SoftwareX 23 (2023) 101480

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

SNEToolkit: Spatial named entities disambiguation toolkit

Rodrique Kafando [a,b], Rémy Decoupes [a], Mathieu Roche [a,c], Maguelonne Teisseire [a,*]

[a] TETIS, Univ Montpellier, AgroParisTech, CIRAD, CNRS, INRAE, 500 rue Jean François Breton, Montpellier, 34090, France
[b] CITADEL, Univ. Virtuelle BF, Ouagadougou, Burkina Faso
[c] CIRAD, F-34398 Montpellier, France

ARTICLE INFO

Article history:
Received 13 January 2023
Received in revised form 13 July 2023
Accepted 17 July 2023

Keywords:
Spatial named entity
Disambiguation
Geocoding
Software

ABSTRACT

"Can you tell me where San Jose is located?" "Uh! Do you know that there are more than 1700 locations named San Jose in the world?" The official name of a location is often not the name with which we are familiar. Spatial named entity (SNE) disambiguation is the process of identifying and assigning precise coordinates to a place name that can be identified in a text. This task is not always straightforward, especially when the place name in question is ambiguous for various reasons. In this context, we are interested in the disambiguation of spatial named entities that can be identified in a textual document on a country level. The solution that we propose is based on a set of techniques that allow us to disambiguate the spatial entity considering the context in which it is mentioned from a certain number of characteristics that are specific to it. The solution uses as input a textual document and extricates the named entities identified therein while associating them with the correct coordinates. SNE disambiguation is designed to support the process of fast exploration of spatiotemporal data analysis, most often for event tracking. The proposed approach was tested on 1360 SNEs extracted from the GeoVirus dataset. The results show that SNEToolkit outperformed the baseline, the standard Geonames geocoder, with a recall value of 0.911 against a recall value of 0.871 for the baseline. A flexible Python package is provided for end users.

Code metadata

Current code version	v0.1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00004
Legal Code License	BSD-2-Clause license
Code versioning system used	Git
Software code languages, tools, and services used	Python 3.7
Compilation requirements, operating environments & dependencies	geocoder(1.38.1), nltk(3.5), numpy(1.21.5), pandas(1.3.5), similarity(0.0.1), spaCy(3.2.3), strsim(0.0.3), titlecase(2.3), matplotlib(3.1.0)
If available Link to developer documentation/manual	
Support email for questions	maguelonne.teisseire@inrae.fr

1. Motivation and significance

In natural language processing (NLP), the notion of space is an important component in analyzing textual data that describe or report spatiotemporal events. Commonly identified and called spatial named entities (SNEs) correspond to the place names that are mentioned in a document. We can distinguish two types of spatial named entities: the first type comprises absolute named entities or those that can be identified by longitude and latitude coordinates, such as city names (e.g., Montpellier, France, etc.) and the second type consist of relative named entities that are generally defined by complements of indication, direction, etc. (e.g., north of Paris, south of France) [1]. We only focus on absolute spatial named entities in this paper.

In analyzing a document, consideration of the place names can be challenging since the names can contain several nuances, rendering them ambiguous. We distinguish three common cases

* Corresponding author.
E-mail addresses: rodrique.kafando@inrae.fr (Rodrique Kafando), remy.decoupes@inrae.fr (Rémy Decoupes), mathieu.roche@cirad.fr (Mathieu Roche), maguelonne.teisseire@inrae.fr (Maguelonne Teisseire).

https://doi.org/10.1016/j.softx.2023.101480

of ambiguity:

- Case No. 1[1]: a spatially named entity can be shared by different places (e.g., Montpellier from France and Montpellier from Canada). The perfect example is San Jose, which matches 1716 place names in the world.[2]
- Case No. 2[3]: a spatially named entity can designate both a place and a nonplace or generally refers to places named after people (e.g., Washington);
- Case No. 3[4]: the same place can be designated by several names or several appellations (e.g., United Kingdom and Great Britain, Paris and Panama).

In this paper, we present a Python module named ***SNEToolkit*** that assists in overcoming several forms of ambiguity related to spatial named entities mentioned in textual documents.

In the literature, we can distinguish four main approaches for spatial named entity disambiguation. (i) Approaches based on rules and criteria, such as the Geonames[5] database, for native disambiguation that uses geographic level and population size. In [2], the authors include a similarity measure of the basics rules of Geonames by using the Levenshtein distance measure to compute the similarity between the extracted place name in the text and the others proposed by the Geonames database. The authors in [3] also applied fuzzy logic techniques to resolve spatial ambiguities. Some approaches, such as [4], use knowledge graphs or gazetteers [5] to disambiguate named entities. (ii) The second approach uses Machine Learning methods [6] to disambiguate spatial named entities within the textual data. (iii) We also note Deep Learning approaches such as [7] that attempt to disambiguate spatial named entities. Spatial named entities is applied and used in many fields. Several machine learning techniques are listed in this review paper [8].

In [9], authors designed NewsStand system, which is focused on news collection and representation. The system retrieves, analyses, and visualizes news articles on a map interface, extracting the inherent geographic content using a custom-built geotagger. This approach allows users to access news stories based on topical relevance and geographic region, making implicit geographic content more accessible and understandable to readers. A different perspective is offered by a text-driven approach [10] that leverages document-level geotags to indirectly create training instances for text classifiers used in toponym resolution. This method, effective even in historical contexts, integrates textual cues with commonly used toponym resolution techniques. It highlights the importance of considering distance measurements in toponym resolution, offering insights into the shaping of historical empires and the geographical content in news articles and microblogs. Geographical Information Retrieval (GIR) research also plays a crucial role in toponym resolution. New strategies for determining a document's geographical coverage or scope are proposed in [11], complemented by novel techniques for query expansion and relevance ranking in GIR. This approach emphasizes the significance of geographical scope in GIR tasks and paves the way for further research in related areas. An interesting turn is taken in the development of a gazetteer-independent system [12] that grounds toponyms using overlaps in the distributions of toponyms and other contextual words in local clusters. It proves that such a system can function effectively without the aid of gazetteer resources and extensive metadata, demonstrating superior performance over gazetteer-bound methods on toponyms found by a named-entity recognizer. In [13], authors designed an approach for toponym disambiguation. It involves using a corpus-based method that utilizes unsupervised machine learning to create disambiguation rules and tag toponyms with information from publicly available gazetteers. This method shows commendable accuracy in disambiguating toponyms in news articles. An other method uses text-based geolocation [14] prediction for Twitter users. It examines the impact of nongeotagged tweets, language, user-declared metadata, and temporal variance on geolocation prediction, offering insights into user variability in terms of geolocatability. This approach provides an in-depth exploration of text-based geolocation prediction systems, delivering robust and practical solutions. In [15] authors discuss various toponym resolution techniques, including entity linking and toponym resolution using traditional machine learning algorithms based on feature engineering and deep learning algorithms. The paper proposes a spatial clustering-based voting approach that combines several individual approaches to improve the robustness and generalizability of the techniques. The paper compares the proposed approach with 20 latest and commonly-used approaches based on 12 public datasets, including several highly challenging datasets. The detailed evaluation results can inform future methodological developments and guide the selection of proper approaches based on application needs.

We note that spatial named entity recognition and disambiguation has been notably applied in various domains, including the analysis of news articles, research papers, as well as in the geolocation of brief messages sourced from social media platforms like Tweeter. The nature of these applications, however, is considerably distinct, each posing unique context disambiguation challenges. In the case of news articles, there is not typically an implicit assumption that mentioned spatial named entities are in close proximity to each other. Conversely, social media texts, due to their brevity and personal context, often contain only a single named place, necessitating a different approach for effective resolution. Nonetheless, it is important to acknowledge that hybrid methodologies, akin to the one put forth in Ref. [15], have been found to be capable of delivering superior performance across diverse types of documents. Such an approach consolidates the strengths of various techniques, thereby enhancing the effectiveness of spatial named entity recognition and resolution irrespective of the document context.

Among these strategies, we designed and added SNEToolkit, which is based on (i) rules that combine fuzzy matching, (ii) alias resolving and (iii) popular country code (based on context) approaches with gazetteers that address ambiguous spatial named entities at the country level. All these sets of rules are ranked based on population size. SNEToolkit offers two access modes: a Python package for advanced users and a graphical interface based on Streamlit for all users. The baseline of this study is the Geonames standard geocoder on which we have implemented SNEToolkit, adding the disambiguation approaches as our main contribution. SNEToolkit is applied as a postprocessing step to the output of the Geonames geocoder. SNEToolkit is primarily implemented for on WikiNews data, which does not guarantee its effectiveness with other types of data, particularly social network data such as tweets and other short messages. Its performance may also be uncertain in instances where spatial named entities are employed metaphorically.

2. Software description

This section provides details on the architectural composition and the different functionalities offered by SNEToolkit.

1 https://en.wikipedia.org/wiki/List_of_homonymous_states_and_regions
2 https://en.wikipedia.org/wiki/List_of_popular_place_name
3 https://en.wikipedia.org/wiki/List_of_places_named_after_people
4 https://en.wikipedia.org/wiki/List_of_double_placenames
5 https://www.geonames.org/

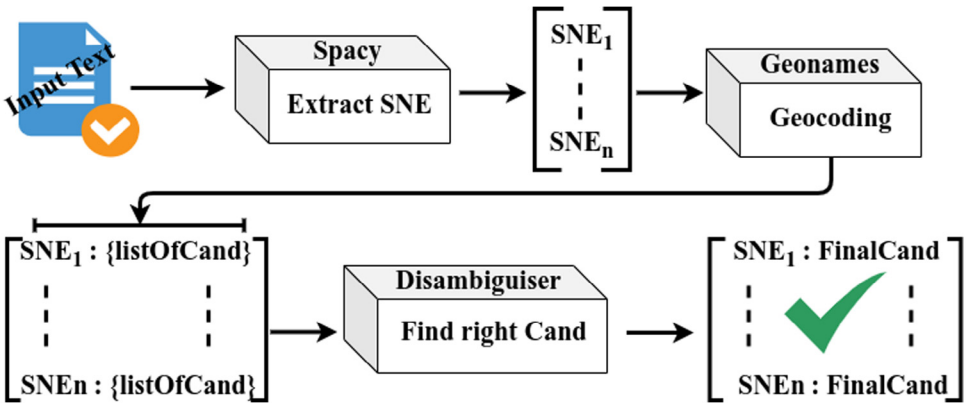

Fig. 1. Disambiguation Pipeline.

2.1. Architecture

SNEToolkit is written in Python 3. The SNE extraction process is supported by **spaCy** [16], and the **geocoding** step is supported by the Geonames API. Most techniques also depend on NumPy. All the output is standardized so that it is compatible with data processing packages such as Pandas.

The frontend is implemented using Streamlit to establish the interface to the backend. Users have the flexibility to use their desired step, including SNE extraction, SNE geocoding, and SNE disambiguation. A tutorial with an illustrative example that uses the user interface is available on the GitHub repository.

Fig. 1 summarizes the SNE disambiguator processing steps. Textual data are passed as input for SNE extraction with spaCy. The set of SNEs is geocoded with Geonames, where each SNE has a set of candidates (listOfCand). Each SNE is disambiguated based on its set of candidates. In the final output, only the candidate that is identified as correct is retained.

In the next section, we define each step of the entire pipeline.

2.2. Functionalities

While analyzing spatiotemporal data, spatial named entity disambiguation assists us in being more precise and concise and helps us provide an accurate analysis. In the disambiguation process, we highlighted some strategies to address this quest for accuracy. Each step of the pipeline is described in the following subsections.

2.2.1. Spatial named entities extraction

Spatial named entities extraction is a part of the named entities recognition (NER) task, which consists of identifying key information in the text and its classification into a set of predefined categories. Several tools are available to perform this task [16–18]. In this study, we chose to use spaCy, one of the best optimizing tools for NER extraction [19]. As mentioned above, there are two types of spatial named features, namely, absolute spatial entities and relative spatial named entities. The process, as implemented, is only concerned with the disambiguation of absolute spatial named entities.

This functionality uses raw textual data (e.g., documents) as input to extract all the spatial named entities within it. spaCy allows the extraction of multilevel spatial named entities, such as GPE (geo-political entities) for countries, cities, states, etc., and LOC is used for non-GPE locations such as mountain ranges and bodies of water.

2.2.2. Spatial named entity geocoding

This step consists of associating the geographic coordinates with each spatial named entity. We use Geonames,[6] a geographical database that contains a large volume of places and location names associated with their coordinates. The Geonames database is enriched and populated by external databases and sources.[7]

For a given spatial named entity, Geonames returns a set of candidates with the associated coordinates that may suit the input. The native Geonames API incorporates a static disambiguation approach (as it does not take semantic context into account), and unfortunately, it cannot resolve context issues. Without additional attributes such as feature class or country code, Geonames will always return the candidate that has the largest number of inhabitants or a large area by default. In complex situations, we need to add more techniques and criteria that consider the context in which the spatial named entity is mentioned. To address the limitations found in the default geocoding of the Geonames database, our approach aims to refine the extracted Spatial Named Entity (SNE) from the output provide by Geonames. This refinement is not integrated within Geonames but implemented as a post-processing measure where we introduce additional selection criteria. Specifically, the context is carefully considered with a scoring strategy which takes into account nonambiguous SNEs as well as population size. We also utilize a fuzzy matching technique to effectively correct any syntactic or spelling inaccuracies that may have emerged in the initial Geonames results.

In the following subsection, we describe the entire process utilized to select the final candidate.

2.2.3. Spatial named entities disambiguation

- `splitAmbiguousAndNonAmnbiguous 1`: This first step consists of splitting the set of geocoded SNEs with corresponding candidates into two types: *ambiguous* and - *non_ambiguous*.
  We assume that by querying the geographic databases in the geocoding step, we have the possibility of returning one or more candidates that can be completely associated with different countries. Thus, the same place name can be associated with several unique locations, where the same name is identified by several coordinates (Case No. 2) and the same name can have several names (Case No. 3). Generally, Case No. 2 is resolved by spaCy but has weaknesses in some cases [20].
  We assume that a spatial named entity is nonambiguous only if

[6] https://www.geonames.org/
[7] https://www.geonames.org/datasources/

- the SNE has one and only one candidate in the geographic database,
- all the candidates have the same name (spelling) as the input SNE and if and only if all the candidates are from the same country (identified by the country code). The country is associated with its country code.

We assume that a spatial named entity is ambiguous only if

- the SNE has more than one candidate from different countries in the geographic database.

We can distinguish two categories of ambiguous SNE, ambiguous named entities whose candidates have the same name (or spelling) as the main or input SNE that we note ***AsSNE*** and those whose candidates have a different name (or spelling) than the main or input SNE that we note ***NotAsSNE***. Depending on the category, the adopted disambiguation approach will differ from other ambiguous SNEs.

- `resolveByFuzzyMatch` 3 : Fuzzy Matching (also referred to as approximate string matching) is a technique that helps identify two elements of text, strings, or entries that are approximately similar [21]. Similar to [3], we introduced these techniques as new strategies with the Geonames API to address the ***NotAsSNE*** set to solve problems such as typing problems (e.g., *United States* instead of *United States* and incomplete problems (e.g., *Angeles* instead of *Los Angeles*. We employed the Levenshtein [22] similarity measure to compute SNE proximity. We normalized the Levenshtein value from 0 to 1. A perfect match is equal to 1, and 0 indicates the lowest match, which means that the input and output do not have any semantic proximity. Two types of fuzzy matching strategies are defined in this context and are described below.
  - `SWHighMatch1`: This function is used to estimate single word similarities based on a high threshold. For example, *AmÂ©rica* and *America*. During the experiments (see Section 3), the semantic proximity is computed between the input SNE and all the candidates in the list. We consider the candidate with the highest proximity value. If the proximity value is greater than a given threshold, we assume that this value is the best and then is the final candidate *FinalCand*. Sometimes, several candidates can fulfill this high value condition. We introduce the population size to help make the final decision. Notably, based on the process of disambiguation by the Geonames API, the candidate with the largest population is probably the best candidate.
  - `SWHighMatch2`: For a single word match, this function is used to disambiguate the SNE with incomplete typing that `SWHighMatch1` is unable to resolve due to the weakness of the Levenshtein value. For example, *"Al-Minya"* and *"Minya"*. In this case, *"Minya"* is part of *"Al-Minya"*, and *"Al-Minya"* contains *"Minya"* as the longest substring. By querying "Minya", Geonames will return "Al-Minya", which is the correct spelling. To address this situation, the best candidate should contain or be a part of the input SNE, and their Levenshtein value should be greater than 0.7, a value that we experimentally fixed. Similar to the previous step, if several candidates meet these conditions, we use population size to make the final decision.
- `resolveByAlias` 4: An alias resolver is called after the fuzzy match strategies fail. Each country has an alias of two or three letters, Such as USA for United States, RDC for Republic Democratic of Congo, and BF for Burkina Faso. We also noticed some cases in which the SNE has multiple names. By considering this information, when *USA, U.S.A, US, or U.S* are extracted from the text, the candidate *United States* extracted from Geonames will be considered the best candidate.
- `scoreWithPopularCountryCode` 5: This function is only used for the AsSNE set. The main idea of this strategy is that when several nonambiguous SNEs from the same country are identified in the same document [23], the ambiguous SNE probably belongs to that country. We initialize a score of 0 for each candidate of the ambiguous SNE. Then, the score is incremented by 1 if its country code belongs to the country code of a nonambiguous SNE. At the end of the process, each candidate of the ambiguous SNE is scored by the number of increments, and its country code corresponds to the country code of the nonambiguous SNE in the same text. The largest population filter techniques are also applied in some studies, such as [2] and the Geonames disambiguation approach. Once the set candidates for each ambiguous SNE are scored, we can select the best candidate as follows:
  - `onePopularCountryCode`: In this case, only one country code has the highest score. First, we select all the candidates that belong to this country code, and second, we keep the candidate with the largest population as the best or final candidate.
  - `multiPopularCountryCode`: In this situation, more than one country code has the highest value. This is a case of equal possibilities. The final candidate is also selected based on the largest number of inhabitants.
- `defaultGeocoding`: Default geocoding consists of considering the default value that Geonames return for a given query. This step is applied in the latest phase, when the `resolveByFuzzyMatch` and `resolveByAlias` techniques failed to disambiguate an ambiguous SNE.

Fig. 2 illustrates the entire process and associated functionalities of the **SNEToolkit**.

3. Illustrative examples

To illustrate the functionality of the proposed approach for spatial named entity disambiguation, we utilized the GeoVirus [24] corpora.

Geovirus contains 229 articles from WikiNews[8] related to global disease outbreaks and epidemics. Place name mentions are manually tagged and assigned Wikipedia page URLs with their global coordinates. The GeoVirus dataset provides 1516 unique toponyms. For evaluation purposes, there were no duplicated toponyms in the same article. We filtered the GeoVirus corpus based on the steps illustrated in Fig. 3. 1360 of the 1362 SNEs are utilized for the experiments, as the two SNEs (*Karo* and *Soviet Union*) do not match any country code in Geonames although they have some coordinates.

Users can apply disambiguation from textual documents. In Fig. 4, we illustrate the line code with the main functions that achieve each step of the process. We assume that the user already extracted a set of SNEs from the document and retrieved the candidates for each SNE from Geonames (more details are available on the Github repository).

At Line 2, the user reads a JSON file that contains each set of SNEs with corresponding candidates by document. At Lines 4 & 5, we keep the two categories of SNE, i.e., the AsSNE set and NotAsSNE set. At Line 7, the user separates the ambiguous and

[8] https://en.wikinews.org

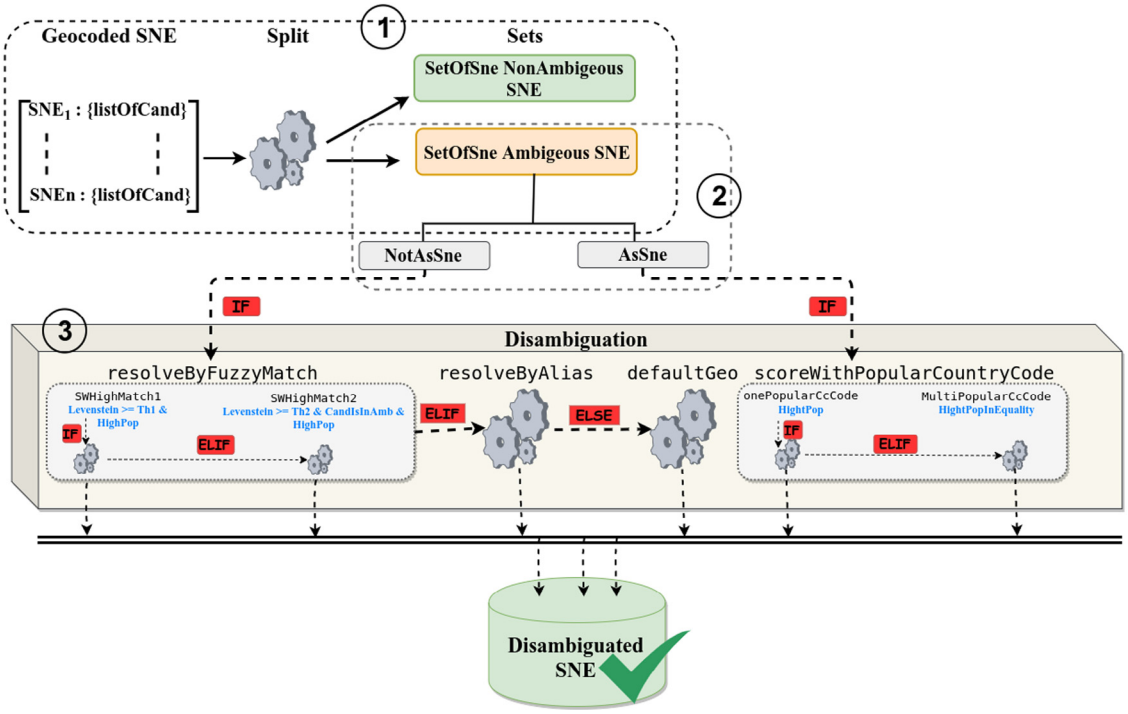

Fig. 2. Disambiguation functions.

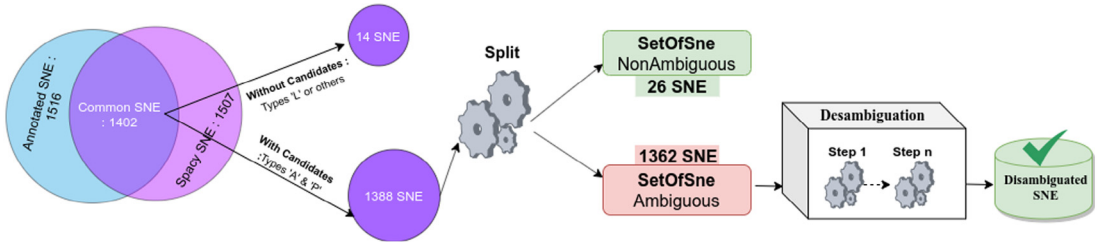

Fig. 3. GeoVirus data selection.

```python
def desambiguate(file : str, country_alias:dict):
    tmpCand = read_record(file)
    FinalCand = {}
    assne = tmpCand['assne']
    notassne = tmpCand['notassne']

    NonAmbigusSne, AmbigusSne, AllCcode, NonAmbCcode, r_notassne = splitAmbiguousAndNonAmnbiguous(assne, notassne)

    NonAmbigusSne_df = simplDicToDf(NonAmbigusSne)

    desambWithFuzzy, toBeDesambAgain = resolveByFuzzyMatch(r_notassne)
    FinalCand.update(desambWithFuzzy)

    resolvedSne, remainToBeDesambAgain = resolvByAlias(toBeDesambAgain, country_alias)
    FinalCand.update(resolvedSne)
    remainToBeDesambAgain_df =  simplDicToDf(remainToBeDesambAgain)

    DesambWithSc = scoreWithPopularCountryCode(AmbigusSne, AllCcode, NonAmbCcode)
    FinalCand.update(DesambWithSc)

    defaultDesamb = defaultGeocoding(remainToBeDesambAgain)
    FinalCand.update(defaultDesamb)

    desambiguated_df = dicToDf(FinalCand)

    desambiguated_df.to_csv('../desambiguated/desambiguated.csv', index = None)
    NonAmbigusSne_df.to_csv('../desambiguated/NonAmbigusSne.csv', index = None)
    remainToBeDesambAgain_df.to_csv('../remaining/toBeDesambAgain.csv', index = None)

    return desambiguated_df, NonAmbigusSne_df , remainToBeDesambAgain_df
```

Fig. 4. Disambiguation steps.

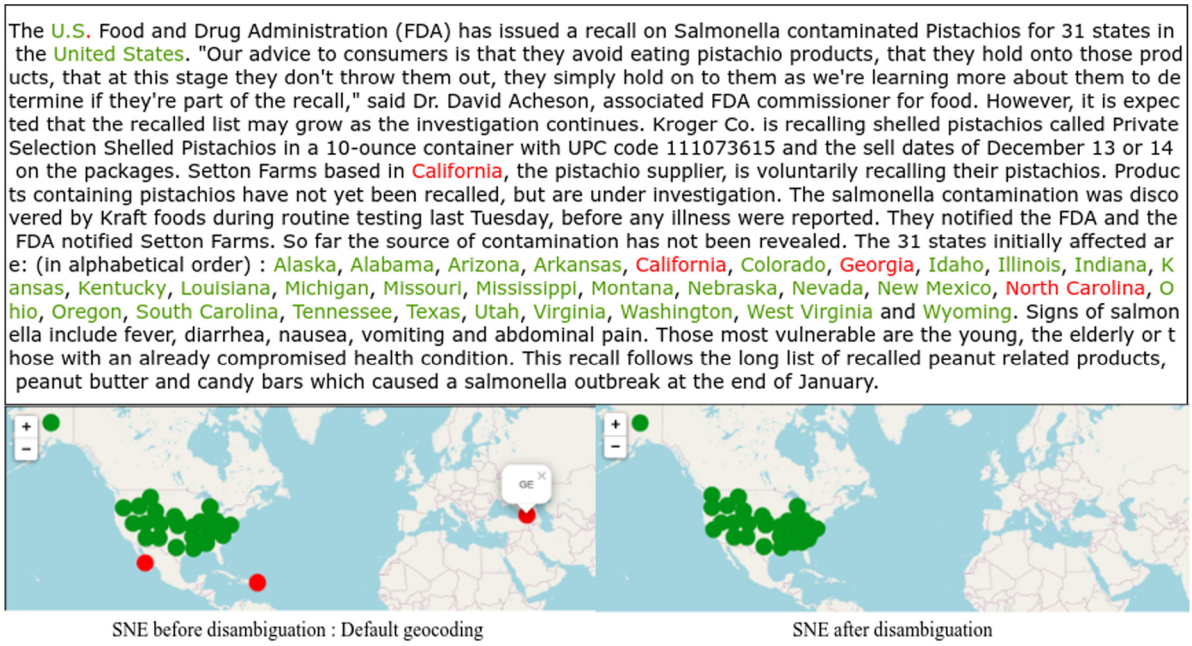

Fig. 5. SNE on map before and after the disambiguation process.

unambiguous SNEs from the two categories, including the associated set of country codes. At Line 11, the user applies fuzzy match techniques on the NotAsSNE set to disambiguate with the similarity measure. The *ToBeDisambAgain* set represents the set of SNEs that cannot be resolved by the fuzzy matching strategies. These remaining SNEs are then passed to the **resolveByAlias()** method for disambiguation again. At the end of the process, the SNEs that were not disambiguated by the alias (remainToBeDesambAgain) are geocoded by using the Geonames default geocoding approach at Line 21. The set of ambiguous AsSNEs is disambiguated at Line 18. From Lines 26 to 28, we save the outputs for evaluation and validation purposes.

Fig. 5 shows an example of a disambiguated SNE extracted from a textual document in the GeoVirus corpora. From left to right, the default geocoding missed the proper coordinate for three locations, as indicated by red dots. In this text, all 32 locations are related to the United States, with country code US. The errors occurred on *California*, *Georgia*, and *North Carolina*, which are geocoded as part of *Mexico MX, Georgia GE*, and *Puerto Rico, PR*, respectively. The green dots correspond to the locations that are correctly solved with both approaches.

To evaluate the proposed approach, experiments are conducted on the GeoVirus dataset. Table 1 summarizes the different scores for each step. We note that the number of disambiguated SNEs keeps increasing from the `FuzzyMatch` step to the `ScoreWithPop` step (1144 to 1204). As the ground truth data were annotated by experts, each label is considered true (all locations are true), and the precision is always equal to 1.0. The main goal in this context is to improve the value of the recall, which considers the predicted values (disambiguated or default geocoding value). By considering the remaining SNEs after each step as errors, the recall and F-score increase after each step, from 0.806 to 0.850. On the other hand, when the remaining SNEs are resolved or completed by using default geocoding, we also notice a high value of recall with a small variation, from 0.910 to 0.911. For the default geocoding on the whole dataset (1360 SNEs), the recall is equal to 0.871. Note that combining the disambiguation process and the default geocoding helps significantly improve (0.911 vs. 0.871) the disambiguation so that it is more accurate. We demonstrate that our disambiguation approach helps increase the accuracy of assigning relevant coordinates to the SNE that can be ambiguous in textual data. Fig. 6 illustrates the number of SNEs disambiguated per step in the process.

In Fig. B.7, we show the remaining SNEs to be disambiguated with their frequency, i.e., the number of times they are extracted from the whole corpus. We observe that the top 3 SNEs are *Africa*, *Asia*, and *Europe*. In these cases, none of the integrated strategies helps disambiguate them. This result is essentially attributed to the continental hierarchical level of these SNEs, which is higher than the country level. Thus, when querying, for example, Africa, Geonames returns all the countries that are on this continent, rendering the existing disambiguation criteria ineffective. We also have examples such as *South Kotabato* and *Wester Kasai*, which could be considered relative spatial named entities and cannot be address with the criteria that we defined.

4. Impact

Addressing named entities related to the location or place name (or spatial named entities) is in demand among natural language processing tasks. The disambiguation (i.e., giving the correct location, name & coordinates to a given SNE identified in the text) can be considered a main task goal or an intermediate task such as a prepossessing task. This tool can be utilized for many purposes: (1) textual data analysis, (2) spatial named entity extraction and geocoding, (3) enrichment of event monitoring systems based on textual data, or (4) spatiotemporal data analysis based on unstructured data. Practically, in epidemiological surveillance (human, plant, and animal) and in food safety [25], this tool can be integrated into such tools to improve

Table 1
Evaluation of the disambiguation approach on the GeoVirus dataset. *Remain* corresponds to the number of SNEs that remain after applying the three disambiguation approaches. *Remain as errors:* corresponds to the remaining SNEs for each step as errors due to disambiguation, and *Remain resolved by defaultGeocoding* corresponds to the default geocoding values. The precision, recall and F-score are calculated for each case.

Disambiguation steps	Total number of SNE to be disambiguated: 1360		Evaluation metrics	
	Disambiguated	*Remain as error* | *Remain resolved by defltGeo*	Recall	F-Score
`FuzzyMatch (F)`	1144	216 | 216	0.806 | 0.910	0.892 | 0.953
`F+ResolveByAlias (FA)`	1201	159 | 159	0.848 | 0.911	0.917 | 0.953
`FA+ScoreWithPop`	1204	156 | 156	0.850 | **0.911**	0.912 | 0.953
`DefaultGeocoding (Baseline)`	1360	0	**0.871**	0.931

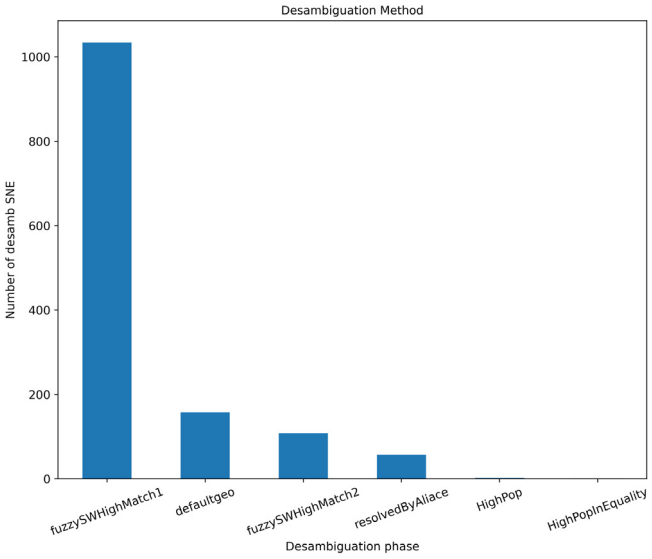

Fig. 6. Number of SNEs disambiguated for each stage of the disambiguation: `FA + ScoreWithPop`. The remaining SNEs are geocoded with the default geocoding values.

the extraction of events. This integration could have an impact on surveillance platforms where the detection of unambiguous epidemiological outbreaks is crucial.

5. Conclusion

In this paper, we presented a spatial named entity toolkit, SNEToolkit, which integrates three main functionalities: (1) SNE extraction from textual documents based on spaCy, (2) SNE geocoding, which returns the coordinates of a given SNE based on the Geonames database. There are two types of geocoding: (i) default candidate geocoding, which returns one candidate that corresponds to the default result of Geonames, and (ii) multicandidate geocoding, which returns the candidates from the Geonames results. (3) Disambiguation of the ambiguous SNEs within a document. This last point is our main research contribution. Based on multiple techniques, the disambiguiser helps disambiguate ambiguous SNEs from the multicandidate geocoding results. The outputs of the disambiguation process are the ambiguous SNEs that are disambiguated (relevant candidate from multiple candidates) and the nonambiguous SNEs that are extracted from the document. The analysis shows that SNEToolkit performs well with the standard geocoding of geonames. Therefore, SNEs such as continent names and relative spatial entities need to be addressed with other techniques that we expect to include in the next version of SNEToolkit. All functionalities are accessible and can be tested via the following repository: https://github.com/rdius/Snetoolkit.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Maguelonne Teisseire reports was provided by INRAE. Maguelonne Teisseire reports a relationship with French National Institute for Agricultural Research INRAE that includes: employment.

Data availability

Data will be made available on request

Acknowledgments

This study was funded by the EU grant 874850 MOOD and is cataloged as MOOD057 and by the BEYOND project (ANR Contract No. 20-PCPA-0002). The contents of this publication are the sole responsibility of the authors and do not necessarily reflect the views of the European Commission.

Appendix A. Disambiguation process pseudocode

Algorithm 1 Ambiguous & non-ambiguous SNE split function

Require: *AllSneCandidate* ← `json object, keys:sne, value: candidates`

NonAmbigusSne = ϕ
AmbigusSne= ϕ
AllCountryCode= ϕ
NonAmbigusCountryCode= ϕ
for all *sne* ∈ *AllSneCandidate* **do**
  for all *cc* ∈ *AllSneCandidate*[*sne*] **do**
    AllCountryCode[*sne*].*append*(*cc*)
  end for
  if *size*(*AllSneCandidate*[*sne*]) == 1 **then**
    NonAmbigusSne.*append*(*sne*)
    NonAmbigusCountryCode.*append*(*sne*[*cc*])
  else if *size*(*AllCountryCode*[*sne*]) == 1 **then**
    NonAmbigusSne.*append*(*sne*)
    NonAmbigusCountryCode.*append*(*sne*[*cc*])
  else
    AmbigusSne.*append*(*sne*)
    AmbigusCountryCode.*append*(*sne*[*cc*])
  end if
end for
Return *NonAmbigusSne*, *AmbigusSne*, *AllCountryCode*, *NonAmbigusCountryCode*

Algorithm 2 computeNL: More similar SNE using the normalized Levenshtein function

Require: *sne* # input spatial SNE
Require: *sneCandList* # List of candidates of the SNE
NL = *NormalizedLevenshtein*() # python implementation of Levenshtein
tempList= ϕ
for all *candSne* ∈ *sneCandList* **do**
  sim_val = *NL.similarity*(*sne*, *candSne*)
  tempList.append((*candSne*, *sim_val*))
  if *size*(*tempList*) > 0 **then**
    tempCand = *max*(*tempList.sim_val*) # candidate with the highest similarity value
  end if
end for
Return *tempCand* # tempCand is a tuple: (candSne,sim_val)

Algorithm 3 ResolveByFuzzy function

Require: *AmbigusSne* #json object, keys:ambsne, value: candidates
Require: *Th*1, *Th*2 # similarity thresholds
NotAsSNE ∈ *AmbigusSne* # Set of SNEs where the names of candidates are not the same as the input
DisambWithFuzzy= ϕ
ToBeDisambAgain= ϕ
for all *ambSne* ∈ *NotAsSNE* **do**
  listOfCandidate = *NotAsSne*[*ambSne*]
  Disambiguated = ϕ
  sneCandList = *AllSneCand_name* ∈ *listOfCandidate*
  tempCand = *computeNL*(*ambSne*, *sneCandList*)
  sim_val = *tempCand*[1] # similarity measure value
  candSne = *tempCand*[0] # sne name
  if *size*(*tempCand*) > 0 & *sim_val* > *Th*1 **then**
    potentialCandList = [*candSne*]
    potentialCand = *max*(*potentialCandList.population*) # keep SNEs with a large population
    Disambiguated[*ambSne*] = *potentialCand* # SWHighMatch1
    DisambWithFuzzy.update(*Disambiguated*)
  else if *checkCompletude*(*amb*, *ambL*, *Th*2) **then**
    sneCandList = *AllSneCand_name* ∈ *listOfCandidate*
    potentialCand = *max*(*potentialCandList.population*)
    Disambiguated[*ambSne*] = *potentialCand*
    DisambWithFuzzy.update(*Disambiguated*)
  else
    ToBeDisambAgain[*ambSne*] = *NotAsSne*[*ambSne*]
  end if
end for
Return *DesambWithFuzzy*, *ToBeDesambAgain*

Algorithm 4 ResolveByAlias function

Require: *toBeDesambAgain* #json object, keys:ambsne, value: candidates
Require: *Country_alias* # json object, keys:country code, value: corresponding country name
DisambWithAlias= ϕ
RemainToBeDisambAgain= ϕ
for all *ambSne* ∈ *toBeDesambAgain.keys*() **do**
  if *ambSne* ∈ *country_alias.keys*() **then**
    potentialCandList = *country_alias.values*()
    potentialCand = *max*(*potentialCandList.population*)
    DisambWithAlias[*ambSne*] = *potentialCand*
  else
    RemainToBeDisambAgain[*ambSne*] = *toBeDesambAgain*[*ambSne*]
  end if
end for
Return *DisambWithAlias*, *RemainToBeDisambAgain*

Algorithm 5 ScoreWithPopular function

Require: *AmbigusSne*
Require: *AllCountryCode*
Require: *NonAmbigusCountryCode*
AsSNE ∈ *AmbigusSne* `#Set of SNEs where names of candidates are the same as the input`
DisambWithScore= ϕ
Score= ϕ
for all *ambSne* ∈ *AsSNE*.*keys*() **do**
  Score[*ambSne*] = ϕ
  for *country_code* ∈ *AllCountryCode*[*ambSne*] **do**
    Score[*ambSne*][*country_code*] = 0
    for *Ccode* ∈ *AllCountryCode* **do**
      if *country_code* ∈ *AllCountryCode*[*Ccode*] **then**
        Score[*ambSne*][*country_code*]+ = 1
      else
        continue
      end if
    end for
  end for
end for
for *ambSne* ∈ *Score* **do**
  potentialLoc = *list*(*CountryCodeWithTopValue*)
  if *size*(*potentialLoc*) == 1 **then**
    bestCc = *potentialLoc*.*Cc* `#one candidate match with the highest score`
    potentialCandidList.*append*(*ambSne*, *Where*(*AsSNE*.*Cc* == *bestCc*)
    DisambWithScore[*ambSne*] = *max*(*potentialCandList*.*population*)
  else if *size*(*potentialLoc*) > 1 **then**
    tpotentialCandidList = [∗*ambsne*, *Where*(*ambsne*.*Cc* ∈ *potentialLoc*)]
    DisambWithScore[*ambSne*] = *max*(*potentialCandList*.*population*)
  end if
end for
Return *DisambWithScore*

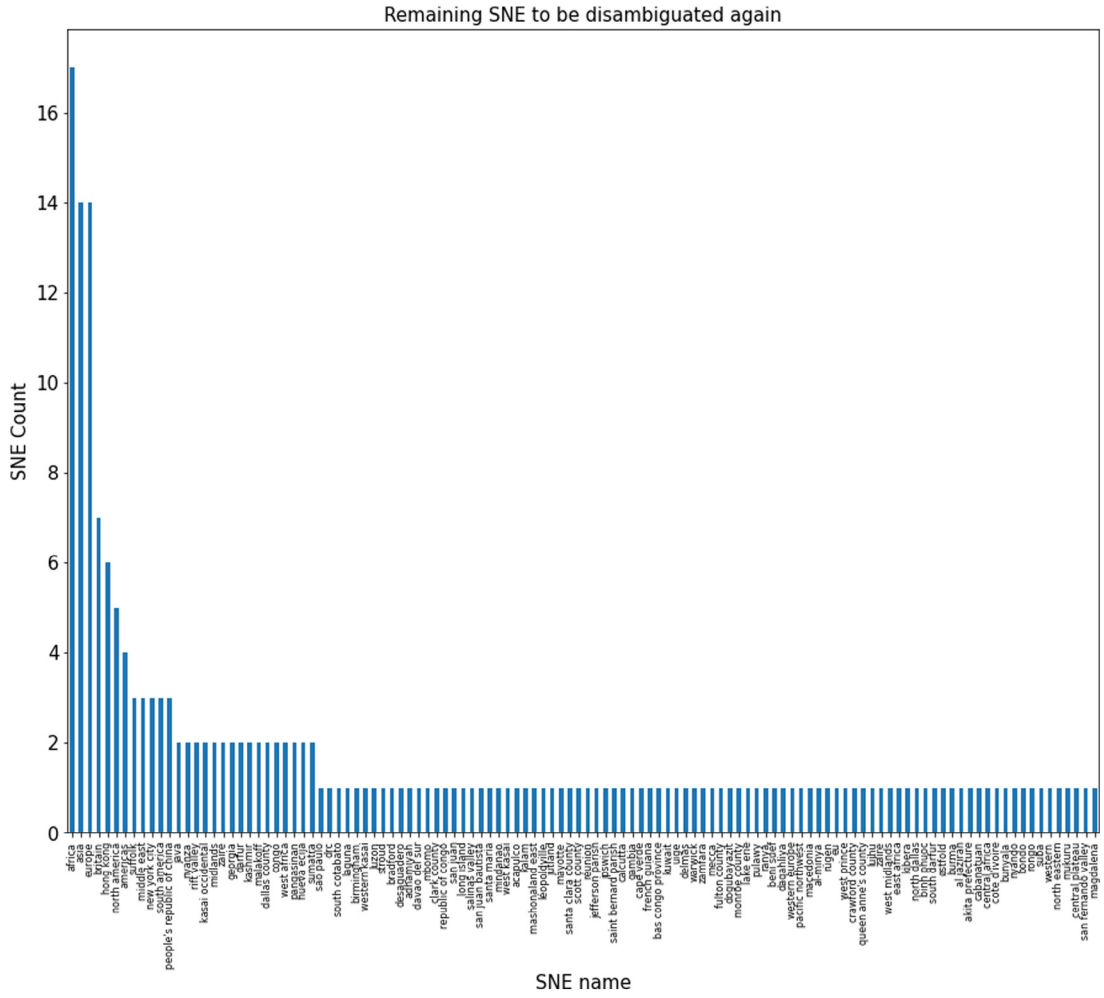

Fig. B.7. Remaining SNE to be disambiguated.

Appendix B. Errors analysis

See Fig. B.7.

References

[1] Kergosien E, Alatrista-Salas H, Gaio M, Güttler FN, Roche M, Teisseire M. When textual information becomes spatial information compatible with satellite images. In: 2015 7th International joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K), Vol. 1. IEEE; 2015, p. 301–6.

[2] Gritta M, Pilehvar MT, Limsopatham N, Collier N. What's missing in geographical parsing? Lang Resour Eval 2018;52(2):603–23.

[3] Kanagavalli V, Raja K. Detecting and resolving spatial ambiguity in text using named entity extraction and self learning fuzzy logic techniques. 2013, CoRR, abs/1303.0445.

[4] Bouarroudj W, Boufaida Z, Bellatreche L. Named entity disambiguation in short texts over knowledge graphs. Knowl Inf Syst 2022;64(2):325–51.

[5] Aldana-Bobadilla E, Molina-Villegas A, Lopez-Arevalo I, Reyes-Palacios S, Muñiz-Sanchez V, Arreola-Trapala J. Adaptive geoparsing method for toponym recognition and resolution in unstructured text. Remote Sens 2020;12(18):3041.

[6] Santos J, Anastácio I, Martins B. Using machine learning methods for disambiguating place references in textual documents. GeoJournal 2015;80(3):375–92.

[7] Kulkarni S, Jain S, Hosseini MJ, Baldridge J, Ie E, Zhang L. Spatial language representation with multi-level geocoding. 2020, arXiv preprint arXiv: 2008.09236.

[8] Mai G, Janowicz K, Hu Y, Gao S, Yan B, Zhu R, et al. A review of location encoding for geoai: methods and applications. Int J Geogr Inf Sci 2022;36(4):639–73.

[9] Teitler BE, Lieberman MD, Panozzo D, Sankaranarayanan J, Samet H, Sperling J. NewsStand: A new view on news. In: Proceedings of the twenty-ninth AAAI conference on artificial intelligence. New York, NY, USA: Association for Computing Machinery; 2008, p. 10.

[10] Speriosu M, Baldridge J. Text-driven toponym resolution using indirect supervision. In: Proceedings of the 51st annual meeting of the association for computational linguistics (Volume 1: Long papers). 2013, p. 1466–76.

[11] Andogah G. Geographically constrained information retrieval. University Library Groningen; 2010.

[12] DeLozier G, Baldridge J, London L. Gazetteer-independent toponym resolution using geographic word profiles. In: Proceedings of the twenty-ninth AAAI conference on artificial intelligence. AAAI Press; 2015, p. 2382–8.

[13] Garbin E, Mani I. Disambiguating toponyms in news. In: Proceedings of human language technology conference and conference on empirical methods in natural language processing. 2005, p. 363–70.

[14] Han B, Cook P, Baldwin T. Text-based twitter user geolocation prediction. J Artificial Intelligence Res 2014;49:451–500.

[15] Hu X, Sun Y, Kersten J, Zhou Z, Klan F, Fan H. How can voting mechanisms improve the robustness and generalizability of toponym disambiguation? Int J Appl Earth Obs Geoinf 2023;117:103191.

[16] Honnibal M, Montani I. spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. 2017, in press.

[17] Akbik A, Bergmann T, Blythe D, Rasul K, Schweter S, Vollgraf R. FLAIR: An easy-to-use framework for state-of-the-art NLP. In: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics (demonstrations). 2019, p. 54–9.

[18] Qi P, Zhang Y, Zhang Y, Bolton J, Manning CD. Stanza: A python natural language processing toolkit for many human languages. 2020, arXiv preprint arXiv:2003.07082.

[19] Shelar H, Kaur G, Heda N, Agrawal P. Named entity recognition approaches and their comparison for custom ner model. Sci Technol Libr 2020;39(3):324–37.

[20] Tahrat S, Kergosien E, Bringay S, Roche M, Teisseire M. Text2Geo: From textual data to geospatial information. In: Proceedings of the 3rd international conference on web intelligence, mining and semantics. New York, NY, USA: Association for Computing Machinery; 2013, p. 1–4. http://dx.doi.org/10.1145/2479787.2479796.

[21] Chen G, Pham TT. Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems. CRC Press; 2000.

[22] Levenshtein VI, et al. Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady, Vol. 10. Soviet Union; 1966, p. 707–10.

[23] Fize J, Shrivastava G, Ménard PA. GeoDict: an integrated gazetteer. In: Proceedings of language, ontology, terminology and knowledge structures workshop. Montpellier, France: Association for Computational Linguistics; 2017, p. 1–11, URL https://aclanthology.org/W17-7004.

[24] Gritta M, Pilehvar MT, Collier N. Which Melbourne? Augmenting Geocoding with Maps. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long papers). Melbourne, Australia: Association for Computational Linguistics; 2018, p. 1285–96. http://dx.doi.org/10.18653/v1/P18-1119, URL http://aclweb.org/anthology/P18-1119.

[25] Morris CE, Geniaux G, Nédellec C, Sauvion N, Soubeyrand S. One health concepts and challenges for surveillance, forecasting, and mitigation of plant disease beyond the traditional scope of crop production. Plant Pathol 2022;71(1):86–97.