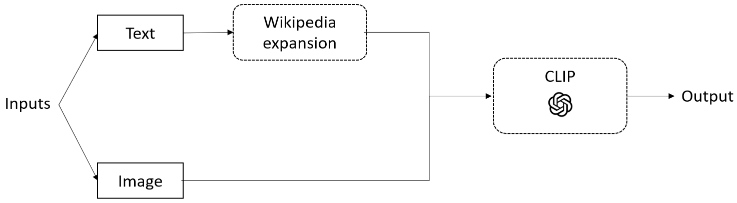


Figure 7: CLIP with context expansion

Accuracies

Model Name	Validation Set	Test set
ViLT	7%	N.A.
Base Model	35.5%	10.9%
Reg. + Finetune	41.2%	11.2%
Vanilla CLIP	71.9%	55%
Expanded CLIP	84%	49.4%

million in the first two baseline models and around 39.2 million for the final model which includes fine-tuning of ResNet) combined with the relatively small training set.

Our exploration of the CLIP Approach resulted in a significant jump in accuracy. We had seemingly achieved improvements by using the Wikipedia API to provide an expanded context to our target phrases, however, these did not seem to materialize on the test set, where Vanilla CLIP performed better. As mentioned earlier, this may be due to Wikipedia's context diverging too far from the ambiguous word itself. However, we should note that we noticed many more calls to GPT-3.5 while using the testing dataset, meaning that we relied on Wikipedia a lot less than when we evaluated the training set. Again, we imagine that this must be because the testing set contains much more obscure terminology and, probably, images too.

Our study's results have several important implications. They demonstrate the potential of models that learn joint representations of text and image data, suggesting promising directions for future research. The project also points to the importance of context in natural language understanding tasks, especially when dealing with ambiguous terms. Providing rich, relevant context could be a key factor in tackling more complex disambiguation problems in the future.

We now add a table that summarizes the results obtained with all our models:

References

[1] Semeval-2023 github repository. `https://semeval.github.io/SemEval2023/tasks`, 2023.

[2] Vision-and-language transformer hugging face website. `https://huggingface.co/docs/transformers/v4.18.0/en/model_doc/vilt`, 2023.

[3] Vision and text dual encoder hugging face website. `https://huggingface.co/docs/transformers/v4.18.0/en/model_doc/vilt`, 2023.

[4] Sławomir Dadas. Opi at semeval 2023 task 1: Image-text embeddings and multimodal information retrieval for visual word sense disambiguation. *arXiv preprint arXiv:2304.07127*, 2023.

[5] Miller et al. Multi-modal classification using images and text. *SMU Data Science Review: Vol. 3: No. 3, Article 6*, 2020.