From Fundamentals to Recent Advances A Tutorial on Keyphrasification

dlkp - A Deep Learning Library for Keyphrase Extraction and Generation

Rui Meng, Debanjan Mahata, Florian Boudin

ECIR 2022

salesforce research

MOODY'S ANALYTICS

LS2N LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Why dlkp

- No deep learning library for latest keyphrase extraction and generation methods.
- No programmatic access to prepared benchmark datasets
- No standard pipeline for evaluating deep learning based extraction and generation methods
- No library for exploring modern contextual language models
  - Encoder models
  - Generative models
- No one stop for - Training, Evaluating and Benchmarking keyphrase extraction and generation methods.

What is dlkp

- A deep learning library for identifying keyphrases
- Keyphrase extraction using sequence tagging
- Keyphrase generation using seq2seq
- Benchmark datasets
- Evaluation metrics
- A one stop for training and evaluating latest keyphrase extraction and generation methods

midas-research / **dlkp**

https://github.com/midas-research/dlkp

Training and Evaluating Keyphrase Extraction Models

- **Step 1** - Import the required modules

```
from dlkp.models import KeyphraseTagger
from dlkp.extraction import (
    KEDataArguments,
    KEModelArguments,
    KETrainingArguments,
)
```

- **Step 2** - Initialize the data arguments.

```
data_args = KEDataArguments(
    dataset_name="midas/inspec",
    dataset_config_name="extraction",
    pad_to_max_length=True,
    overwrite_cache=True,
    label_all_tokens=False,
    preprocessing_num_workers=8,
    return_entity_level_metrics=True,
)
```

https://huggingface.co/datasets/midas/inspec

Benchmark Datasets

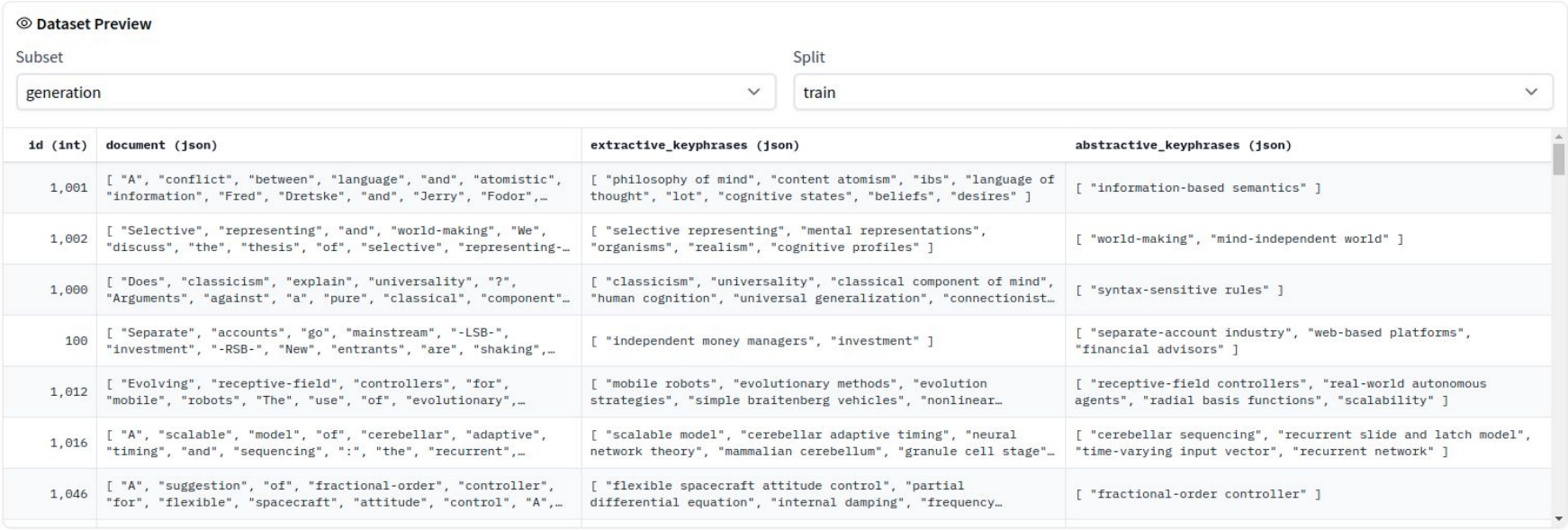
Dataset Preview — Go to dataset viewer

Subset: extraction | Split: train

id (int)	document (json)	doc_bio_tags (json)
1,001	["A", "conflict", "between", "language", "and", "atomistic", "information", "Fred",…	["O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O",…
1,002	["Selective", "representing", "and", "world-making", "We", "discuss", "the", "thesis",…	["B", "I", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O",…
1,000	["Does", "classicism", "explain", "universality", "?", "Arguments", "against",…	["O", "B", "O", "B", "O", "O", "O", "O", "O", "B", "I", "I", "I", "O", "O", "O", "O",…
100	["Separate", "accounts", "go", "mainstream", "-LSB-", "investment", "-RSB-", "New",…	["O", "O", "O", "O", "O", "B", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O",…
1,012	["Evolving", "receptive-field", "controllers", "for", "mobile", "robots",…	["O", "O", "O", "O", "B", "I", "O", "O", "O", "B", "I", "O", "O", "O", "O", "O", "O",…
1,016	["A", "scalable", "model", "of", "cerebellar", "adaptive", "timing", "and",…	["O", "B", "I", "O", "B", "I", "I", "O", "O", "O", "O", "O", "O", "O", "O", "O", "O",…
1,046	["A", "suggestion", "of", "fractional-order", "controller", "for", "flexible", "spacecraft"…	["O", "O", "O", "O", "O", "O", "B", "I", "I", "I", "O", "O", "O", "O", "O", "B", "I",…

https://huggingface.co/datasets/midas/inspec

Benchmark Datasets



Dataset Preview

Subset: generation

Split: train

id (int)	document (json)	extractive_keyphrases (json)	abstractive_keyphrases (json)
1,001	["A", "conflict", "between", "language", "and", "atomistic", "information", "Fred", "Dretske", "and", "Jerry", "Fodor",…	["philosophy of mind", "content atomism", "ibs", "language of thought", "lot", "cognitive states", "beliefs", "desires"]	["information-based semantics"]
1,002	["Selective", "representing", "and", "world-making", "We", "discuss", "the", "thesis", "of", "selective", "representing-…	["selective representing", "mental representations", "organisms", "realism", "cognitive profiles"]	["world-making", "mind-independent world"]
1,000	["Does", "classicism", "explain", "universality", "?", "Arguments", "against", "a", "pure", "classical", "component"…	["classicism", "universality", "classical component of mind", "human cognition", "universal generalization", "connectionist…	["syntax-sensitive rules"]
100	["Separate", "accounts", "go", "mainstream", "-LSB-", "investment", "-RSB-", "New", "entrants", "are", "shaking",…	["independent money managers", "investment"]	["separate-account industry", "web-based platforms", "financial advisors"]
1,012	["Evolving", "receptive-field", "controllers", "for", "mobile", "robots", "The", "use", "of", "evolutionary",…	["mobile robots", "evolutionary methods", "evolution strategies", "simple braitenberg vehicles", "nonlinear…	["receptive-field controllers", "real-world autonomous agents", "radial basis functions", "scalability"]
1,016	["A", "scalable", "model", "of", "cerebellar", "adaptive", "timing", "and", "sequencing", ":", "the", "recurrent",…	["scalable model", "cerebellar adaptive timing", "neural network theory", "mammalian cerebellum", "granule cell stage"…	["cerebellar sequencing", "recurrent slide and latch model", "time-varying input vector", "recurrent network"]
1,046	["A", "suggestion", "of", "fractional-order", "controller", "for", "flexible", "spacecraft", "attitude", "control", "A",…	["flexible spacecraft attitude control", "partial differential equation", "internal damping", "frequency…	["fractional-order controller"]

https://huggingface.co/datasets/midas/inspec

Training and Evaluating Keyphrase Extraction Models

- **Step 3** - Initialize the training arguments.

```
training_args = KETrainingArguments(
    output_dir="{path_to_your_directory}",
    learning_rate=4e-5,
    overwrite_output_dir=True,
    num_train_epochs=50,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=4,
    do_train=True,
    do_eval=True,
    do_predict=False,
    evaluation_strategy="steps",
    save_steps=1000,
    eval_steps=1000,
    logging_steps=1000
)
```

- **Step 4** - Initialize the model arguments.

```
model_args = KEModelArguments(
    model_name_or_path="bloomberg/KBIR",
    use_crf=True,
)
```

- **Step 5** - Train and evaluate the model.

```
KeyphraseTagger.train_and_eval(
    model_args=model_args,
    data_args=data_args,
    training_args=training_args,
)
```

- **Step 6** - Visualize your training progress using tensorboard.

```
tensorboard --logdir {path_to_your_log_dir}
```

Extracting Keyphrases

- **Step 7** - Load the trained model for prediction

```
tagger = KeyphraseTagger.load(
    model_name_or_path="{path_to_your_directory_where_model_is_saved}"
    )

input_text = "In this work, we explore how to learn task-specific language models aimed towards learning rich " \
             "representation of keyphrases from text documents. We experiment with different masking strategies for " \
             "pre-training transformer language models (LMs) in discriminative as well as generative settings. In the " \
             "discriminative setting, we introduce a new pre-training objective - Keyphrase Boundary Infilling with " \
             "Replacement (KBIR), showing large gains in performance (upto 9.26 points in F1) over SOTA, when LM " \
             "pre-trained using KBIR is fine-tuned for the task of keyphrase extraction. In the generative setting, we " \
             "introduce a new pre-training setup for BART - KeyBART, that reproduces the keyphrases related to the " \
             "input text in the CatSeq format, instead of the denoised original input. This also led to gains in " \
             "performance (upto 4.33 points in F1@M) over SOTA for keyphrase generation. Additionally, we also " \
             "fine-tune the pre-trained language models on named entity recognition (NER), question answering (QA), " \
             "relation extraction (RE), abstractive summarization and achieve comparable performance with that of the " \
             "SOTA, showing that learning rich representation of keyphrases is indeed beneficial for many other " \
             "fundamental NLP tasks."

keyphrases = tagger.predict(input_text)

print(keyphrases)
```

Output:

```
[['text documents', 'masking strategies', 'models', 'Keyphrase Boundary Infilling with Replacement', 'KBIR', 'KeyBART',
'CatSeq', 'named entity recognition', 'question answering', 'relation extraction', 'abstractive summarization', 'NLP']]
```

Training and Evaluating Keyphrase Generation Models

- **Step 1** - Import the required modules

```
from dlkp.models import KeyphraseGenerator
from dlkp.generation import KGTrainingArguments, KGModelArguments, KGDataArguments
```

- **Step 2** - Initialize the data arguments.

```
data_args = KGDataArguments(
    dataset_name="midas/inspec",
    dataset_config_name="generation",
    text_column_name="document",
    keyphrases_column_name="extractive_keyphrases",
    n_best_size=5,
    num_beams=3,
    cat_sequence=True,
)
```

- **Step 3** - Initialize the training arguments.

```
training_args = KGTrainingArguments(
    output_dir="{path_to_your_directory}",
    predict_with_generate=True,
    learning_rate=4e-5,
    overwrite_output_dir=True,
    num_train_epochs=50,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=4,
    do_train=True,
    do_eval=True,
    do_predict=False,
    eval_steps=1000,
    logging_steps=1000
)
```

Training and Evaluating Keyphrase Generation Models

- **Step 4** - Initialize the model arguments.

```
model_args = KGModelArguments(model_name_or_path="bloomberg/KeyBART")
```

- **Step 5** - Train and evaluate the model.

```
KeyphraseGenerator.train_and_eval(model_args, data_args, training_args)
```

- **Step 6** - Visualize your training progress using tensorboard.

```
tensorboard --logdir {path_to_your_log_dir}
```

Training and Evaluating Keyphrase Generation Models

- **Step 7** - Load the trained model for prediction

```
generator = KeyphraseGenerator.load(
    "{path_to_your_directory_where_model_is_saved}"
)
input_text = "Random forests or random decision forests is an ensemble learning method for classification, regression and

generator_out = generator.generate(input_text)
print(generator_out)
```

Output:

```
['random decision forests [KP_SEP] ensemble learning method [KP_SEP] classification [KP_SEP] regression [KP_SEP] data char
```

From Fundamentals to Recent Advances
A Tutorial on Keyphrasification

All materials available at
https://keyphrasification.github.io/