Multi-Lingual Natural Language Processing

Homeworks Report

SAPIENZA
UNIVERSITÀ DI ROMA

Dario Loi — 1940849

M.Sc. in AI & Robotics,
Sapienza, University of Rome.

A. Y. 2023–2024

Tasks

For the first homework, we were assigned two datasets:

1. WiC-ITA: Detect whether two italian words in two different sentences are used with the same meaning.
2. ITAmoji: Predict which emoji was used in a given italian tweet.

For each, we need to transform the samples into a format that is suitable for training an LLM, we also need to write the prompts that the model will use to make predictions. Additionally, we need to generate distractors for the ITAmoji dataset.

WiC-ITA

The first dataset's original samples' schema is shown in listing 1.

Listing: Sample 23 from WiC-ITA `train.jsonl`.

```
{
  "id": "lira.noun.15",
  "lemma": "lira",
  "sentence1": "In caso di inosservanza degli obblighi stabiliti dal comma 1 , si applica la
       sanzione amministrativa pecuniaria da lire dieci milioni a lire cento milioni .",
  "sentence2": "Per le finalità di cui all' Art. 1 , comma 2 , della legge regionale 26 aprile 1
       995 , n. 31 recante \" Norme in materia di musei degli Enti locali e di interesse locale
        \" è autorizzata per l' esercizio finanziario 2000 la spesa di lire 200.000.000 .",
  "start1": 115,
  "end1": 119,
  "start2": 230,
  "end2": 234,
  "label": 1
}
```

Desired Format

The desired schema is shown in listing 2.

Listing: Same sample with required changes.

```
{
  "id": "lira.noun.15",
  "lemma": "lira",
  "sentence1": "In caso di inosservanza degli obblighi stabiliti dal comma 1 , si applica la
       sanzione amministrativa pecuniaria da lire dieci milioni a lire cento milioni .",
    "sentence2": "Per le finalità di cui all' Art. 1 , comma 2 , della legge regionale 26 aprile
         1995 , n. 31 recante \" Norme in materia di musei degli Enti locali e di interesse
        locale \" è autorizzata per l' esercizio finanziario 2000 la spesa di lire 200.000.000
         .",
  "start1": 115,
  "end1": 119,
  "start2": 230,
  "end2": 234,
  "choices": [
    "DIVERSO",
    "UGUALE"
  ],
  "label": 1
}
```

ITAmoji

The second task's JSON schema is shown in listing 3.

Listing: Sample 27 from `ITAmoji_2018_TRAINdataset_v1.ANON.list`.

```
{
  "uid": "447352763",
  "text_no_emoji": "... il rumore del mare \ufe0f #28Settembre <URL>",
  "created_at": "Thu Sep 28 15:32:06 +0000 2017",
  "label": "red_heart",
  "tid": "913426094002458626"
}
```

For this task, we also had to add distractors, that is, plausible alternatives to the correct label.

Desired Format... Again

The desired output, with generated distractors, is shown in listing 4.

Listing: Sample 27 with generated distractors.

```
{
  "id": "ITA-emoji-train-00000027",
  "sentence": "... il rumore del mare \ufe0f #28Settembre <URL>",
  "choices": [
      "red_heart",
      "two_hearts",
      "rose",
      "kiss_mark"
  ],
  "label": 0
}
```

Distractor Clusters

To generate distractors, we used a hand-crafted list of semantic clusters, which we define as a set of emoji that are semantically related. For example, the cluster `love` contains the emojis `red_heart`, `two_hearts`, `blue_heart`, `rose`, and `kiss_mark`. When augmenting a sample, we select a cluster in which the correct label is present, and then randomly select three other labels from the same cluster. This way, we ensure that the distractors are capable to confuse the model.

Distractor Sampling

To obtain a set of distractors from an ITAmoji sample, we follow this process:

- Randomly sample a cluster from the list of clusters that contain the correct emoji.
- Randomly sample three other emojis from the same cluster.
- Return the new sample with the correct label and the three distractors.

Distractor Sampling

To obtain a set of distractors from an ITAmoji sample, we follow this process:

- Randomly sample a cluster from the list of clusters that contain the correct emoji.
- Randomly sample three other emojis from the same cluster.
- Return the new sample with the correct label and the three distractors.

Distractor Sampling

To obtain a set of distractors from an ITAmoji sample, we follow this process:

- Randomly sample a cluster from the list of clusters that contain the correct emoji.
- Randomly sample three other emojis from the same cluster.
- Return the new sample with the correct label and the three distractors.

Prompts

For both tasks, we adopted an *Chain-of-Thought* approach, where we show a couple of correct examples to the model before asking it to make predictions.
In the next slides we will show examples of this approach applied to both tasks.

WiC-ITA Prompts

Here is an example of a prompt for the WiC-ITA dataset:

```
Il tuo compito e’ effettuare delle classificazioni su un sample di un dataset Word in Context.
L’input consiste in due frasi e una parola,
l’output e’ una classificazione binaria che indica se il significato della parola
e’ diverso (0) o lo stesso (1) nelle due frasi.
Adesso elenchero una serie di esempi, per l’ultimo esempio dovrai completare la classificazione.
Fai attenzione a rispondere solo con 0 se la scelta e’ "DIVERSO"
o 0 se la scelta e’ "UGUALE", non scrivere altro.

Parola: RE
Frase 1: "Carlo magno fu incoronato re dei romani ad aquisgrana"
Frase 2: "La scala maggiore di re ha due alterazioni, fa e do"
Risposta corretta: 1

Parola: ARCO
Frase 1: "L’arco di trionfo e’ uno dei monumenti più famosi di parigi"
Frase 2: "Grazie all’invenzione dell’arco di volta,
i romani erano capaci di costruire monumenti impressionanti"
Risposta corretta: 0

Parola: {{lemma}}
Frase 1: {{sentence1}}
Frase 2: {{sentence2}}
Risposta corretta:
```

ITAmoji Prompts

Here is an example of a prompt for the ITAmoji dataset:

```
Il tuo compito e' quello di identificare l'emoji corretta
associata a un tweet prodotto da un utente italiano,
ti verranno fornite tre opzioni tra cui scegliere,
una delle quali e' corretta. Le opzioni sono simili tra di loro,
quindi presta attenzione ai dettagli.

Ecco alcuni esempi, completa l'ultimo rispondendo solo con l'emoji corretta.

Tweet: <MENTION_1> Avete visto cosa ha fatto il governo? <MENTION_2> #governo #italia
Scelte: ["thinking_face", "face_screaming_in_fear", "face_with_tears_of_joy"]
Risposta: thinking_face

Tweet: Ho dimenticato le chiavi in Macchina!!! Maledizione!!
Scelte: ["face_with_tears_of_joy", "face_screaming_in_fear", "loudly_crying_face"]
Risposta: face_screaming_in_fear

Tweet: {{sentence}}
Scelte: {{choices}}
Risposta:
```

Homework 1-B

The second homework required the training of some classifier models on the HaSpeeDe dataset, in order to detect whether a given sentence is hateful or non-hateful.
We were provided two datasets, one on news articles and one on tweets, both in the Italian language.

The Models

The models we trained were:

1. A näive classifier that always predicts the majority class.
2. A Logistic Regression model that operates on Word2Vec embeddings.
3. An LSTM model that automatically learns embeddings.

The Models

The models we trained were:

1. A näive classifier that always predicts the majority class.
2. A Logistic Regression model that operates on Word2Vec embeddings.
3. An LSTM model that automatically learns embeddings.

The Models

The models we trained were:

1. A näive classifier that always predicts the majority class.
2. A Logistic Regression model that operates on Word2Vec embeddings.
3. An LSTM model that automatically learns embeddings.

Näive Classifier

The näive classifier always predicted the non-hateful label, and achieved an accuracy of 0.638 on the news dataset and 0.508 on the tweets dataset.
We naturally achieved perfect precision, with a recall equal to that of the accuracy.

Word2Vec Regression

For the embeddings, we used the `gensim` library to train a Word2Vec model on the training data, with an embedding size of 300. We do not use pre-trained embeddings since the semantic meaning of some of the words differ from those of general language (we notice a general anti-establishment sentiment in the hateful tweets that we want to capture).
The Word2Vec embeddings of the sentence's tokens are averaged to obtain a single embedding that is then projected by a logistic regression model.

LSTM Model

Our LSTM model employs an embedding layer to automatically learn the embeddings of the words. We used a bidirectional LSTM with an embedding size of 300 and a hidden size of 128,
The model was trained for 5 epochs with a batch size of 32, embedding size was set to 300 (same as Word2Vec), and the hidden size was set to 128.

Results (News)

Figure: Results on the news test set

Model	Accuracy	Precision	Recall	F1
Logistic regression	0.666	0.776	0.824	0.800
LSTM	0.714	0.869	0.800	0.833
Baseline	0.638	1.000	0.638	0.779

Results (Tweets)

Figure: Results on the tweets test set

Model	Accuracy	Precision	Recall	F1
Logistic regression	0.605	0.650	0.898	0.754
LSTM	0.702	0.772	0.885	0.825
Baseline	0.508	1.000	0.508	0.673

Homework 2

The third homework focused on the Neural Logical Inference (NLI) task, performed on the FEVER dataset. The task consists of determining whether a given hypothesis is supported, refuted, or simply indeterminate given a premise that we assume to be true.

Data Augmentation

A key part of the homework was the generation of adversarial samples, these samples should have a more complex sentence structure than the original sample, in order to challenge the model's ability to generalize.
To assist in the algorithmic generation of this dataset, we were also provided with WSD (Word Sense Disambiguation) and SRL (Semantic Role Labeling) annotations for each sample.

Augmentation Pipeline

We show our augmentation pipeline in fig. 3. It allows for quick experimentation with different augmentation strategies.

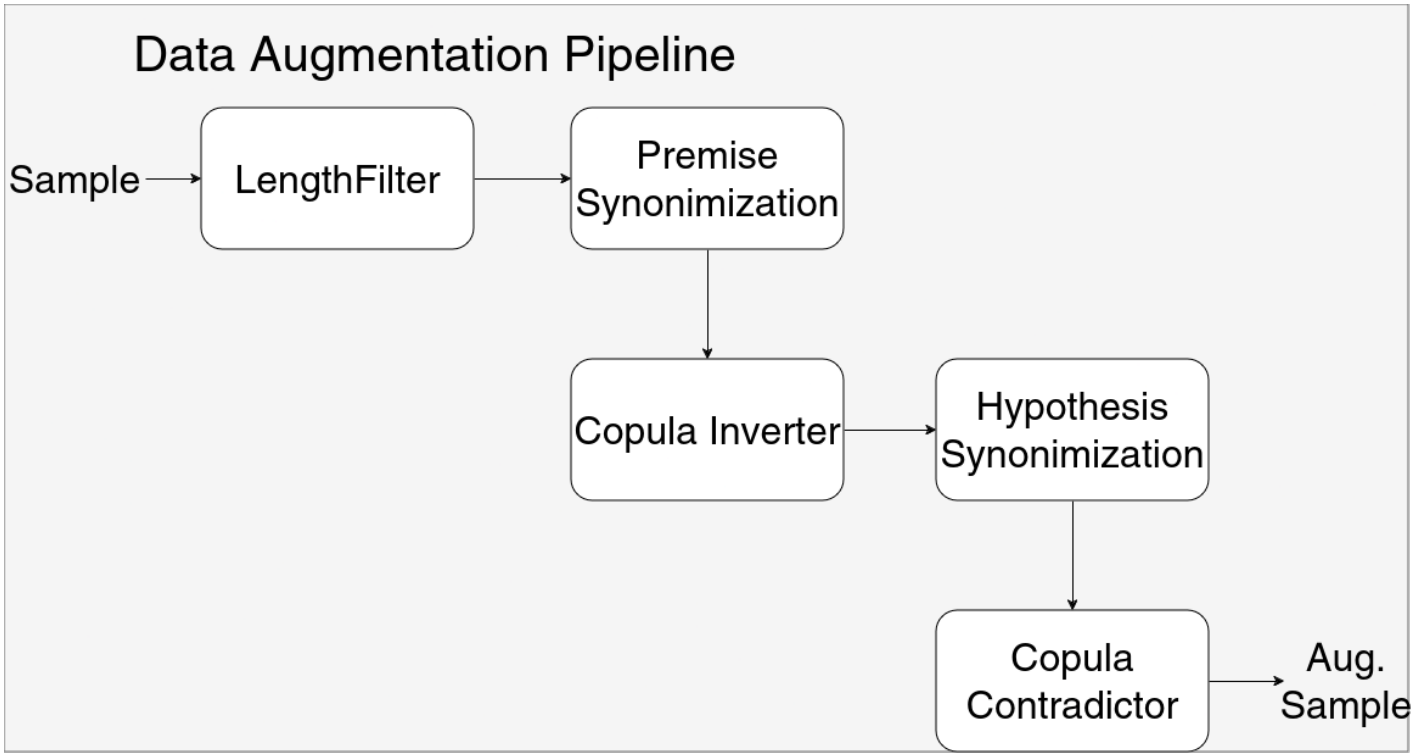

Figure: Augmentation pipeline.

Transformations

As shown in the figure, we implemented four transformations:

1. `Synonimization`: This transform uses WSD information to produce a set of sentences where some tokens are replaced with their synonyms.
2. `CopulaContradictor`: This transform replaces the copula in the sentence with its negation, while also negating the hypothesis (`ENTAILMENT` $\leftrightarrow$ `CONTRADICTION`).
3. `CopulaInverter`: This transform switches the subject and the object of the copula in the sentence, preserving the entailment relation.
4. `LengthFilter`: This transform discards samples that have different token lengths in the WSD and SRL annotations.

Transformations

As shown in the figure, we implemented four transformations:

1. `Synonimization`: This transform uses WSD information to produce a set of sentences where some tokens are replaced with their synonyms.
2. `CopulaContradictor`: This transform replaces the copula in the sentence with its negation, while also negating the hypothesis (`ENTAILMENT` $\leftrightarrow$ `CONTRADICTION`).
3. `CopulaInverter`: This transform switches the subject and the object of the copula in the sentence, preserving the entailment relation.
4. `LengthFilter`: This transform discards samples that have different token lengths in the WSD and SRL annotations.

Transformations

As shown in the figure, we implemented four transformations:

1. `Synonimization`: This transform uses WSD information to produce a set of sentences where some tokens are replaced with their synonyms.
2. `CopulaContradictor`: This transform replaces the copula in the sentence with its negation, while also negating the hypothesis (`ENTAILMENT` $\leftrightarrow$ `CONTRADICTION`).
3. `CopulaInverter`: This transform switches the subject and the object of the copula in the sentence, preserving the entailment relation.
4. `LengthFilter`: This transform discards samples that have different token lengths in the WSD and SRL annotations.

Transformations

As shown in the figure, we implemented four transformations:

1. `Synonimization`: This transform uses WSD information to produce a set of sentences where some tokens are replaced with their synonyms.
2. `CopulaContradictor`: This transform replaces the copula in the sentence with its negation, while also negating the hypothesis (`ENTAILMENT` $\leftrightarrow$ `CONTRADICTION`).
3. `CopulaInverter`: This transform switches the subject and the object of the copula in the sentence, preserving the entailment relation.
4. `LengthFilter`: This transform discards samples that have different token lengths in the WSD and SRL annotations.

Models

We trained two models (as asked by bonus task 2):

1. A baseline on the train set of FEVER.
2. An augmented model on a concatenation of the train and augmented sets.

Both were fine-tunings of the `distillRoBERTa-base` model. We chose RoBERTa as the focus of the task is on the production of a robust model, we opted for a distilled version due to our limited resources.

Test Sets

The final metrics are reported on two test sets, the first is the default FEVER test set, and the second is the adversarial test set that was provided to us. We assume that the augmented model will be more robust to the adversarial test set.

Results

We report our results in tables 1 and 2.

Table: Results on the base test set

Model	**Accuracy**	**Precision**	**Recall**	**F1**
Baseline	0.704	0.699	0.704	0.694
Augmented	0.683	0.681	0.683	0.672

Table: Results on the adversarial test set

Model	**Accuracy**	**Precision**	**Recall**	**F1**
Baseline	0.496	0.511	0.496	0.495
Augmented	0.519	0.542	0.519	0.522

Conclusions

It appears that our augmentation strategies improved the robustness of the model to adversarial data, however, we lost a bit of performance on the base test set. As our task is to improve the model's generality, we consider this a success.