Department of Computer, Control and Management Engineering

Master in Artificial Intelligence and Robotics

Report Homework 3

Elective in Artificial Intelligence - HoTNLP Module

Professor:
Roberto Navigli

Student:
Valerio Spagnoli
1887715

Academic Year 2023/2024

Contents

1 Introduction

1.1 Visual Word Sense Disambiguation

Given a word and some limited textual context, the Visual Word Sense Disambiguation task is to select among a set of candidate images the one which corresponds to the intended meaning of the target word.

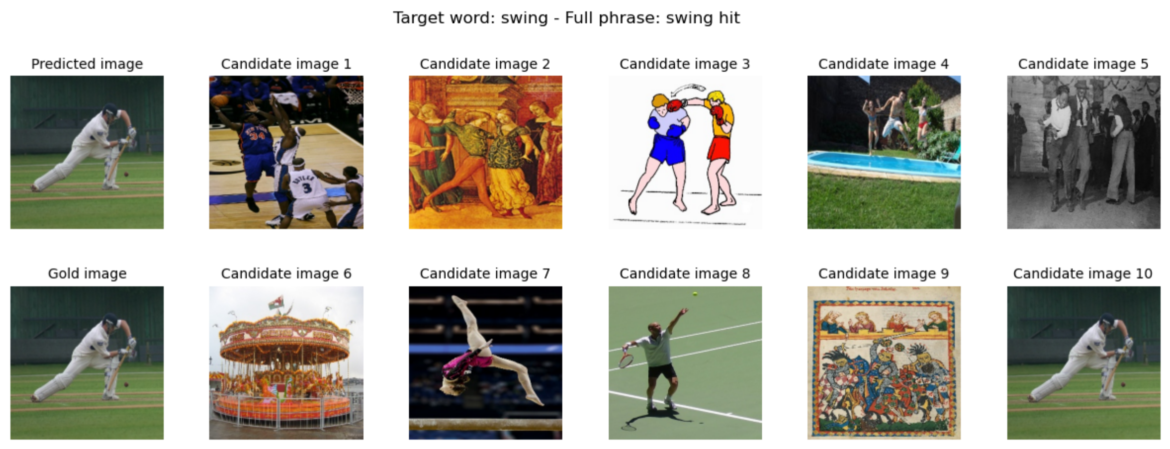

Figure 1: Example of Visual Word Sense Disambiguation

1.2 Dataset

To perform this task I used the datasets provided for SemEval-2023 Task-1[1] competition.

1.3 Proposed solutions

In this work two solutions have been proposed:

- fine-tune an existing model on the train set and then perform the inference on the resulting model;
- create a more complete description of the target image starting from the text provided during the inference step.

[1]https://raganato.github.io/vwsd/

2 Proposed solutions

The proposed solution uses CLIP[2], from OpenAI, to compute the embedding of the images and the texts and then compute the cosine similarity in order to figure out which candidate image is more similar to the given description.

2.1 Base solution

Given the `target_word`, the `full_phrase` and a list `candidate_images`, compute the `text_encoding` of the `full_phrase`, the `image_encoding` of each image in the list `candidate_images`, and then compute the cosine similarity between the `text_encoding` and each `image_encoding` in order to find the image with the highest similarity.

2.2 First solution: fine-tune CLIP

A train dataset with 12820 entries was provided for the Task 1 of the SemEval-2023 competition, where each entry was composed by the following fields (`target_word`, `full_phrase`, `image_1`, ..., `image_10`, `gold_image`). So, the first idea was fine-tune CLIP using this train dataset.

2.2.1 Training stage

The dataset of the training was composed by a set of pairs `text, image`, where `text` was the `full_phrase`, while `image` was the `gold_image`.

The loss was computed as the mean of the following losses:

- the MSE loss between the `logits_per_image` and the `ground_truth`;
- the MSE loss between the `logits_per_text` and the `ground_truth`.

The `ground_truth` is a tensor computed as `torch.arange(len(images))`, while the `logits_per_image` and `logits_per_text` are the outputs of the CLIP model.

2.2.2 Results

The results of this method weren't good.

2.3 Second solution: create a description of the target image

The second solution proposed for this task was to create a more complete description of the image, as opposed to the `full_phrase`, starting from the `full_phrase`. The reason for this idea is simple: trying to disambiguate the image is more difficult if the description is ambiguous, so using a more complete and less ambiguous description should facilitate the disambiguation process.

[2]https://openai.com/research/clip, https://github.com/openai/CLIP

2.3.1 Dataset

The test dataset provided for Task 1 of the SemEval-2023 competition was used for this solution, which contains 968 entries consisting of the following fields (`target_word`, `full phrase`, `image_1`, ..., `image_10`, `gold_image`). The dataset is divided in three languages: English (463 entries), Italian (304 entries), and Farsi (200 entries).

2.3.2 Main steps

The first two steps for this solution were:

1. **translation** of the texts from Italian and Farsi to English;
2. **create a description** for each image, starting from the `full_phrase`.

All these data was collected in a new dataset, which contains the data of the original test set, plus the translated texts and the descriptions.

The third step was the **inference step**, to predict the target images.

2.3.3 Translation

To translate the Italian and Farsi texts into English it was used `GoogleTranslate` from the `deep_translator`[3] library.

2.3.4 Create the description of the target image

The geneal idea for creating a description of the target image was to use WordNet[4] in order to find words that are releated with the words in the `full_phrase`.

I have implemented two solution for this idea:

- **sysnets-based solution**: define the description using the synsets of the words in `full_phrase`.
- **definition-based solution**: define the description using the definition of the words in `full_phrase`.

The words which compose the `full_phrase` were removed from the description (and, as consequence, also the `target_word` was removed).

Synsets-based solution This solution aims to generate a description from the synsets of each word in the `full_phrase`. The synsets are taken by WordNet using `wn.synset(token)`. Then, for each `synset` is taken the list of lemmas. Using the cosine similarity computed with CLIP the lemmas most similar to the words in the `full_phrase` are chosen. This lemmas will constitute the description.

[3]https://pypi.org/project/deep-translator/

[4]https://wordnet.princeton.edu/

The pseudocode to generate the description from the synsets is shown in the Algorithm 1.

Algorithm 1 Generate description from synsets

```
procedure SYNSET_BASED_DESCRIPTION(text, maximum_tokens)
    Tokenize text and save the tokens in tokenized_text.

    for token in tokenized_text do
        Find all synsets and all the lemmas of token and compute
        the associated similarity as the mean of the similarities between
        each lemma and all the words in tokenized_text.
        Save the pairs (lemma, similarity) in a list synonyms.
    end for

    Compute the mean of the similarities in synonyms.
    Remove from synonyms all the synonyms with a similarity
    score lower than the mean.

    Compute the description joining all the synonyms.
    Compute the description_weight as the mean of the
    similarities of the words in synonyns.
end procedure
```

Definition-based solution This solutions aims to generate a description from the definitions of each word in the `full_phrase`. To find all the definitions of each token in the `full_phrase`, the following steps were performed:

- tokenize the full_phrase;
- for each token take the entire list of synsets using `wn.synsets(token)`;
- for each synset `syn` take the definition of it using `syn.definition()`.

Using the cosine similarity the defintions most similar to the words in the `full_phrase` were selected. Then, from the remaining definitions the words that are more similar to the words in the `full_phrase` have been taken.

The pseudocode to generate the description from the definitions is shown in the Algorithm 2.

2.3.5 Predict images

Given the `full_phrase` and a list of candidate images, the prediction of the `target_image` is calculated by computing for each image in the list:

Algorithm 2 Generate description from definitions

```
procedure DEFINITION_BASED_DESCRIPTION(text, maximum_tokens)
    Tokenize text and save the tokens in tokenized_text.

    for token in tokenized_text do
        Find the definitions of the token using WordNet and
        compute the similarity score between each definition
        and the other tokens using CLIP.
    end for

    Compute the mean of the similarities of the definitions and
    take the definitions with a similarity score higher than the mean.

    Take the words in the definitions that are not in tokenized_text and
    save them in a list description.

    Compute the similairity between text and the words in description.

    Sort the words in description by similarity and
    take the maximum_tokens most similar.

    Compute the description by joining all the words in description.
    Compute the description_weight as the mean of the similarity
    of the words in description.
end procedure
```

- the cosine similarity between the `full_phrase` and the image (sim_1)
- the cosine similarity between the `description` and the image (sim_2)

The weighted similarity in computed as:

$$\texttt{similarity} = (1 - dw * 0.5) * sim_1 + dw * 0.5 * sim_2 \tag{1}$$

where dw = `description_weight`. In this way the `similarity` is:

$$\begin{cases} \texttt{similarity} = sim_1 & \text{if } \texttt{desacription\_weight} = 0 \\ \texttt{similarity} = 0.5 * (sim_1 + sim_2) & \text{if } \texttt{desacription\_weight} = 1 \\ \texttt{similarity} = (sim_1, 0.5 * (sim_1 + sim_2)) & \text{if } \texttt{desacription\_weight} = (0, 1) \end{cases} \tag{2}$$

2.3.6 Results

In the following tables the results of the description-based method are shown (Translation = True, Description = True), compared with the results that CLIP obtains using only the translation, only the description, or neither.

Synset-based solution The results of the synset-based solution are shown in the table 1. As we can see, the translation has greatly increased the accuracy for the test set in Italian and Farsi.

The synset-based description helps only in case the text is already in English. This is probably due to errors introduced during translation. So, in this case the most notable result is the difference in accuracy on the English test set between the third and fourth columns.

Translation Description	False False	False True	True False	True True
English accuracy	0.5745	**0.5918**	0.5745	**0.5918**
Italian accuracy	0.1836	0.2787	**0.5311**	**0.5311**
Farsi accuracy	0.0950	0.2300	**0.5150**	0.4550
Weighted Accuracy	0.3523	0.4184	**0.5486**	0.5444

Table 1: Synset-based solution results

Definition-based solution The results of the definition-based solution are shown in the table 2.

Using the definition-based solution we obtain the best results. The weighted accuracy in the case with translation and description is the highest, but even for this approach (as for the synset-based solution) a good result is obtained only for texts that are already in English. This is probably due to errors introduced during translation. Also in this case, the most notable result is the difference in accuracy on the English test between the third and fourt column.

Translation Description	False False	False True	True False	True True
English accuracy	0.5745	**0.6112**	0.5745	**0.6112**
Italian accuracy	0.1836	0.2754	**0.5311**	0.5115
Farsi accuracy	0.0950	0.2350	**0.5150**	0.5000
Weighted Accuracy	0.3523	0.4277	0.5486	**0.5568**

Table 2: Definition-based solution results