famie

NLP Group at the University of Oregon

Jun 07, 2022

INTRODUCTION

FAMIE is a comprehensive and efficient **active learning** (AL) toolkit for **multilingual information extraction** (IE). FAMIE is designed to address a fundamental problem in existing AL frameworks where annotators need to wait for a long time between annotation batches due to the time-consuming nature of model training and data selection at each AL iteration. With a novel proxy AL mechanism and the integration of our SOTA multilingual toolkit Trankit, FAMIE can quickly provide users with a labeled dataset and a ready-to-use model for different IE tasks over 100 languages.

FAMIE's github: https://github.com/nlp-uoregon/famie

FAMIE's demo website: http://nlp.uoregon.edu:9000/

CHAPTER ONE

INSTALLATION

Installing *FaMIE* is easily done via one of the following methods:

1.1 Using pip

```
pip install famie
```

The command would install *FaMIE* and all dependent packages automatically.

1.2 From source

```
git clone https://github.com/nlp-uoregon/famie
cd famie
pip install -e .
```

This would first clone our github repo and install FaMIE.

If you encounter any other problem with the installation, please raise an issue here to let us know. Thanks.

CHAPTER

TWO

QUICK EXAMPLES

2.1 Initialization

To start an annotation session, please use the following command:

```
famie start
```

This will run a server on users' local machines (no data or models will leave users' local machines), users can access FAMIE's web interface via the URL: http://127.0.0.1:9000/

To start a new project, users need to upload an unlabeled dataset file with an entity type file (in text format) to the web interface. After that, they will be directed to a data statistic page. Clicking on the bottom left corner will start the labeling process.

2.2 Annotation

for each data sample, annotators first select a label from dropdown, then proceed to highlight appropriate spans for the corresponding labels.

Annotators continue labeling until all entities in the given sentence are covered, from which they can proceed by clicking save button and then next arrow to go to the next example.

After finishing labeled every unlabeled data of the current iteration, clicking on **Finish Iter** will take users to a waiting page for the next iteration (during this time, the proxy model is being retrained with the new labeled data, which usually takes about 3 to 5 minutes).

2.3 Output

FAMIE allows users to download the trained models and annotated data of the current round via the web interface.

FAMIE also provides a simple and intuitive code interface for interacting with the resulting labeled dataset and trained main models after the AL processes.

```python
import famie

# access a project via its name
p = famie.get_project('named-entity-recognition')

# access the project's labeled data
```

(continues on next page)

(continued from previous page)

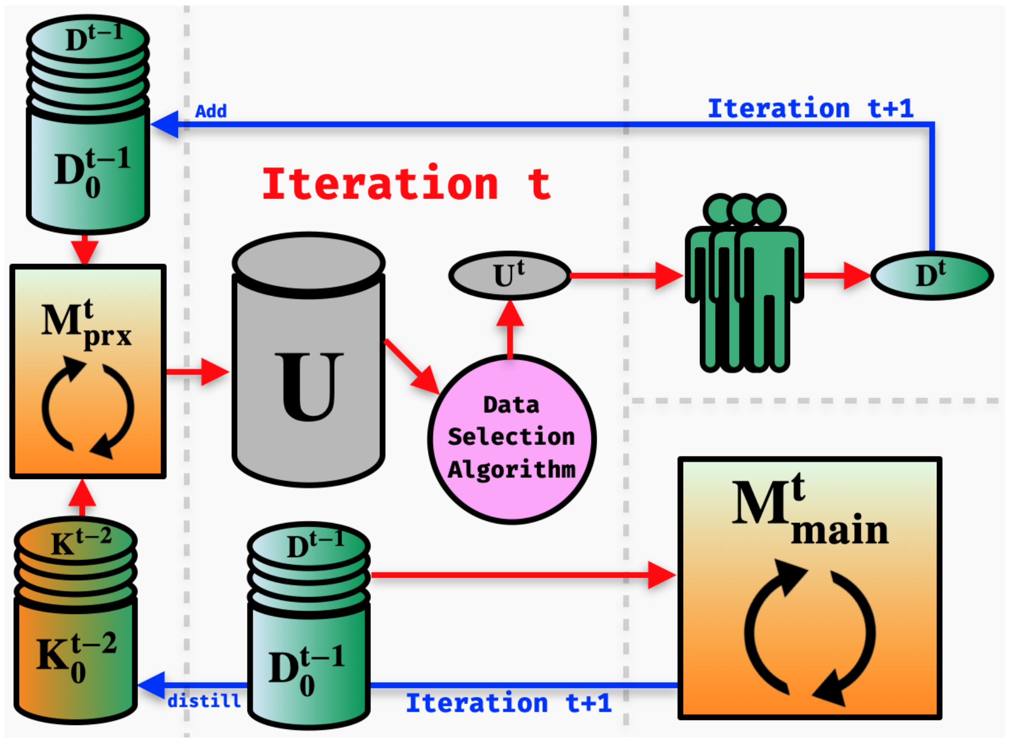
```
data = p.get_labeled_data() # a Python dictionary

# export the project's labeled data to a file
p.export_labeled_data('data.json')

# export the project's trained model to a file
p.export_trained_model('model.ckpt')

# access the project's trained model
model = p.get_trained_model()

# access a trained model from file
model = famie.load_model_from_file('model.ckpt')

# use the trained model to make predictions
model.predict('Oregon is a beautiful state!')
# ['B-Location', 'O', 'O', 'O', 'O']
```

CHAPTER

THREE

HOW FAMIE WORKS

In this section, we briefly present the most important details of the technologies used by FaMIE.



Incorporating current large-scale language models into traditional AL process would dramatically increase the model training time, thus introducing a long idle time for annotators that might reduce annotation quality and quantity. To address this issue without sacrificing final performance, FAMIE introduces **Proxy Active Learning**. In particular, a small proxy model is used to unlabeled data selection, while the main model is trained during the long annotation time of the annotators (i.e., main model training and data annotation are done in parallel). Given the main model trained at previous iteration, knowledge distillation will be employed to synchronize the knowledge between the main and proxy models at the current iteration.