Improving Toponym Resolution with Better Candidate Generation and Transformer-based Reranking

Anonymous ACL submission

Abstract

Geocoding is the task of converting location mentions in text into structured geospatial data. We propose a new architecture for geocoding, SSPART, that first uses information retrieval techniques to generate a list of candidate entries from the geospatial ontology, and then reranks the candidates using a transformer-based neural network. The reranker compares the location mention to each candidate entry, while incorporating additional information such as the entry's population, the entry's type of location, and the sentences surrounding the mention in the text. Our proposed toponym resolution framework achieves state-of-the-art performance on multiple datasets. Code and models are available at `https://<anonymized>`.

1 Introduction

Geospatial information extraction has seen a recent surge in interest from the natural language processing community due to its critical role in tasks such as geographical document classification and retrieval (Bhargava et al., 2017), historical event analysis based on location data (Tateosian et al., 2017), tracking the evolution and emergence of infectious diseases (Hay et al., 2013), and disaster response mechanisms (Ashktorab et al., 2014; de Bruijn et al., 2018). Such information extraction can be challenging because different geographical locations can be referred to by the same place name (e.g., *San Jose* in Costa Rica vs. *San Jose* in California, USA), and different place names can refer to the same geographical location (e.g., *Leeuwarden* and *Ljouwert* are two names for the same city in the Netherlands). It is thus critical to resolve these place names by linking them with their corresponding coordinates from a geospatial ontology or knowledge base.

Geocoding, also called toponym resolution or toponym disambiguation, is the subtask of geoparsing that disambiguates place names (known as *toponyms*) in text. The goal of geocoding is, given a textual mention of a location, to choose the corresponding geospatial coordinates, geospatial polygon, or entry in a geospatial database. Most existing geocoding systems produce geospatial ontology entries by first generating candidate entries with an information retrieval system and then reranking those entries with a supervised feature-based classifier using a variety of hand-engineered heuristics (Speriosu and Baldridge, 2013; Zhang and Gelernter, 2014; DeLozier et al., 2015; Kamalloo and Rafiei, 2018; Wang et al., 2019). More recently, deep neural network approaches to geocoding have been introduced that predict small tiles of the map rather than ontology entries (Gritta et al., 2018; Cardoso et al., 2019; Kulkarni et al., 2021). The neural network approaches have been generally more successful, but because of their output encoding, they do not naturally produce an ontology entry, which may contain a variety of metadata needed by a user.

We propose a new architecture SSPART (Search, Sort by Population, And Rerank by Transformer), shown in Figure 1, which has the advantages of both: it uses pre-trained deep neural networks for the improved robustness in matching place names, while leveraging a generate-then-rank architecture to produce ontology entries as output instead of map tiles. SSPART generates candidate ontology entries with an information retrieval system coupled with a simple population heuristic, and then uses a pre-trained transformer-based classifier to rerank the candidate entries based on analyzing the place name, candidate ontology entry, lexical context, and geospatial ontology features.

Our work makes the following contributions:

- Our proposed architecture for geocoding achieves new state-of-the-art performance on multiple datasets.
- Our candidate generator, based on simple information retrieval techniques, outperforms recent more complex neural models.

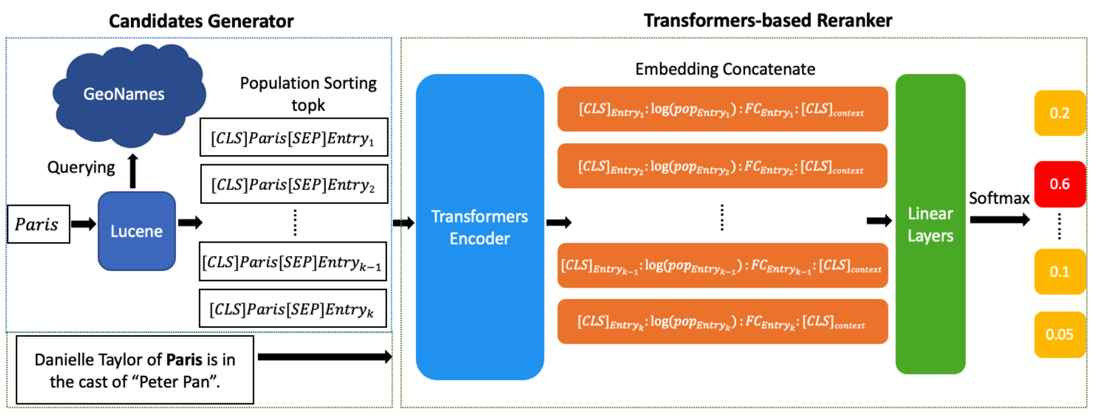
Candidates Generator
Transformers-based Reranker
GeoNames
Population Sorting topk
Querying
Paris
Lucene
$[CLS]Paris[SEP]Entry_1$
$[CLS]Paris[SEP]Entry_2$
$[CLS]Paris[SEP]Entry_{k-1}$
$[CLS]Paris[SEP]Entry_k$
Danielle Taylor of **Paris** is in the cast of "Peter Pan".
Transformers Encoder
Embedding Concatenate
$[CLS]_{Entry_1}:\log(pop_{Entry_1}):FC_{Entry_1}:[CLS]_{context}$
$[CLS]_{Entry_2}:\log(pop_{Entry_2}):FC_{Entry_2}:[CLS]_{context}$
$[CLS]_{Entry_{k-1}}:\log(pop_{Entry_{k-1}}):FC_{Entry_{k-1}}:[CLS]_{context}$
$[CLS]_{Entry_k}:\log(pop_{Entry_k}):FC_{Entry_k}:[CLS]_{context}$
Linear Layers
Softmax
0.2
0.6
0.1
0.05

Figure 1: The architecture of our model: Search, Sort by Population, And Rerank by Transformer (SSPART).

- Our reranker is the first application of pre-trained transformers for encoding location mentions and context for toponym resolution.
- Our evaluation includes a wider variety of geocoding evaluation metrics than prior work, applying both database entry correctness metrics and point distance metrics.

2 Related Work

The geocoding task can be classified into two distinct categories: document level and mention level. The objective of document-level geocoding is to match an entire text to a corresponding location, such as geolocating Twitter users or microblog posts (Roller et al., 2012; Rahimi et al., 2015; Lee et al., 2015; Rahimi et al., 2017; Hoang and Mothe, 2018; Kumar and Singh, 2019; Luo et al., 2020) and geographic document retrieval and classification (Gey et al., 2005; Adams and McKenzie, 2018). The objective of mention-level geocoding is to match phrases within a text to their corresponding locations. While this task is conceptually related to Wikipedia linking, it differs in that geospatial ontologies include only the geospatial concepts, not in-text examples. (See also appendix C). Mention-level geocoding is typically preceded by geotagging, a named entity recognition task that finds location mentions in a text. The current work focuses on mention-level geocoding, not geotagging.

Many systems for geocoding used hand-crafted rules and heuristics to predict geospatial labels for place name. Examples include the Edinburgh geoparser (Grover et al., 2010), Tobin et al. (2010), Lieberman et al. (2010), Lieberman and Samet (2011), CLAVIN (Berico Technologies, 2012), GeoTxt (Karimzadeh et al., 2013), and Laparra and Bethard (2020). The most common features and heuristics were based on string matching, population count, and type of place (city, country, etc.).

As more shared tasks and annotated datasets were proposed, geocoding systems began to take the heuristics of rule-based systems and use them as features in supervised machine learning models, including logistic regression (WISTR, Speriosu and Baldridge, 2013), support vector machines (Martins et al., 2010; Zhang and Gelernter, 2014), random forests (MG, Freire et al., 2011; Lieberman and Samet, 2012), stacked LightGBMs (DM_NLP, Wang et al., 2019) and other statistical learning methods (Topocluster, DeLozier et al., 2015; CBH, SHS, Kamalloo and Rafiei, 2018). These systems typically operated in a two-step generate-then-rerank framework, where first an information retrieval system produced candidate geospatial ontology entries, a supervised machine-learning model produced a score for each candidate, and the candidates were reranked by those scores.

Recently, deep learning methods have been introduced for toponym resolution. Some such models perform only the generation step, converting mentions and entities into vectors and measuring vector similarity, but ignoring the context around the mentions (Hosseini et al., 2020; Ardanuy et al., 2020). We show that simple information retrieval techniques outperform such models.

Other deep learning models approach geocoding as a one-step classification problem by dividing the Earth's surface into an $N \times N$ grid, where the neural network attempts to map place names and their features to one of these $N \times N$ categories (CamCoder, Gritta et al., 2018; Cardoso et al., 2019; MLG, Kulkarni et al., 2021). Each system has a unique neural architecture for combining inputs to make predictions, based on convolutional neural networks (CNNs) (CamCoder, Gritta et al., 2018; MLG, Kulkarni et al., 2021) or recurrent neural networks (RNNs) (Cardoso et al., 2019). Though the grid-based output formulation results in a large label space for classification, the neural network models are able to more flexibly encode location mentions and the nearby context, leading to performance gains in distance-based metrics across several corpora.

Our proposed approach combines the tight ontology integration of the generate-and-rerank systems with the robust text encoding of the deep neural network grid-classification systems.

3 Proposed Methods

We define the task of toponym resolution as follows. We are given an ontology or knowledge base with a set of entries $E = \{e_1, e_2, ..., e_{|E|}\}$. Each input is a text made up of sentences $T = \{t_1, t_2, \dots, t_{|T|}\}$ and a list of location mentions $M = \{m_1, m_2, ..., m_{|M|}\}$ in the text. The goal is to find a mapping function $e_j = f(m_i)$ that maps each location mention in the text to its corresponding entry in the ontology.

We approach toponym resolution using a candidate generator followed by a candidate reranker. The candidate generator, $G(m, E) \to E_m$, takes a mention m and ontology E as input, and generates a list of candidate entries E_m, where $E_m \subseteq E$ and $|E_m| \ll |E|$. As the candidate generator must search a large ontology and produce only a short list of candidates, the goal for G will be high recall and high runtime efficiency. The candidate reranker, $R(m, E_m) \to \widehat{E_m}$, takes a mention m and the list of candidate ontology entries E_m, and sorts them by their relevance or importance to produce a new list, $\widehat{E_m}$. As the candidate ranker needs to work only with a short list of candidates, the goal for R will be high precision, especially at rank 1, with less of a focus on runtime efficiency.

3.1 Candidate Generator

Our candidate generator is inspired by prior work on geocoding in using information retrieval techniques to search for candidates in the ontology (Grover et al., 2010; Berico Technologies, 2012). Accurate candidate generation is essential, since the generator's recall is the ceiling performance for the reranker. As we will see in section 5, our proposed candidate generator alone is competitive with complex end-to-end systems from prior work.

Algorithm 1: Candidate generator.

```
Input: a location mention, $m$
       a maximum number of candidates, $k$
       the GeoNames ontology, $E$
Output: a list of candidate entries $E_m$
// Index ontology
$I \leftarrow \emptyset$
for $e \in E$ do
    $name \leftarrow \text{CANONICALNAME}(E, e)$
    $synonyms \leftarrow \text{SYNONYMS}(E, e)$
    for $n \in \{name\} \cup synonyms$ do
        $I \leftarrow I \cup \{\text{CREATEDOCUMENT}(e, n)\}$
// Search for candidates
$E_m \leftarrow \emptyset$
for $t \in$ { EXACT, FUZZY, CHARACTERNGRAM,
  TOKEN, ABBREVIATION, COUNTRYCODE } do
    $E_m \leftarrow \text{SEARCH}(index, m, t)$
    if $E_m \neq \emptyset$ then
        break
// Sort by population
$key \leftarrow (e \to \text{POPULATION}(E, e))$
$E_m \leftarrow \text{SORT}(E_m, key)$
// Select top entries
return top $k$ elements of $E_m$
```

Our sieve-based approach, detailed in alg. 1, tries searches ordered from least precise to most precise until we find ontology entries that match the location mention. We create one document in the index for each name n_e of an entry e in the GeoNames ontology. A location mention m is matched to a name n_e by attempting a search with each of the following matching strategies, in order:

EXACT m exactly matches (ignoring whitespace) the string n_e

FUZZY m is within a 2 character Levenshtein edit distance (ignoring whitespace) of n_e

CHARACTERNGRAM m has at least one character 3-gram overlap with n_e

TOKEN m has at least one token (according to the Lucene StandardAnalyzer) overlap with n_e

ABBREVIATION m exactly matches the capital letters of n_e

COUNTRYCODE e is a country and m exactly matches a e's country code

Once one of the searches has retrieved a list of matching names, we recover the ontology entry for each name, sort those ontology entries by their population in the GeoNames ontology, and return the k most populous ontology entries. This list, E_m is then the input to the candidate reranker.

3.2 Candidate Reranker

Our candidate reranker is inspired by prior work on medical concept normalization (Xu et al., 2020;

Ji et al., 2020), extended to incorporate aspects uniquely important for geocoding. Similar to prior work, and as shown in fig. 1, the candidate reranker takes a mention to be classified, m, and the list of candidate entities from the candidate generator, E_m, encodes them with a transformer neural network, and uses these encoded representations to perform classification over the list to select the most probable entry. For each candidate $e = E_{m_i}$, the input to the transformer is of the form `[CLS]` m `[SEP]` C^e `[SEP]` S^e_1 `[SEP]` $\ldots$ `[SEP]` $S^e_{|S^e|}$ `[SEP]`, where $C^e = \textsc{CanonicalName}(E, e)$ is the canonical name of e in the ontology, and $S^e = \textsc{Synonyms}(E, e)$ is the list of alternate names of e in the ontology. We then represent each candidate with the contextualized representation of its `[CLS]` token from the last layer of the transformer, a vector we will refer to as `[CLS]`$_{E_{m_i}}$. Note that `[CLS]`$_{E_{m_i}} \in \mathbb{R}^H$, where H is the size of the transformer's contextualized representations.

We extend this architecture with three features that are important for geocoding: population, type of geographical feature, and mention context.

Population: Locations in text are more likely to refer to high population places than low population places (e.g., Paris, France vs. Paris, Texas, USA). We look up the population of E_{m_i} in the ontology, and take the logarithm of that population. We refer to this scalar as $\log(POP_{E_{m_i}})$.

Feature Code: Locations in text are more likely to refer to some types of geographical features than others (e.g., San José, the capital of Costa Rica, vs. San José, the province). We look up the feature code[1] of E_{m_i} in the ontology, and transform the feature code into a one-hot vector $FC_{E_{m_i}} \in \mathbb{R}^N$ where N is the total number of feature codes in the GeoNames ontology

Mention context: The text around a mention may provide clues (e.g., the context *Minnesota State Patrol urges motorists to drive with caution... in Becker, Clay, and Douglas* suggests that *Clay* refers to Clay County, Minnesota, even though Clay County, Missouri is more populous). We find the sentence in the text T containing m, $T_{\textsc{Sent}(m)}$, and encode the c-sentence window including it with the same transformer as was used to encode m. The input is of the form `[CLS]` $T_{\textsc{Sent}(m)-c} \cdots T_{\textsc{Sent}(m)+c}$ `[SEP]`. We take the contextualized representation of the `[CLS]` token from the last layer of the transformer, `[CLS]`$_{\textsc{Sent}(m)\pm c} \in \mathbb{R}^H$.

[1]In GeoNames, for example, the feature code PPLC means *capital of a political entity*. GeoNames feature codes are listed in detail at http://download.geonames.org/export/dump/featureCodes_en.txt

Dataset	Train		Dev.		Test	
	Toponyms	Articles	Toponyms	Articles	Toponyms	Articles
LGL	3112	411	419	58	931	119
GeoWebNews	1641	140	281	20	477	40
TR-News	925	82	68	11	282	25

Table 1: Numbers of articles and manually annotated toponyms in the train, development, and test splits of the toponym resolution corpora.

To combine all of these new features with the traditional representation of the candidate from prior work, `[CLS]`$_{E_{m_i}}$, we concatenate all the vectors before the classification layers. So the concatenated vector for each candidate entry E_{m_i} would be $V_{E_{m_i}} =$ `[CLS]`$_{E_{m_i}} \oplus \log(POP_{E_{m_i}}) \oplus FC_{E_{m_i}} \oplus$ `[CLS]`$_{\textsc{Sent}(m)\pm c}$ with $V_{E_{m_i}} \in \mathbb{R}^{2H+N+1}$. Aggregating across the k candidates, we form a matrix $M_{E_m} \in \mathbb{R}^{k\times(2H+N+1)}$. We then feed this matrix into two linear layers with the weights $W_1 \in \mathbb{R}^{150\times(2H+1+N)}$ and $W_2 \in \mathbb{R}^{1\times 150}$, and compute a standard classification loss:

$$L_R = y \cdot log(\text{softmax}((M_{E_m} W_1^T) W_2^T)) \quad (1)$$

where y is a one-hot vector, and $|y| = |E_m|$. At prediction time, the model predicts the entry with the highest probability in the softmax.

4 Experiments

4.1 Datasets

We conduct experiments on three toponym resolution datasets[2]. Local Global Lexicon (LGL; Lieberman et al., 2010) was constructed from 588 news articles from local and small U.S. news sources. GeoWebNews (Gritta et al., 2019) was constructed from 200 articles from 200 globally distributed news sites. TR-News (Kamalloo and Rafiei, 2018) was constructed from 118 articles from various global and local news sources. As there are no standard publicly available splits for these datasets, we split each dataset into a train, development, and

[2]We also considered Weissenbacher et al. (2019), but the test set was never released (we requested it from the authors), making comparison to prior work on that dataset difficult.

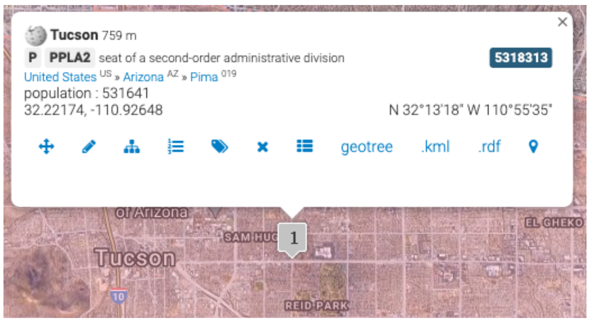
Tucson 759 m
P PPLA2 seat of a second-order administrative division 5318313
United States US » Arizona AZ » Pima 019
population : 531641
32.22174, -110.92648 N 32°13'18" W 110°55'35"

Figure 2: An entry for *Tucson* in GeoNames

test set according to a 70%, 10% , and 20% ratio. To enable replicability, we will release these splits upon publication. The statistics of all datasets are shown in table 1.

4.2 Database

GeoNames is a crowdsourced database of geospatial locations, with almost 7 million entries and a variety of information such as geographic coordinates (latitude and longitude), alternative names, feature type (country, city, river, mountain, etc.), population, elevation, and positions within a political geographic hierarchy. An example entry from GeoNames is shown in fig. 2. The data in GeoNames comes from multiple sources[3], such as public and open gazetteers, which can vary in quality, scope, resolution, or age (Ahlers, 2013). Users can edit data in a wiki-like interface.

In our experiments, the GeoNames ontology plays an important role in both the candidate generator and the candidate reranker. The candidate generator produces a list of candidate entries from GeoNames and the candidate reranker selects the best entry by utilizing multiple meta-data obtained from GeoNames, including canonical names, alternate names, populations, and feature codes.

4.3 Evaluation Metrics

There is not yet agreement in the field of toponym resolution on a single evaluation metric. Therefore, we gather several metrics from prior work and use all of them for evaluation. We consider metrics both measuring matching of ontology entry IDs and measuring geographical distance between system predictions and human annotations.

Accuracy is the number of location mentions where the system predicted the correct database entry ID, divided by the number of location mentions. Higher is better, and a perfect model would have accuracy of 1.0.

Accuracy@161km measures the fraction of system-predicted (latitude, longitude) points that were less than 161 km away from the human-annotated (latitude, longitude) points. Higher is better, and a perfect model would have Accuracy@161km of 1.0.

Mean error distance calculates the mean over all predictions of the distance between each system-predicted and human-annotated (latitude, longitude) point. Lower is better, and a perfect model would have a mean error distance of 0.0.

Area Under the Curve (AUC) calculates the area under the curve of the distribution of geocoding error distances. Lower is better, and a perfect model would have a mean error distance of 0.0.

4.4 Implementation details

We implement the candidate reranker with Lucene[4] v8.4.1 under Java 1.8. When indexing GeoNames, we also index countries under their adjectival forms in Wikipedia[5]. We implement the candidate reranker with the PyTorch[6] v1.7.0 APIs in Huggingface Transformers v2.11.0 (Wolf et al., 2020), using either `bert-base-uncased` or `bert-multilingual-uncased`. We train with the Adam optimizer, a learning rate of 1e-5, a maximum sequence length of 128 tokens, and a number of epochs of 30. We explored a small number of learning rates (1e-5, 1e-6, 5e-6) and epoch numbers (10, 20, 30, 40). When training without context, we use one Tesla V100 GPU with 32GB memory and a batch size of 8. When training with context, we use four Tesla V100 GPU with 32GB memory and a batch size of 32. The total number of parameters in our model is 168M and the training time is about 3 hours.

4.5 Systems

We compare a variety of geocoding systems:

SSPART is our proposed Search, Sort by Population, And Rerank by Transformer architecture.

[3] http://www.geonames.org/data-sources.html

[4] https://lucene.apache.org/

[5] https://en.wikipedia.org/wiki/List_of_adjectival_and_demonymic_forms_for_countries_and_nations

[6] https://pytorch.org/

We consider several variants of SSPART, comparing English vs. multilingual versions of BERT, and comparing no context sentences vs. with $c \in \{0, 1, 2\}$ (i.e., just the sentence containing m up to the 5 sentences around m).

SSP is the Search-and-Sort-by-Population candidate generator of SPART, without the transformer-based candidate reranker.

CamCoder Gritta et al. (2018) introduced an approach, CamCoder, where the model predicts one of 7823 classes, each a 2x2 degree tile representing part of the world's surface. CamCoder's lexical input combines context words, location mentions, and the target mention, encoding them with convolutional neural network layers. CamCoder's geographic input is a map vector that encodes a population distribution over the 7823 map tiles based on the location mentions in the target mention's context[7]. The two kinds of inputs are concatenated and fed into dense layers and a final softmax layer to make the prediction. CamCoder is the state-of-the-art on several geocoding datasets (including LGL), and its code is publicly available[8].

Edinburgh Grover et al. (2010) introduced a rule-based system that uses heuristics such as population count, clustering (spatial minimization), type and country and some contextual information (containment, proximity, locality, clustering) to score, rank, and choose a candidate. While this is an older system, it was still state-of-the-art on LGL before CamCoder, and as we will see below, it still outperforms recent techniques on several datasets.

Mordecai Halterman (2017) introduced an approach, Mordecai, that uses Elasticsearch to generate candidates and neural networks based on word2vec (Mikolov et al., 2013) to rank those candidates. Its models are trained on proprietary data.

DeezyMatch Hosseini et al. (2020) introduced an approach, DeezyMatch, for fuzzy string matching and candidate ranking. DeezyMatch uses only the mention and entry names; it does not use context or other database information. The approach first pretrains an LSTM-based classifier on the database taking string pairs as input, and then fine-tunes the pair classifier on the target dataset. The trained DeezyMatch model is then used to generate vector representations for all known variations of entity names in the database. Given a mention, the same DeezyMatch model is used to generate its vector representation, and entries are ranked by comparing the mention vector to all entry vectors using L2-norm distance or cosine similarity.

SAPBERT Liu et al. (2021) introduced a pre-train-then-finetune approach, SAPBERT, for biomedical entity linking. SAPBERT uses only the mention and entry names; it does not use context or other database information. The approach pretrains a transformer-based language model on the database using a self-alignment metric learning objective and an online hard pairs mining mechanism to cluster synonyms of the same concept together and move different concepts further away. The pretrained SAPBERT is then fine-tuned on the target dataset. SAPBERT was trained for the biomedical domain, but is easily retrained for other domains. We pretrain SAPBERT on the geospatial database and finetune it on the toponym resolution datasets.

Dataset	Models	test	
		R@1	R@20
LGL	DeezyMatch	0.172	0.538
	SAPBERT	0.245	0.742
	SSP	0.606	0.962
GeoWebNews	DeezyMatch	0.262	0.671
	SAPBERT	0.428	0.746
	SSP	0.694	0.866
TR-News	DeezyMatch	0.206	0.702
	SAPBERT	0.355	0.780
	SSP	0.716	0.965

Table 2: Performance of candidate generators on the test sets. R@1 is useful for measuring the performance (Accuracy) of the candidate generator when used directly as a geocoder. R@20 is useful for estimating the ceiling performance of a top-20 reranker based on that candidate generator.

5 Results

We first evaluate our Lucene-based candidate generator, SSP. We compare it with the recent context-free candidate generators, SAPBERT and DeezyMatch. Table 2 shows that SSP outperforms both

[7]The original CamCoder code, when querying GeoNames to construct its input population vector from location mentions in the context, assumes it has been given canonical names for those locations. Since canonical names are not known before locations have been resolved to entries in the ontology, we have CamCoder use mention strings instead of canonical names for querying GeoNames.

[8]We also wanted to compare against MLG (Kulkarni et al., 2021), which slightly outperforms CamCoder on two datasets (while being slightly worse on LGL), but this was impossible as neither its code nor its data splits are available.

Model				LGL (test)				GeoWebNews (test)				TR-News (test)			
Name	Multilingual?	Context?	Combine 3 sets?	Accuracy	Accuracy@161km	Mean Error	AUC	Accuracy	Accuracy@161km	Mean Error	AUC	Accuracy	Accuracy@161km	Mean Error	AUC
Edinburgh				.611	.632	119	.290	.738	.773	146	.203	.750	.756	149	.218
CamCoder				.580	.651	82	.288	.572	.665	155	.290	.660	.778	**89**	.196
Mordecai				.322	.375	926	.594	.291	.333	1072	.633	.472	.553	6558	.427
DeezyMatch				.172	.182	654	.704	.262	.323	537	.601	.206	.220	741	.705
SAPBERT				.245	.260	566	.630	.428	.499	357	.446	.355	.362	595	.568
SSPART	✓	1		**.759**	**.783**	**67**	**.166**	–	–	–	–	–	–	–	–
SSPART	✓	0		–	–	–	–	.782	.832	60	.131	–	–	–	–
SSPART	✓	1		–	–	–	–	–	–	–	–	.777	.798	92	.166
SSPART	✓	1	✓	.740	.771	73	.182	**.818**	**.855**	**50**	**.113**	**.805**	**.812**	97	**.158**

Table 3: Performance on the test sets. Higher is better for Accuracy and Accuracy@161km. Lower is better for Mean Error and AUC. Edinburgh made no predictions for 28, 49, and 106 toponyms in LGL, GeoWebNews, and TR-News, respectively. It is impossible to calculate distance-based metrics without coordinates, so we skip those toponyms. As a result, distance-based metrics are overestimated for Edinburgh while the Accuracy metric is not.

of these approaches by large margins, both in accuracy of the top entry (R@1) and whether the correct entry is in the top 20 (R@20).

We next evaluate our complete generate-and-rank system, SSPART, against other geocoders. We first performed model selection on the development set as described in appendix A, selecting the best models for LGL (multilingual transformer and three sentence context $c = 1$), GeoWebNews (multilingual transformer and one sentence context $c = 0$), and TR-News (multilingual transformer and three sentence context $c = 1$). We also trained a model where we combined the LGL, GeoWebNews, and TR-News training sets, and used the setting that achieved the best average rank across all metrics and development datasets (multilingual transformer and three sentence context $c = 1$). The first three models represent systems tuned to specific datasets, while the last model represents a system trained for more general use.

Table 3 shows that our proposed model, SSPART, outperforms Edinburgh, CamCoder, Mordecai, DeezyMatch, and SAPBERT across all three public toponym resolution test sets on all metrics except for mean error on TR-News[9]. The more general SSPART model trained on the combined LGL, GeoWebNews, and TR-News outperforms the dataset-specific models on GeoWebNews and TR-News, though not on LGL. We release the general model for English geocoding under the Apache License v2.0, for off-the-shelf use at `https://<anonymized>`.

[9]The Friedman test for comparing system average ranks is only reliable when comparing more than 6 systems, so we do not report p-values here.

6 Qualitative Analysis

We qualitatively analyzed some of the errors that CamCoder and different variants of SSPART made. Example 1 from table 4 shows an example where CamCoder fails but SSPART succeeds, by more effectively using geospatial metadata, such as the population and an alternate name for *District of Columbia* in GeoNames, *Washington, D.C.*. Example 2 from table 4 shows an example where Lucene search fails but SSPART without context succeeds, by not relying too heavily on population alone and instead jointly considering the name, population, and feature code information (ADM2 represents a county, PPLA2 represents a city). Example 3 from table 4 shows an example where SSPART without context fails but SSPART with context succeeds, by taking advantage of the *Minnesota* in the context to select the *Clay County* that would otherwise seem implausible due to its lower population.

Finally, example 4 from table 4 shows an example where our best SSPART model still fails. Though the candidate generator finds the correct ontology entry in its top-k list, the candidate reranker is unable to sort that entry to the top. Neither the name, population, nor feature code would suggest the correct candidate, and the nearby context is also insufficient to disambiguate. Looking at the entire document, many of the other toponym mentions

	Example	Candidate				Rank			
		Name	Population	Feature Code	State	CamCoder	SSP	SSPART c = -	SSPART
1	*Ahlstrom said that the results found at the Washington Latin Public Charter School in Washington, D.C.*	**District of Columbia**	552433						1
		Washington County	147430			1			
2	*It was Los Angeles police officers she attempted to blow up.*	Los Angeles County	9818605	ADM2			1	2	
		Los Angeles	3971883	PPLA2			2	1	
		Los Angeles	125430	PPLA2			3	3	
		Los Angeles	4217	PPL			4	4	
3	*the Minnesota State Patrol urges motorists to drive with caution as flooding continues to affect area highways. Water over the roadway is currently affecting the following areas in Becker, Clay, and Douglas*	Clay County	221939		Missouri			1	4
		Clay County	190865		Florida			2	3
		Clay County	58999		Minnesota			3	1
		Clay County	26890		Indiana			4	2
4	*he writes, as do my efforts to insure New London is a safe community.*	New London County	274055	ADM2			1		3
		New London	27179	PPL			2		1
		New London	7172	PPL			3		2
		New London	1882	PPL			4		4

Table 4: Examples of predictions from CamCoder, our candidate generator with no reranking (SSP), our generate-and-rerank system without context (SSPART c = -), and our full system including context (SSPART). Target location mentions are underlined. Human annotated ontology entries are in bold.

are located in the same state, but even if we expanded SSPART's context window, it would not be able to tell they were in the same state without first resolving them to ontology entries.

7 Limitations and future research

SSPART's context window is currently limited to five sentences, and thus cannot take advantage of document-level signals. In the future, we will explore integrating document-level features into SSPART, such as spatial minimality (Grover et al., 2010) which assumes that place names in a text tend to refer to geospatial regions that are in close spatial proximity to each other. This might be achieved by jointly reasoning over the candidate entries of all location mentions in a document, or by a strategy of resolving easy location mentions before hard ones.

SSPART's candidate generator is currently based on information retrieval. This is efficient but not very flexible in string matching, and when the candidate generator fails to produce the correct candidate entry, the candidate reranker also necessarily fails. In the future, we will explore whether it is possible to replace this generator with a neural network candidate generator to provide more robust string matching while still retraining reasonable efficiency. We are also interested in architectures for combining the candidate generator and candidate reranker into a single model, which would avoid the problems of a candidate generator that fails.

SSPART is limited by its training and evaluation data, which covers only thousands of English toponyms from news articles, while there are many millions of toponyms across the world. It is likely that there are regional differences in SSPART's accuracy that will need to be addressed by future research.

8 Conclusion

We propose a new toponym resolution architecture – Search, Sort by Population, And Rerank by Transformer (SSPART) – that combines the tight ontology integration of generate-and-rerank systems with the robust text encoding of deep neural networks. SSPART consists of an information retrieval-based candidate generator and a BERT-based reranker that incorporates features important to toponym resolution such as population, type of location, and textual context around the toponym. We evaluate our proposed architecture against prior state-of-the-art, using multiple evaluation metrics and multiple datasets. SSPART achieves new state-of-the-art performance on all datasets.

References

Benjamin Adams and Grant McKenzie. 2018. Crowdsourcing the character of a place: Character-level convolutional networks for multilingual geographic text classification. *Transactions in GIS*, 22(2):394–408.

Dirk Ahlers. 2013. Assessment of the accuracy of geonames gazetteer data. In *Proceedings of the 7th workshop on geographic information retrieval*, pages 74–81.

Mariona Coll Ardanuy, Kasra Hosseini, Katherine McDonough, Amrey Krause, Daniel van Strien, and Federico Nanni. 2020. A deep learning approach to geographical candidate selection through toponym matching. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 385–388.

Zahra Ashktorab, Christopher Brown, Manojit Nandi, and Aron Culotta. 2014. Tweedr: Mining twitter to inform disaster response. In *ISCRAM*, pages 269–272.

Berico Technologies. 2012. Cartographic location and vicinity indexer (clavin).

Preeti Bhargava, Nemanja Spasojevic, and Guoning Hu. 2017. Lithium NLP: A system for rich information extraction from noisy user generated text on social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 131–139, Copenhagen, Denmark. Association for Computational Linguistics.

Ana Bárbara Cardoso, Bruno Martins, and Jacinto Estima. 2019. Using recurrent neural networks for toponym resolution in text. In *EPIA Conference on Artificial Intelligence*, pages 769–780. Springer.

Jens A de Bruijn, Hans de Moel, Brenden Jongman, Jurjen Wagemaker, and Jeroen CJH Aerts. 2018. Taggs: grouping tweets to improve global geoparsing for disaster response. *Journal of Geovisualization and Spatial Analysis*, 2(1):2.

Grant DeLozier, Jason Baldridge, and Loretta London. 2015. Gazetteer-independent toponym resolution using geographic word profiles. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2382–2388. AAAI Press.

Nuno Freire, José Borbinha, Pável Calado, and Bruno Martins. 2011. A metadata geoparsing system for place name recognition and resolution in metadata records. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 339–348.

Fredric Gey, Ray Larson, Mark Sanderson, Hideo Joho, Paul Clough, and Vivien Petras. 2005. Geoclef: the clef 2005 cross-language geographic information retrieval track overview. In *Workshop of the cross-language evaluation forum for european languages*, pages 908–919. Springer.

Milan Gritta, Mohammad Taher Pilehvar, and Nigel Collier. 2018. Which Melbourne? augmenting geocoding with maps. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1285–1296, Melbourne, Australia. Association for Computational Linguistics.

Milan Gritta, Mohammad Taher Pilehvar, and Nigel Collier. 2019. A pragmatic guide to geoparsing evaluation. *Language Resources and Evaluation*, pages 1–30.

Claire Grover, Richard Tobin, Kate Byrne, Matthew Woollard, James Reid, Stuart Dunn, and Julian Ball. 2010. Use of the edinburgh geoparser for georeferencing digitized historical collections. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1925):3875–3889.

Andrew Halterman. 2017. Mordecai: Full text geoparsing and event geocoding. *The Journal of Open Source Software*, 2(9).

Simon I Hay, Katherine E Battle, David M Pigott, David L Smith, Catherine L Moyes, Samir Bhatt, John S Brownstein, Nigel Collier, Monica F Myers, Dylan B George, et al. 2013. Global mapping of infectious disease. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 368(1614):20120250.

Thi Bich Ngoc Hoang and Josiane Mothe. 2018. Location extraction from tweets. *Information Processing & Management*, 54(2):129–144.

Kasra Hosseini, Federico Nanni, and Mariona Coll Ardanuy. 2020. DeezyMatch: A flexible deep learning approach to fuzzy string matching. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 62–69, Online. Association for Computational Linguistics.

Zongcheng Ji, Qiang Wei, and Hua Xu. 2020. Bert-based ranking for biomedical entity normalization. *AMIA Summits on Translational Science Proceedings*, 2020:269.

Ehsan Kamalloo and Davood Rafiei. 2018. A coherent unsupervised model for toponym resolution. In *Proceedings of the 2018 World Wide Web Conference*, pages 1287–1296.

Morteza Karimzadeh, Wenyi Huang, Siddhartha Banerjee, Jan Oliver Wallgrün, Frank Hardisty, Scott Pezanowski, Prasenjit Mitra, and Alan M MacEachren. 2013. Geotxt: a web api to leverage place references in text. In *Proceedings of the 7th workshop on geographic information retrieval*, pages 72–73.

Sayali Kulkarni, Shailee Jain, Mohammad Javad Hosseini, Jason Baldridge, Eugene Ie, and Li Zhang. 2021. Multi-level gazetteer-free geocoding. In

Proceedings of Second International Combined Workshop on Spatial Language Understanding and Grounded Communication for Robotics, pages 79–88, Online. Association for Computational Linguistics.

Abhinav Kumar and Jyoti Prakash Singh. 2019. Location reference identification from tweets during emergencies: A deep learning approach. *International journal of disaster risk reduction*, 33:365–375.

Egoitz Laparra and Steven Bethard. 2020. A dataset and evaluation framework for complex geographical description parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 936–948, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sunshin Lee, Mohamed Farag, Tarek Kanan, and Edward A Fox. 2015. Read between the lines: A machine learning approach for disambiguating the geo-location of tweets. In *Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 273–274.

Michael D Lieberman and Hanan Samet. 2011. Multifaceted toponym recognition for streaming news. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 843–852.

Michael D Lieberman and Hanan Samet. 2012. Adaptive context features for toponym resolution in streaming news. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 731–740.

Michael D Lieberman, Hanan Samet, and Jagan Sankaranarayanan. 2010. Geotagging with local lexicons to build indexes for textually-specified spatial data. In *2010 IEEE 26th international conference on data engineering (ICDE 2010)*, pages 201–212. IEEE.

Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2021. Self-alignment pretraining for biomedical entity representations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4228–4238.

Xiangyang Luo, Yaqiong Qiao, Chenliang Li, Jiangtao Ma, and Yimin Liu. 2020. An overview of microblog user geolocation methods. *Information Processing & Management*, 57(6):102375.

Bruno Martins, Ivo Anastácio, and Pável Calado. 2010. A machine learning approach for resolving place references in text. In *Geospatial thinking*, pages 221–236. Springer.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Afshin Rahimi, Timothy Baldwin, and Trevor Cohn. 2017. Continuous representation of location for geolocation and lexical dialectology using mixture density networks. *arXiv preprint arXiv:1708.04358*.

Afshin Rahimi, Duy Vu, Trevor Cohn, and Timothy Baldwin. 2015. Exploiting text and network context for geolocation of social media users. *arXiv preprint arXiv:1506.04803*.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1500–1510.

Michael Speriosu and Jason Baldridge. 2013. Text-driven toponym resolution using indirect supervision. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1476.

Laura Tateosian, Rachael Guenter, Yi-Peng Yang, and Jean Ristaino. 2017. Tracking 19th century late blight from archival documents using text analytics and geoparsing. In *Free and open source software for geospatial (FOSS4G) conference proceedings*, volume 17, page 17.

Richard Tobin, Claire Grover, Kate Byrne, James Reid, and Jo Walsh. 2010. Evaluation of georeferencing. In *proceedings of the 6th workshop on geographic information retrieval*, pages 1–8.

Xiaobin Wang, Chunping Ma, Huafei Zheng, Chu Liu, Pengjun Xie, Linlin Li, and Luo Si. 2019. Dm_nlp at semeval-2018 task 12: A pipeline system for toponym resolution. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 917–923.

Davy Weissenbacher, Arjun Magge, Karen O'Connor, Matthew Scotch, and Graciela Gonzalez-Hernandez. 2019. SemEval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 907–916, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Dongfang Xu, Zeyu Zhang, and Steven Bethard. 2020. A generate-and-rank framework with semantic type regularization for biomedical concept normalization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8452–8464, Online. Association for Computational Linguistics.

Ikuya Yamada, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto. 2022. Global entity disambiguation with BERT. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3264–3271, Seattle, United States. Association for Computational Linguistics.

Wei Zhang and Judith Gelernter. 2014. Geocoding location expressions in twitter messages: A preference learning method. *Journal of Spatial Information Science*, 2014(9):37–70.

A Model selection

We performed model selection on the development sets, comparing the SSPART model variants described in section 4.5. Results are shown in table 5. Because we have many different evaluation metrics that do not always tell exactly the same story, we include an average rank metric that calculates a ranking of all SSPART models according to that metric, and scores each SSPART model as the average over its ranks across all metrics. So, for example, the best model, SSPART with a multilingual transformer and context of the three sentences around the mention ($c = 1$), achieves average rank of 2.0 on LGL because it was rank 3 for Accuracy, rank 1 for Accuracy@161km, rank 3 for Mean Error, and rank 1 for AUC.

All of our SSPART models outperform both CamCoder and Edinburgh on all development sets and across all metrics. The average rank of SSPART is consistently above CamCoder and Edinburgh (p values between 0.0000 and 0.0066), according to the Friedman test with the Conover post hoc test for pairwise comparisons. All SSPART models also outperform the candidate generator alone (SSP), with the candidate reranker substantially improving performance (p values between 0.0000 and 0.0024). For monolingual SSPART models, adding context slightly worsened their average rank, while for multilingual models, adding context slightly improved their average rank. The impact of population and feature code were small; see appendix D for details. Overall, we conclude that the main driver of increased performance is the transformer-based reranker.

B Artifact intended use and coverage

The intended use of `bert-base-uncased` and `bert-multilingual-uncased` is to be “fine-tuned on tasks that use the whole sentence”[10]. We have used them for that purpose when encoding the context, but also for the related task of encoding place names, which are usually short phrases. These artifacts are trained on English books and English Wikipedia and released under an Apache 2.0 license which is compatible with our use.

The intended use of our geocoding model is matching English place names in text to the GeoNames ontology. Though GeoNames covers millions of place names, our evaluation corpora cover only English news articles, and thus the performance we report is only predictive of performance in that domain.

C Difference between Toponym resolution and Wikipedia linking

Wikipedia includes in-text examples for all its concepts, while GeoNames is an ontology only; it has no in-text examples for its concepts. The only in-text examples come from small-scale training corpora like LGL, GeoWebNews, or TR-News, which, as shown in table 1, include only a tiny fraction of GeoNames’s 7 million geographical concepts. As a result, many approaches to Wikipedia linking are difficult to apply to geocoding. For example, Yamada et al. (2022)’s approach of jointly training token and concept embeddings assumes there are text examples of all concepts.

D Feature ablation

Table 6 shows performance when the feature code and population features are removed from the SSPART model. The features help slightly on LGL and GeoWebNews and hurt slightly on TR-News.

[10]`https://huggingface.co/bert-base-uncased`

Model			LGL (dev)					GeoWebNews (dev)					TR-News (dev)					All (dev)
Name	Multilingual?	Context?	Accuracy	Accuracy@161km	Mean Error	AUC	Average Rank	Accuracy	Accuracy@161km	Mean Error	AUC	Average Rank	Accuracy	Accuracy@161km	Mean Error	AUC	Average Rank	Average Rank
Edinburgh			.666	.676	147	.260		.687	.721	174	.250		.662	.676	183	.273		
CamCoder			.604	.695	144	.285		.509	.712	174	.293		.824	.882	128	.119		
Mordecai			.303	.341	999	.621		.356	.480	943	.528		.530	.559	605	.419		
DeezyMatch			.239	.279	531	.626		.260	.299	480	.617		.250	.250	1005	.719		
SAPBERT			.279	.327	474	.584		.434	.480	375	.463		.324	.324	917	.649		
SSP			.594	.671	201	.289		.644	.858	73	.165		.677	.735	187	.242		
SSPART		-	.797	.821	57	.140	4.8	**.886**	**.940**	29.8	.060	**1.8**	.882	.897	63.5	.090	2.8	3.0
SSPART		0	.807	.823	**55**	.132	3.0	.865	.915	39.3	.075	5.5	.882	.882	110	.109	5.5	4.7
SSPART		1	.807	.816	65	.142	6.0	.868	.918	40.3	.073	4.8	.882	.897	64.9	.092	4.0	4.9
SSPART		2	.802	.814	68	.145	7.0	.865	.911	42.8	.078	6.5	.897	.912	64.0	.081	2.3	5.3
SSPART	✓	-	.814	.828	60	.132	2.8	.879	.922	43.2	.072	4.0	.882	.897	65.0	.092	4.3	3.7
SSPART	✓	0	**.816**	.831	62	.133	3.3	.872	**.940**	**23.5**	**.057**	**1.8**	.882	.897	64.6	.090	3.5	2.8
SSPART	✓	1	.809	**.833**	59	**.129**	**2.0**	.875	.922	35.4	.073	3.8	**.912**	**.927**	**40.6**	**.063**	**1.0**	**2.3**
SSPART	✓	2	.807	.823	61	.137	4.5	.872	**.940**	29.4	.060	2.5	.868	.882	72.6	.103	5.3	4.1

Table 5: Performance on the development sets. Higher is better for Accuracy and Accuracy@161km. Lower is better for Mean Error and AUC. Edinburgh made no predictions for 24, 16, and 0 toponyms in LGL, GeoWebNews, and TR-News, respectively. Since it is impossible to calculate distance-based metrics without coordinates, we skip those toponyms, and as a result overestimate the distance-based metrics for Edinburgh.

Model			LGL (dev)				GeoWebNews (dev)				TR-News (dev)			
Name	Multilingual?	Context?	Accuracy	Accuracy@161km	Mean Error	AUC	Accuracy	Accuracy@161km	Mean Error	AUC	Accuracy	Accuracy@161km	Mean Error	AUC
SSPART -fc -pop			.802	.819	64	.141	.865	.925	39.5	.072	.897	.912	64.0	.081
SSPART -fc			.792	.819	68	.141	.861	.918	34.7	.072	.868	.882	65.7	.100
SSPART -pop			.807	.828	61	.134	.865	.915	31.9	.073	.897	.912	42.7	.074
SSPART			.809	.831	56	.133	.886	.932	29.9	.062	.882	.897	86.0	.096

Table 6: Performance on the development sets when ablating the feature code (-fc) and population (-pop) features. Higher is better for Accuracy and Accuracy@161km. Lower is better for Mean Error and AUC.