Creating Event Data with NGEC

Andy Halterman

Michigan State University

University of Maryland graduate methods workshop series
October 25, 2023

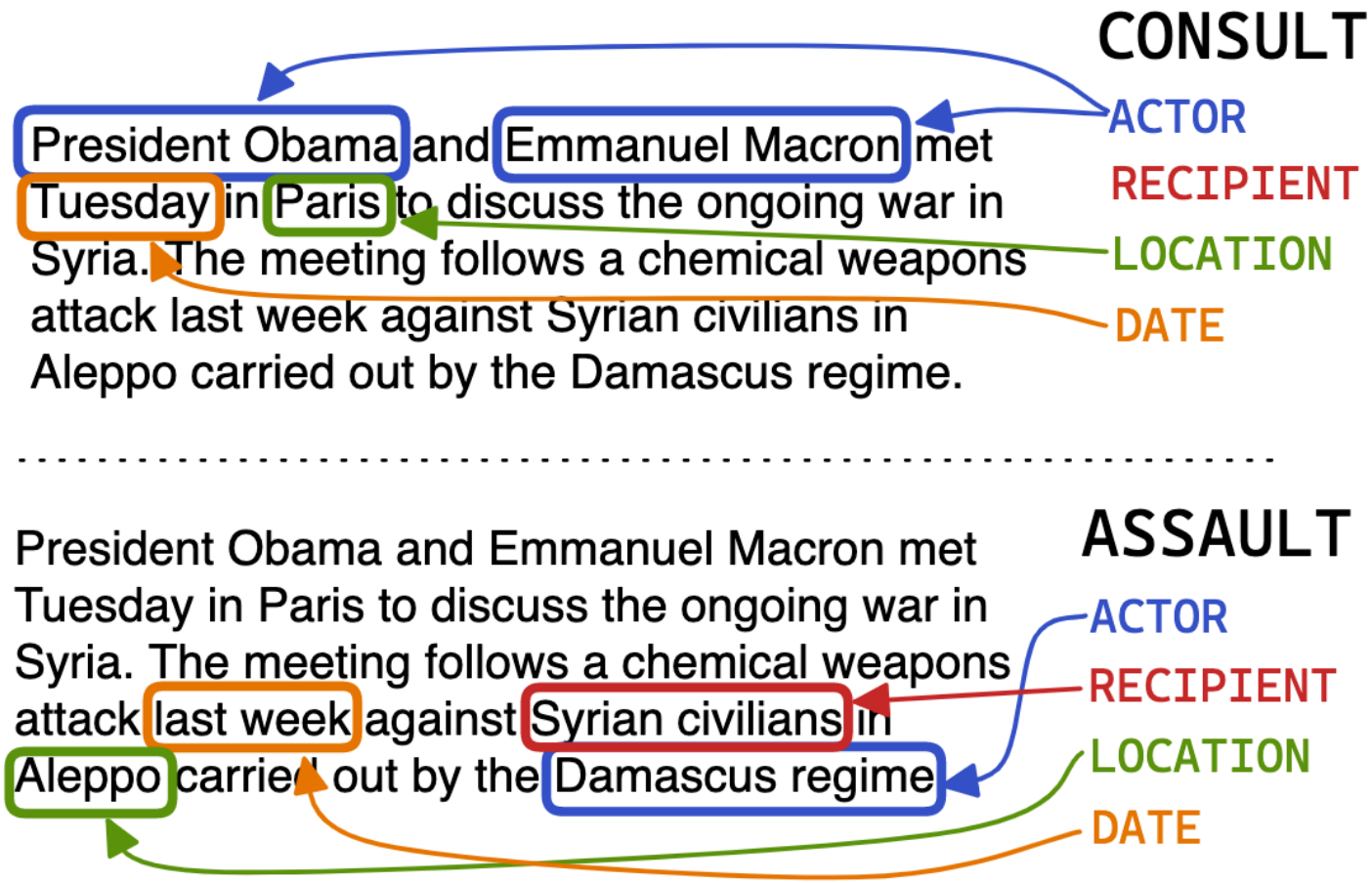
This research was sponsored by the Political Instability Task Force (PITF). The PITF is funded by the Central Intelligence Agency. The views expressed in this paper are the authors' alone and do not represent the views of the U.S. Government.

Table of Contents

What are events?

Events are structured data recording interactions between political actors.

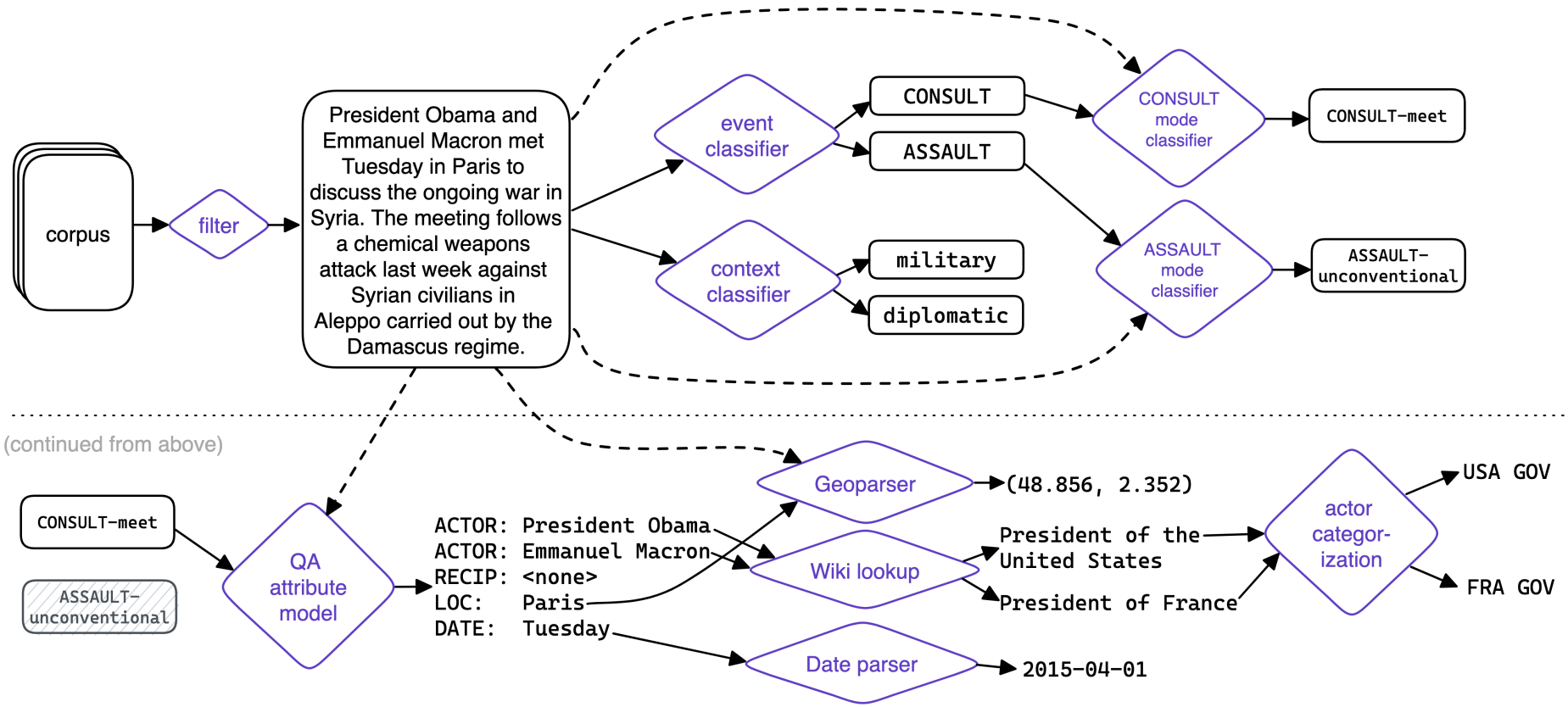
(Or, “who did what to whom, where and when?")



Six Steps

1. **Event classification**: identify the event described in a document (e.g., PROTEST, ASSAULT, AGREE,...)
2. **Sub-event ("mode") classification**: identify a more specific event type (e.g., PROTEST-riot, ASSAULT-aerial)
3. **Context classification**: identify themes or topics in a document (e.g., "human rights", "environment") using a classifier.
4. **Event attribute identification**: identifying the spans of text that report who carried out the event, who it was directed against, where it occurred, etc.
5. **Actor, location, and date resolution**: resolve extracted named actors and recipients to their Wikipedia page and locations to Geonames.
6. **Entity categorization**: Finally, we map the actor to their country and their "sector" code as defined by the PLOVER ontology (e.g., "GOV", "MIL", etc.)

Complete Pipeline



Our objective: produce a new global event dataset, but also provide tools for making custom datasets.

The PLOVER ontology

Creating events requires an ontology or schema:

- What are the event types and how are they defined?
  - E.g., what's an AID event vs. COOPERATE?
- Which events require recipients and locations?
- How do we assign "role" or "sector" codes to political actors?
  - E.g., "news reporter" $\rightarrow$ JRN, "Michael Kurilla" $\rightarrow$ USA MIL, etc.

More details: https://osf.io/preprints/socarxiv/rm5dw/download

Pipeline vs. components

Running these steps together will produce a complete event record from a document.

But we can also run them separately. E.g.:

- Identify documents with a specific event type for qualitative analysis (event and mode classification).
- Classify documents by their general topic or issue (context classification)
- Extract the names and locations from a set of documents with known events (attribute identification)
- Extract the locations from a set of relevant documents (geolocation).
- Resolve a list of names to Wikipedia articles (actor resolution)

Table of Contents

Detecting events in text

- NGEC's default event detector uses a document-level classifier
- E.g., "does this story describe a PROTEST/ASSAULT/etc?"
- This lets us do document-level event detection, vs. sentence-level like most previous coders.

How to collect data?

- Collect document-level annotations by hand (slow, limited diversity, but high quality)
- Use synthetic stories (fast, cheap, lower quality). Described further in Halterman (2023).

Getting started with synthetic data

Step 1: Write Headlines

SANCTION: `{country_1} Downgrades Diplomatic Ties with {country_2} Amid Ongoing Spat`

PROTEST: `Over Half of {country_1} Teachers Now Out on Wildcat Strike`

Step 2: Generate variants

`South Korea Downgrades Diplomatic Ties with Cyprus Amid Ongoing Spat (AFP) --`

`Over Half of Estonia Teachers Now Out on Wildcat Strike (BBC Monitoring) --`

Generating Synthetic Training Data

Step 3: Generate stories (gpt2-xl)

`South Korea Downgrades Diplomatic Ties with Cyprus Amid Ongoing Spat (AFP) --`

> *[SYNTH] Diplomatic ties between Seoul and its neighbor Cyprus were downgraded indefinitely in August amid bitter feuds and political infighting, Seoul said Sunday, amid lingering hostility toward the Mediterranean island nation.*
>
> *"The North Korean nuclear issue and the Cyprus situation are affecting Korea-Cyprus ties due to the unresolved issue of unification," a Foreign Ministry official said at a briefing on the issue. ...*

`Over Half of Estonia Teachers Now Out on Wildcat Strike (BBC Monitoring) --`

> *[SYNTH] "The largest teachers' strike in Estonia since 1997 could have important implications for the country's public education system.*
>
> *Starting on April 20, an estimated 25,500 of the nation's 85,000 elementary and secondary school teachers are on strike, a strike by almost one in seven Estonian teachers*
>
> *...*

Synthetic training data

```
with jsonlines.open("gpt_synthetic_events_2023-04-06.jsonl", "r") as f:
    data = list(f.iter())
```

```
[SYNTHETIC TEXT]
"The largest teachers' strike in Estonia since 1997 could have important
implications for the country's public education system.

Starting on April 20, an estimated 25,500 of the nation's 85,000 elementary and
secondary school teachers are on strike, a strike by almost one in seven
Estonian teachers.

A full-day school drop-out will make it far more difficult for the public to
find teaching jobs as many teachers say they have already been unable to offer
employment.

Estonia's state-run educational administration says it lost roughly 40 percent
of its staff in the first two weeks of April. But the teachers' union suggests
it could be twice as many.
```

A quick and accurate classifier

You can build a simple and accurate document level classifier with two steps:

1. Embed documents using a transformer-based model.
   - We'll use a sentence-transformer model (Reimers and Gurevych 2019) that provides fixed-width embedding for an input document that's optimized for downstream classification and semantic similarity tasks.
2. Fit a classifier of your choice on the embeddings (e.g., logistic regression, SVM, etc).

```
model = SentenceTransformer('sentence-transformers/paraphrase-mpnet-base-v2')

X_train = model.encode(text_train, show_progress_bar=True)
X_val = model.encode(text_val, show_progress_bar=True)
```

Not fine tuning the embeddings lowers performance somewhat, but makes it much faster.

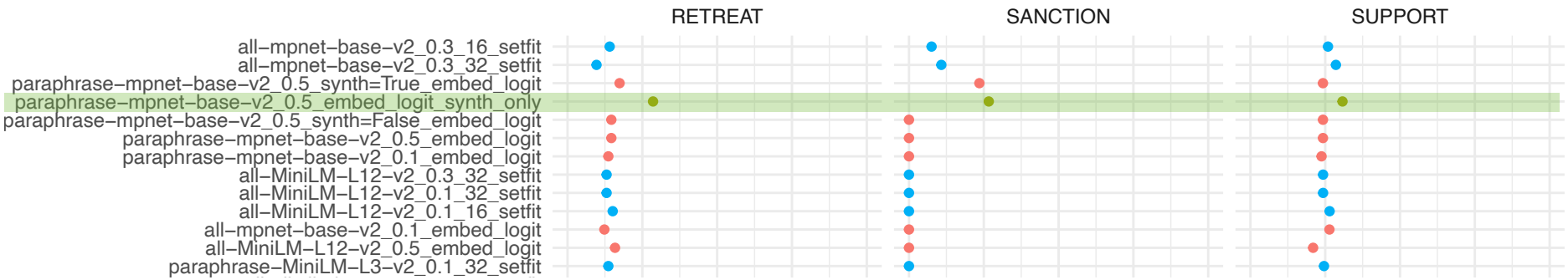
Handling noisy labels

The synthetic labels are noisy: both because of errors in the LLM, but also because our “negative" examples aren’t always true negatives.

```
def fit_initial_model(synth_df, model_name):
    model = load_model(model_name)
    encoded = model.encode(synth_df['text'].values, show_progress_bar=True,
                           device='cuda').tolist()
    y_train = synth_df['label']
    clf = SVC(class_weight="balanced",
              kernel="linear",
              probability=True,
              C=0.1)
    clf.fit(encoded, y_train)
    pred = pd.DataFrame(clf.predict_proba(encoded))
    # rename columns with the event names
    pred.columns = clf.classes_
    return pred
```

```
candidate_neg = synth_df[synth_df[event] < 0.04]
sample_size = min(candidate_neg.shape[0],
                  train_pos_synth.shape[0] * 3)
train_neg_synth = candidate_neg.sample(sample_size).copy()
train_neg_synth['label'] = 0
```

Training a classifier



- Hand collected labels are better, but only if they're high quality and numerous.
- Synthetic data allows rapid prototyping and decent performance.

Benefits/limitations of the approach

- Document-level labels are easier to collect than span-level labels.
- Document classifiers are easy to train and evaluate.
- But struggles with multiple events, hypotheticals, negation.

Table of Contents

Geoparsing example

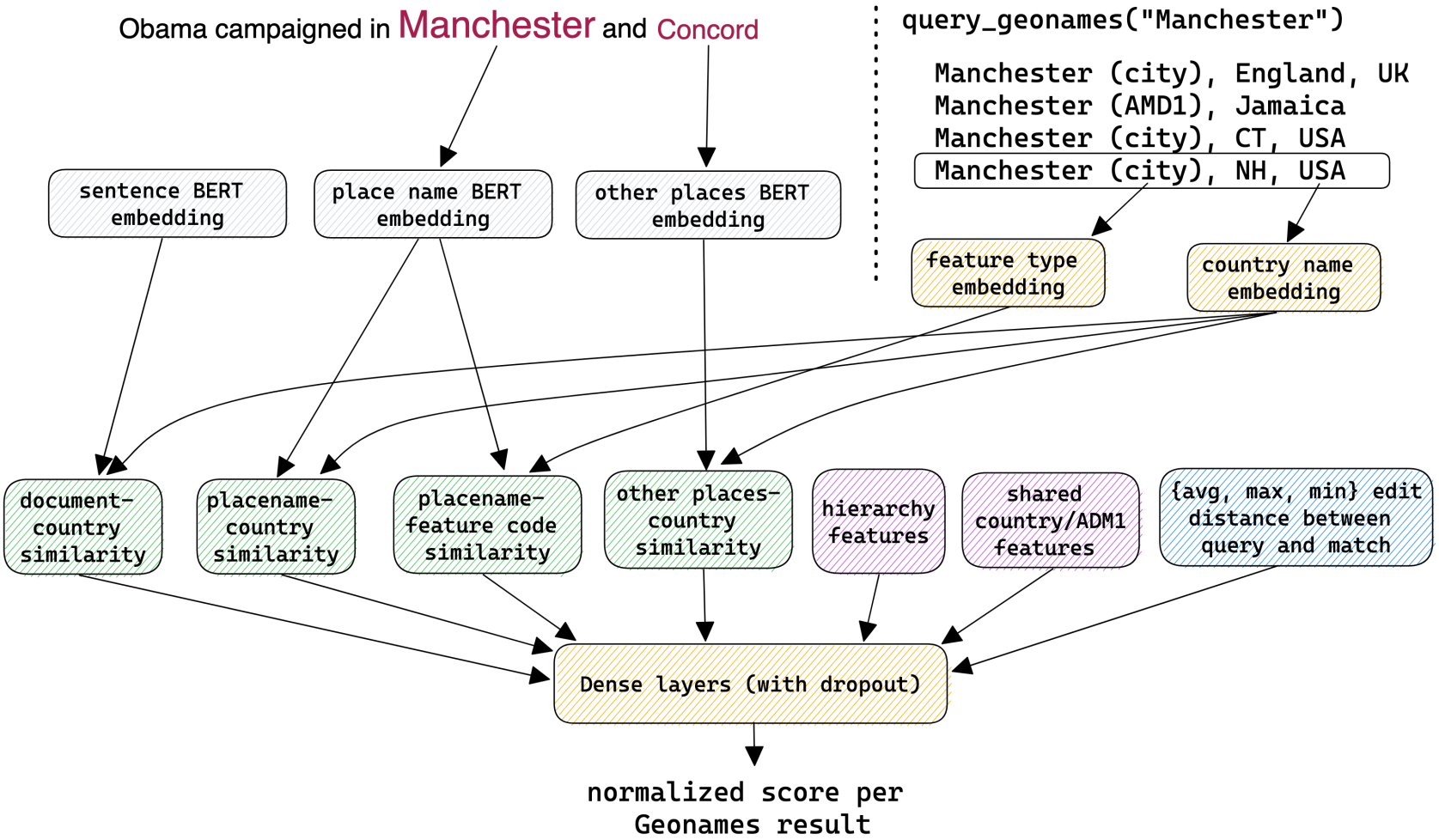
```
from mordecai3 import Geoparser

# Create the Geoparser object
geo = Geoparser("NGEC/assets/mordecai_2023-03-28.pt")

output = geo.geoparse_doc("The Mexican government sent 300 National Guard troopers to bolst
```

```
{'doc_text': 'The Mexican government sent 300 National Guard troopers to '
             'bolster the southern state of Guerrero on Tuesday, where a local '
             'police chief and 12 officers were shot dead in a brutal ambush '
             'the day before.',
 'event_location_raw': '',
 'geolocated_ents': [{'admin1_code': '12',
                      'admin1_name': 'Guerrero',
                      'admin2_code': '',
                      'admin2_name': '',
                      'city_id': '',
                      'city_name': '',
                      'country_code3': 'MEX',
                      'end_char': 97,
                      'feature_class': 'A',
                      'feature_code': 'ADM1',
                      'geonameid': '3527213',
                      'lat': 17.66667,
                      'lon': -100.0,
                      'name': 'Estado de Guerrero',
                      'score': 1.0,
                      'search_name': 'Guerrero',
                      'start_char': 89}]}
```

Mordecai3 under the hood



Geoparsing results

feature_c	featur	countr	lat	lon	name	admi	admin1_name	geonameid	search_name
PPLA2	P	SYR	36.89095	38.35347	‘Ayn al ‘Arab	9	Aleppo	172256	Kobanî
PPL	P	SYR	34.4337	40.98887	Bāghūz	7	Deir ez-Zor	171944	Baghuz
PPLA3	P	SYR	36.39042	41.14983	Al Ḩawl	1	Al-Hasakah	173797	al-Hawl
PPLA3	P	SYR	36.39042	41.14983	Al Ḩawl	1	Al-Hasakah	173797	al-Hawl
PPL	P	SYR	34.4337	40.98887	Bāghūz	7	Deir ez-Zor	171944	Baghuz
PPLA3	P	SYR	36.39042	41.14983	Al Ḩawl	1	Al-Hasakah	173797	al-Hawl
ADM1	A	SYR	35.16667	40.28333	Deir ez-Zor Governorate	7	Deir ez-Zor	170792	Euphrates
PPLC	P	SYR	33.5102	36.29128	Damascus	13	Dimashq	170654	Damascus
PPL	P	SYR	34.4337	40.98887	Bāghūz	7	Deir ez-Zor	171944	Baghuz
PPL	P	SYR	34.4337	40.98887	Bāghūz	7	Deir ez-Zor	171944	Baghuz
PPL	P	SYR	34.4337	40.98887	Bāghūz	7	Deir ez-Zor	171944	Baghuz
PPL	P	SYR	34.4337	40.98887	Bāghūz	7	Deir ez-Zor	171944	Baghuz
ADM1	A	SYR	35.16667	40.28333	Deir ez-Zor Governorate	7	Deir ez-Zor	170792	Euphrates
PPLA	P	SYR	35.95283	39.00788	Ar Raqqah	4	Ar-Raqqah	172955	Raqqa
PPLA2	P	SYR	35.01982	40.45154	Al Mayādīn	7	Deir ez-Zor	173480	Mayadin
PPLA2	P	SYR	34.45226	40.91854	Ālbū Kamāl	7	Deir ez-Zor	174448	Bukamal

Geoparsing wrap-up

- The geoparser can work on structured data as well. The lack of context makes it challenging, but structured data often has country info that can be used.
- The geoparser makes errors–we're still making incremental improvements.
- Note that this doesn't handle event geolocation–identifying the coordinates where an event occurred.
- To do that, we use information from the "event attribute" model (not covered today) and look for overlap between the event's location and the geoparser results.

Code for the geoparser is in a separate repo:
https://github.com/ahalterman/mordecai3

Table of Contents

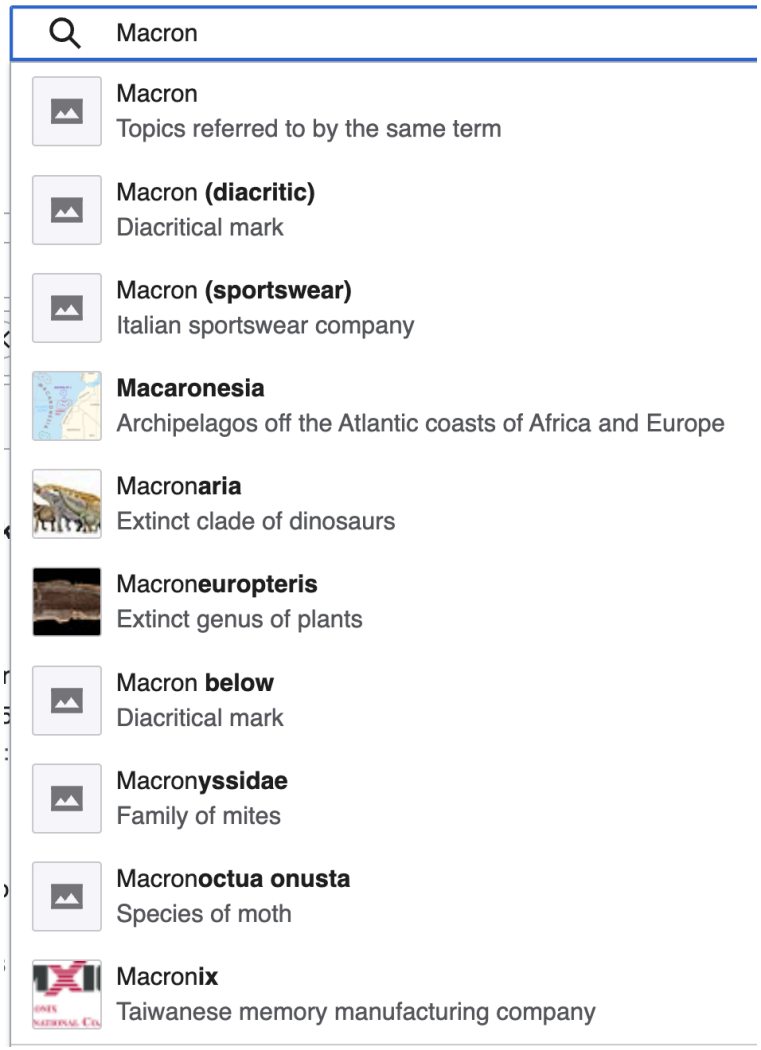
Why resolve actors?

After we extract actors and recipients, we want to resolve them to Wikipedia. (Similarly to resolving place names to Geonames)

Why?

- A single, unique identifier: “Joe Biden" and “President Biden" refer to the same person and we might want to study the actor specifically.
- It provides information on their roles: we want to know the country, office, etc. of actors.

How the model works



Resolving actors to Wikipedia is harder than (we) expected!

Algorithm:

- Query offline Wikipedia for a match on title, redirects, alternative names
- If no results, allow a fuzzy search.
- Pick the best result using:
  - Unique perfect match on title, redirect.
  - Neural similarity between the story context and the actor description.
  - Neural similarity between the actor name and Wiki title.
  - (High) neural similarity on alternative names

Loading the data and model

We'll use a set of news stories from the *Guardian* that include keywords related to the Syrian Democratic Forces.

We can read in the data, then load the ActorResolver model from NGEC.

```
with jsonlines.open("guardian_sdf_sample.jsonl", "r") as f:
    data = list(f.iter())
```

```
from NGEC import ActorResolver
actor_resolution_model = ActorResolver(spacy_model=nlp,
                                       base_path="../NGEC/assets/",
                                       save_intermediate=False,
                                       gpu=False)
```

*The `AttributeModel` for identifying actor and recipient spans, dates, and locations uses similar syntax.

Extracting people and organizations

Because we're not running the pipeline in its entirety, we don't have the event actors and recipients from a previous step. But sometimes, we just want to know what people and organizations and mentioned in a story.

We can use spaCy, a Python library for NLP, to extract these entities:

```
docs = list(nlp.pipe([i['event_text'] for i in data]))
# make a list of lists with the PERSON and ORG entities
entities = []
for doc in docs:
    for ent in doc.ents:
        if ent.label_ in ['PERSON', 'ORG']:
            ent_text = ent.text
            d = {"entity": ent_text,
                 "context": ent.sent.text}
            entities.append(d)
```

Running the model

We can then iterate through our list of extracted people and organizations and try to resolve them to their Wikipedia page.

```
wikis = []
for ent in tqdm(entities):
    wiki = actor_resolution_model.query_wiki(ent['entity'], context = ent['context'])
    if not wiki:
        wiki = {"search_term": ent['entity'],
                "title": None}
    else:
        wiki['search_term'] = ent['entity']
    wikis.append(wiki)
```

Results

On the whole, it works pretty well:

```
Isis                        ---> Islamic State (Salafi jihadist militant Islamist group)
Brett McGurk                ---> Brett McGurk (American diplomat)
the Red Cross               ---> International Red Cross and Red Crescent Movement (International humanita
Scott Morrison              ---> Scott Morrison (Prime Minister of Australia from 2018 to 2022)
Donald Trump                ---> Donald Trump (President of the United States from 2017 to 2021)
Abu Bakr al-Baghdadi        ---> Abu Bakr al-Baghdadi (Leader of the Islamic State from 2013 to 2019)
the International Rescue Committee ---> International Rescue Committee (Nongovernmental humanitarian or
```

Some unintuitive but correct resolutions::

```
Yago Riedijk                 ---> Shamima Begum (Jihadist and former British citizen (born 1999))
British woman who was stripped of citizenship for joining ISIS. She's married to Yago Riedijk,
```

...and some errors:

```
Some errors:
Fabrizio Carboni    ---> Fabrizio Carbone (Italian and Swiss physicist)
SDF                 ---> SDF insurgency in Northern Aleppo (Guerilla attacks carried out by the SDF)
Isis                ---> Isis (journal) ()
```

“Coding" Wikipedia pages

The Wikipedia page/title is useful, but we often want to aggregate or categorize actors.

```
wiki = actor_resolution_model.query_wiki("Ben Rhodes", context = """The former Obama
adviser Ben Rhodes said: "We all owe him our gratitude - he
literally made us safer.""")
code = actor_resolution_model.wiki_to_code(wiki)
```

```
[{'code_1': 'ELI',
  'code_2': '',
  'country': 'USA',
  'description': 'previously held a GOV role, so coded as ELI.',
  'pattern': 'NA',
  'source': 'Infobox',
  'wiki': 'Ben Rhodes (White House staffer)'}]
```

Second example

To resolve actors to their codes, we use information from the “infobox", the short page description, or the intro para.

```
wiki = actor_resolution_model.query_wiki("Niloufar Hamedi", context = """The two
journalists are Niloufar Hamedi,
who broke the news of Amini’s death for wearing
her headscarf too loose, and Elaheh Mohammadi,
who wrote about Amini’s funeral.""")
code = actor_resolution_model.wiki_to_code(wiki)
```

```
{’actor_wiki_job’: ’Iranian journalist’,
  ’code_1’: ’JRN’,
  ’code_2’: ’’,
  ’conf’: 0.9999999999998757,
  ’country’: ’IRN’,
  ’description’: ’journalist’,
  ’pattern’: ’journalist’,
  ’query’: ’journalist’,
  ’source’: ’Wiki short description’,
  ’wiki’: ’Niloofar Hamedi’}
```

We compare the extracted info to a short list of generic job titles.

Getting started

- https://github.com/ahalterman/NGEC
  - See “Quickstart guide"
- https://github.com/ahalterman/mordecai3
- Paper: https://arxiv.org/pdf/2304.01331.pdf