



The GENRE (Generative ENTITY REtrieval) system as presented in Autoregressive Entity Retrieval implemented in pytorch.

```
1 @inproceedings{decao2021autoregressive,  
2   author    = {Nicola {De Cao} and  
3               Gautier Izacard and  
4               Sebastian Riedel and  
5               Fabio Petroni},  
6   title     = {Autoregressive Entity Retrieval},  
7   booktitle = {9th International Conference on Learning Representations  
8               , {ICLR} 2021,  
9               Virtual Event, Austria, May 3-7, 2021},  
10  publisher = {OpenReview.net},  
11  year      = {2021},  
12  url       = {https://openreview.net/forum?id=5k8F6UU39V},  
13 }
```



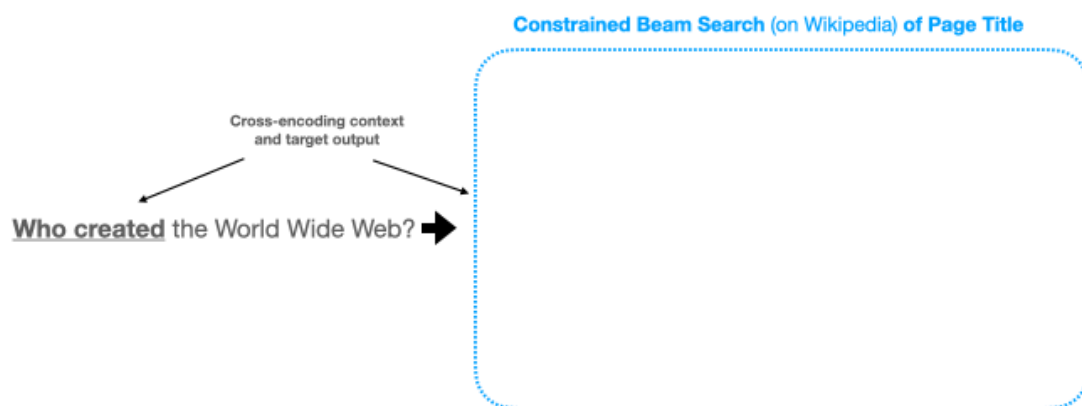
The mGENRE system as presented in Multilingual Autoregressive Entity Linking

```
1 @article{de-cao-et-al-2022-multilingual,  
2   title = "Multilingual Autoregressive Entity Linking",  
3   author = "De Cao, Nicola and  
4           Wu, Ledell and  
5           Popat, Kashyap and  
6           Artetxe, Mikel and  
7           Goyal, Naman and
```

```
8     Plekhanov, Mikhail and
9     Zettlemoyer, Luke and
10    Cancedda, Nicola and
11    Riedel, Sebastian and
12    Petroni, Fabio",
13    journal = "Transactions of the Association for Computational
14              Linguistics",
15    volume = "10",
16    year = "2022",
17    address = "Cambridge, MA",
18    publisher = "MIT Press",
19    url = "https://aclanthology.org/2022.tacl-1.16",
20    doi = "10.1162/tacl_a_00460",
21    pages = "274--290",
22 }
```

**Please consider citing our works if you use code from this repository.**

In a nutshell, (m)GENRE uses a sequence-to-sequence approach to entity retrieval (e.g., linking), based on fine-tuned BART architecture or mBART (for multilingual). (m)GENRE performs retrieval generating the unique entity name conditioned on the input text using constrained beam search to only generate valid identifiers. Here an example of generation for Wikipedia page retrieval for open-domain question answering:



For end-to-end entity linking GENRE re-generates the input text annotated with a markup:

---

↓  
The hero saved Metropolis . ➡

## Constrained Beam Search for end-to-end Entity Linking

GENRE achieves state-of-the-art results on multiple datasets.

mGENRE performs multilingual entity linking in 100+ languages treating language as latent variables and marginalizing over them:



Bidirectional  
Transformer  
Encoder

INPUT: [...] Es steht in Konkurrenz zum etablierten [START] GPS [END] - System der USA, soll aber mit den technischen Spezifikationen der Datenströme des GPS-Systems kompatibel sein. [...]

TRANSLATION: [...] It competes with the established [START] GPS [END] system in the USA, but should be compatible with the technical specifications of the data streams of the GPS system. [...]

---

## Main dependencies

- python>=3.7
- pytorch>=1.6
- fairseq>=0.10 (optional for training GENRE) **NOTE: fairseq is going through changing without backward compatibility. Install [fairseq](#) from source and use this commit for reproducibility. See here for the current PR that should fix [fairseq/master](#).**
- transformers>=4.2 (optional for inference of GENRE)

## Examples & Usage

For a full review of (m)GENRE API see: \* examples for GENRE on how to use GENRE for both pytorch fairseq and huggingface transformers; \* examples for mGENRE on how to use mGENRE.

## GENRE

After importing and loading the model and a prefix tree (trie), you would generate predictions (in this example for Entity Disambiguation) with a simple call like:

```
1 import pickle
2
3 from genre.fairseq_model import GENRE
4 from genre.trie import Trie
5
6 # load the prefix tree (trie)
7 with open("../data/kilt_titles_trie_dict.pkl", "rb") as f:
8     trie = Trie.load_from_dict(pickle.load(f))
9
10 # load the model
11 model = GENRE.from_pretrained("models/
    fairseq_entity_disambiguation_aidayago").eval()
12
13 # generate Wikipedia titles
14 model.sample(
15     sentences=["Einstein was a [START_ENT] German [END_ENT] physicist."
16 ],
17     prefix_allowed_tokens_fn=lambda batch_id, sent: trie.get(sent.
18         tolist()),
19 )
```

```
1 [[{'text': 'Germany', 'score': tensor(-0.1856)},
2    {'text': 'Germans', 'score': tensor(-0.5461)},
3    {'text': 'German Empire', 'score': tensor(-2.1858)}]]
```

---

## mGENRE

Making predictions with mGENRE is very similar, but we additionally need to map (`title`, `language_ID`) to Wikidata IDs and (optionally) marginalize over predictions of the same entity:

```
1 import pickle
2
3 from genre.fairseq_model import mGENRE
4 from genre.trie import MarisaTrie, Trie
5
6 with open("../data/lang_title2wikidataID-normalized_with_redirect.pkl",
7           "rb") as f:
8     lang_title2wikidataID = pickle.load(f)
9
10 # memory efficient prefix tree (trie) implemented with `marisa_trie`
11 with open("../data/titles_lang_all105_marisa_trie_with_redirect.pkl", "
12           "rb") as f:
13     trie = pickle.load(f)
14
15 # generate Wikipedia titles and language IDs
16 model = mGENRE.from_pretrained("../models/
17     fairseq_multilingual_entity_disambiguation").eval()
18
19 model.sample(
20     sentences=["[START] Einstein [END] era un fisico tedesco."],
21     # Italian for "[START] Einstein [END] was a German physicist."
22     prefix_allowed_tokens_fn=lambda batch_id, sent: [
23         e for e in trie.get(sent.tolist()) if e < len(model.task.
24         target_dictionary)
25     ],
26     text_to_id=lambda x: max(lang_title2wikidataID[
27         tuple(reversed(x.split(" >> ")))
28     ], key=lambda y: int(y[1:])),
29     marginalize=True,
30 )
```

```
1 [[{'id': 'Q937',
2     'texts': ['Albert Einstein >> it',
3             'Alberto Einstein >> it',
4             'Einstein >> it'],
5     'scores': tensor([-0.0808, -1.4619, -1.5765]),
6     'score': tensor(-0.0884)},
7   {'id': 'Q60197',
8     'texts': ['Alfred Einstein >> it'],
9     'scores': tensor([-1.4337]),
10    'score': tensor(-3.2058)},
11   {'id': 'Q15990626',
12     'texts': ['Albert Einstein (disambiguation) >> en'],
13     'scores': tensor([-1.0998]),
14     'score': tensor(-3.6478)}]]
```

---

## Models & Datasets

For **GENRE** use this script to download all models and this to download all datasets. See here the list of all individual models for each task and for both pytorch fairseq and huggingface transformers. See the example on how to download additional optional files like the prefix tree (trie) for KILT Wikipedia.

For **mGENRE** we only have a model available here. See the example on how to download additional optional files like the prefix tree (trie) for Wikipedia in all languages and the mapping between titles and Wikidata IDs.

Pre-trained **mBART** model on 125 languages available here.

## Troubleshooting

If the module cannot be found, preface the python command with `PYTHONPATH=.`

## Licence

GENRE is licensed under the CC-BY-NC 4.0 license. The text of the license can be found here.