



Qualitative and quantitative research in the humanities and social sciences: how natural language processing (NLP) can help

Roberto Franzosi¹ · Wenqin Dong² · Yilin Dong²

Accepted: 2 September 2021 / Published online: 23 September 2021
© The Author(s), under exclusive licence to Springer Nature B.V. 2021

Abstract

The paper describes computational tools that can be of great help to both qualitative and quantitative scholars in the humanities and social sciences who deal with words as data. The Java and Python tools described provide computer-automated ways of performing useful tasks: 1. check the filenames well-formedness; 2. find user-defined characters in English language stories (e.g., social actors, i.e., individuals, groups, organizations; animals) (“find the character”) via WordNet; 3. aggregate words into higher-level aggregates (e.g., “talk,” “say,” “write” are all verbs of “communication”) (“find the ancestor”) via WordNet; 4. evaluate human-created summaries of events taken from multiple sources where key actors found in the sources may have been left out in the summaries (“find the missing character”) via Stanford CoreNLP POS and NER annotators; 5. list the documents in an event cluster where names or locations present close similarities (“check the character’s name tag”) using Levenshtein word/edit distance and Stanford CoreNLP NER annotator; 6. list documents categorized into the wrong event cluster (“find the intruder”) via Stanford CoreNLP POS and NER annotators; 7. classify loose documents into most-likely event clusters (“find the character’s home”) via Stanford CoreNLP POS and NER annotators or date matcher; 8. find similarities between documents (“find the plagiarist”) using Lucene. These tools of automatic data checking can be applied to ongoing projects or completed projects to check data reliability. The NLP tools are designed with “a fourth grader” in mind, a user with no computer science background. Some five thousand newspaper articles from a project on racial violence (Georgia 1875–1935) are used to show how the tools work. But the tools have much wider applicability to a variety of problems of interest to both qualitative and quantitative scholars who deal with text as data.

Keywords Words as data · Research in humanities and social sciences · Social movements · Natural language processing · NLP · Computational linguistics

✉ Roberto Franzosi
rfranzo@emory.edu

¹ Department of Sociology/Linguistics Program, Emory University, Atlanta, GA, USA

² Carnegie Mellon University, Pittsburgh, PA, USA

1 Introduction: social movement research (and beyond)

In 1988, Leo Panitch wrote a review article on corporatism for the *Canadian Journal of Political Science* that read:

At the end of the 1970s, the corporatism growth industry in political science passed from its competitive stage (articles in journals) to its organized stage (articles collected in books). ... Alan Cawson's *Corporatism and Political Theory* ... represents, perhaps, the coming of the monopoly stage in the industry's development. ... A growth industry indeed!" (Panitch 1988: 813, 814)

That industrial metaphor would later be adopted by Bert Klandermans and Suzanne Staggenborg in reference to a different scholarly endeavour: social movement research. In the introduction to their edited volume on *Methods of Social Movement Research*, Klandermans and Staggenborg, wrote (2002: xi-xii):

Political process theory also offered an innovative method: protest event analysis provided a way of measuring the effects of political opportunities in comparative designs...The *protest event* became the unit of analysis in an influential new approach to social movements research that used newspapers as a data source. More recently, other sources such as police files and reports of press agencies have been used. But the fundamental methodological approach has remained the same, resulting in a virtual industry of protest event data analysis in the study of social movements and other forms of contention.

This paper contributes to the growth of the social movements industry whether Protest Event Analysis (PEA) or Quantitative Narrative Analysis (QNA).¹ It introduces a set of computer-automated tools that can be profitably used in the process of data-reliability checking in an industry where data collection still relies, at best, on computer-assisted coding of events,² despite the fact that completely automated ways of extracting protest event data are becoming more prominent (Author reference 2021; Lansdall-Welfare et al. 2017; Zhang and Pan, 2019; Lansdall-Welfare and Cristianini 2020). More generally, the paper contributes to a growing industry of computational social sciences and digital humanities.³ After all, the tools provide computer-automated ways of performing very general useful tasks to scholars who deal with text documents as data: 1. check the filenames well-formedness; 2. find user-defined characters in the English language (e.g., social actors, i.e., individuals, groups, organizations; animals) ("find the character") via WordNet; 3. aggregate words into higher-level aggregates (e.g., "talk," "say," "write" are all verbs of "communication") ("find the character's ancestor") via WordNet; 4. evaluate human-created summaries of historical events taken from multiple sources where key social actors found in the sources may have been left out in the summaries ("find the missing character") via Stanford CoreNLP; 5. list the documents in an event cluster where names or locations present close similarities ("check the character's name tag") using Levenshtein word/edit distance

¹ On PEA see Koopmans and Rucht (2002) and (Hutter 2014); on PEA and its more rigorous methodological counterpart rooted in a linguistic theory of narrative and rhetoric, Quantitative Narrative Analysis (QNA), see Franzosi (2010).

² See, for instance, Franzosi's PC-ACE (Program for Computer-Assisted Coding of Events) at www.pc-ace.com (Franzosi 2010).

³ For recent surveys, see Evans and Aceves (2016), Edelman et al. (2020).

and Stanford CoreNLP; 6. list documents categorized into the wrong event cluster (“find the intruder”) via Stanford CoreNLP; 7. classify loose documents into most-likely event clusters (“find the character’s home”) via Stanford CoreNLP; 8. find similarities between documents (“find the plagiarist”) via Lucene.

These automatic data-checking tools, written in both Java and Python 3, can be applied to ongoing projects or completed projects to check data reliability. They come with user-friendly Graphical User Interface (GUI) windows, videos, TIPS files, help buttons, and reminders (Fig. 1 shows the GUI for the overall set of tools but each tool comes with specific GUIs: for filename checker, WordNet, Levenshtein’s word/edit distance). The tools are part of a set of NLP tools, freely available on GitHub (<https://github.com/NLP-Suite/NLP-Suite/wiki>), aimed at delivering cutting-edge algorithms in Natural Language Processing for users with zero computer-science background.⁴ The main algorithm with the Graphical User Interface (GUI) shown in Fig. 1 is the Social Science Research tool in the Suite.

The tools described typically have two computational components: 1. identification of errors/problems of various types in the input data, with output as csv files; 2. visualization of results as Excel charts.⁵

2 Our test data

As a way of illustration, the paper relies on data on lynchings in Georgia (1875–1935) taken from the Beck and Tolnay inventory of lynching events (Beck and Tolnay 2004) and Franzosi’s later elaboration (Franzosi et al. 2012). The lynching data are based on newspapers as sources (some 1000 of them),⁶ with several articles (and from different newspapers) available on each event. But the tools can also be used for different types of data (e.g., protest movement websites, blogs, researchers’ field notes, letters), organized differently from the newspapers-events structure, such as short stories-topics, field notes-informants, letters-recipients. We can generically refer to events, topics, informants, recipients as clusters and the various types of input texts as documents (newspapers, short stories, field notes, letters) (Fig. 2).

The filenames of each of these 1000 documents follow specific file-naming criteria that embed metadata that can be used for analysis: newspaper name, date, page number, and column number, each separated by an underscore symbol _ (e.g., The Atlanta Constitution_01-22-1892_4_2). Since each document is cross-referenced to a specific event,⁷ the filename can also contain information about the event in user-defined forms (e.g., the ID number of the event, the name of the individual lynched, the location of the event, all similarly separated by an underscore symbol, and with the newspaper information separated from the event information by a double underscore __, e.g., The Atlanta

⁴ The GitHub site will automatically install not only all the NLP Suite scripts but also Python and Anaconda required to run the scripts. It also provides extensive help on how to download and install a handful of external software required by some of the algorithms (e.g., Stanford CoreNLP, WordNet). The goal is to make it as easy as possible for non-technical users to take advantage of the tools with minimal investment.

⁵ We rely on the Python package `openpyxl` and ad hoc functions.

⁶ The newspaper collections found in Chronicling America of the Library of Congress (<http://chroniclingamerica.loc.gov/newspapers/>), the Digital Library of Georgia (<http://dlg.galileo.usg.edu/MediaTypes/Newspapers.html?Welcome>), The Atlanta Constitution, Proquest, Readex.

⁷ Multiple cross-references are also possible, whereby a document deals with several different events.

Constitution_01-22-1892_4_2__3, The Atlanta Constitution_05-24-1918_2_1__Jim Cobb).⁸

These 1,000 newspaper articles can be grouped together by event ID, a number that uniquely identifies a specific event (e.g., 3), or by the name of a significant individual (e.g., the lynched man, such as Jim Cobb) or a date (e.g., 1918).

But whatever naming criteria one adopts, each subfolder contains the articles that describe a specific event. Thus, event number 3 (the lynching of Jim Cobb in 1918) contains the following articles.

3 Check the filenames well-formedness

3.1 The problem

Even a quick look at the files listed in Fig. 3 reveals the range of errors users can make when saving newspaper articles that embed metadata in the filename in formatted ways. Thus, some of the filenames include the column number (e.g., Atlanta Constitution_05-24-1918_2_1.pdf, Cordele Dispatch_05_23_1918_1_1.pdf), others do not (e.g., Chicago Defender_06-01-1918_9.pdf, Keowee Courier_05-29-1918_2.pdf). Some have newspaper names that are not 100% correct (e.g., Atlanta Constitution strictly speaking should be *The Atlanta Constitution*). Some do not have a page number (e.g., The State_05-23-1918.pdf). The date is formatted incorrectly in some cases (e.g., Cordele Dispatch_05_23_1918_1_1.pdf) where date items are separated by _ instead of -; and in other cases the month, or day, or year are missing (e.g., The Quitman Free Press_05-1918_1_2.pdf).

None of this is a problem if all you are going to do is to click on a file and read it (but 1000 or more articles?!). It is not even a problem if the filename is imported into a csv file, for instance, in its entirety. It does become a problem, however, if the filename is then processed for the embedded metadata (e.g., to obtain a frequency distribution of the newspaper sources, of the dates of events—Which days of the month? Which months? Which years?—the typical pages where these events were published). Then, a computer algorithm will have precise expectations of where and how to find information.

3.2 The computational solution

The Python script performs a wide range of filename checks. If you have a csv file with the correct names of newspapers, you can use those to catch *Atlanta Constitution* as an error (missing *The*) as the first item in the filename. If your metadata embeds people's names or locations (e.g., Jim Cobb) you could check those against a dictionary list. You can count the number of _ in a filename (the default separator between metadata items) to flag filenames with missing information. In our case, the correct number should be 3 (_ between

⁸ Contrary to some protest event projects based on a single newspaper source (e.g., The New York Times in the “Dynamics of Collective Action, 1960–1995” project that involved several social scientists, notably, Doug McAdam, John McCarthy, Susan Olzak, Sarah Soule, and led to dozens of influential publications; see for all McAdam and Su 2002), the Georgia lynching project is based on multiple newspaper sources for each event.

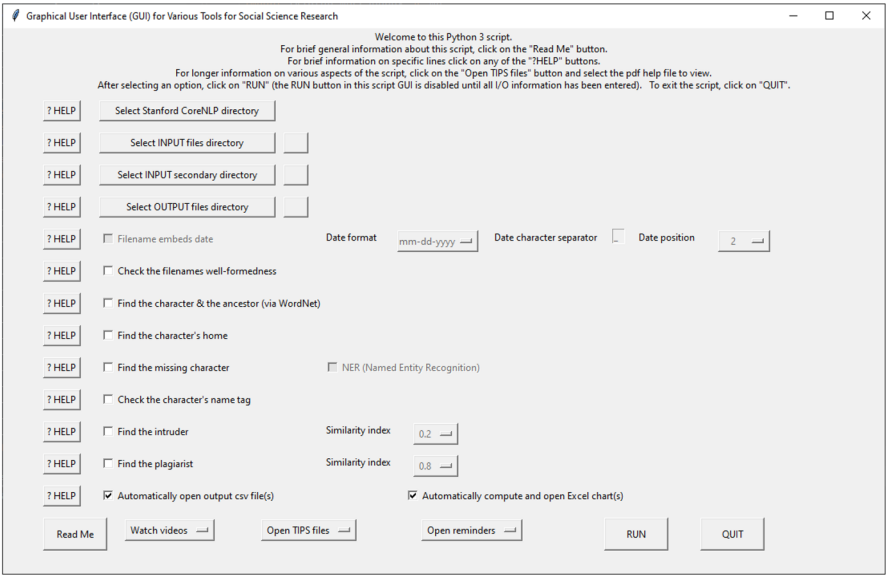


Fig. 1 Graphical user interface (GUI) for all options

Name	Date modified	Type	Size
3	2/18/2020 10:14 PM	File folder	
4	2/18/2020 10:15 PM	File folder	
5	2/18/2020 10:15 PM	File folder	
6	2/18/2020 10:15 PM	File folder	
7	2/18/2020 10:15 PM	File folder	

Name	Date modified	Type	Size
3_Jim Cobb	2/18/2020 10:14 PM	File folder	
4_Frank Hardeman	2/18/2020 10:15 PM	File folder	
5_Palseo	2/18/2020 10:15 PM	File folder	
6_Owen Jones	2/18/2020 10:15 PM	File folder	
7_Jet Hicks	2/18/2020 10:15 PM	File folder	

Name	Date modified	Type	Size
3_Jim Cobb_1918	2/18/2020 10:14 PM	File folder	
4_Frank Hardeman_1900	2/18/2020 10:15 PM	File folder	
5_Palseo_1890	2/18/2020 10:15 PM	File folder	
6_Owen Jones_1890	2/18/2020 10:15 PM	File folder	
7_Jet Hicks_1906	2/18/2020 10:15 PM	File folder	

Fig. 2 Event subdirectories structure

newspaper name and date, between date and page number, between page and column numbers). The script can check for properly formatted dates, if the filenames embed dates.











Name	Date modified	Type	Size
 Atlanta Constitution_05-24-1918_2_1.pdf	8/25/2011 7:05 PM	Adobe Acrobat Docu...	44 KB
 Chicago Defender_06-01-1918_9.pdf	2/9/2013 4:14 PM	Adobe Acrobat Docu...	26 KB
 Cordele Dispatch_05_23_1918_1_1.pdf	11/7/2012 10:28 AM	Adobe Acrobat Docu...	1,152 KB
 Keowee Courier_05-29-1918_2.pdf	2/25/2013 9:09 PM	Adobe Acrobat Docu...	1,255 KB
 Moultrie Semi-Weekly Observer_05-24-1918_1_...	8/25/2011 7:05 PM	Adobe Acrobat Docu...	957 KB
 The Atlanta Georgian_05-23-1918_2_1.pdf	10/27/2014 1:55 PM	Adobe Acrobat Docu...	359 KB
 The Macon News_05-23-1918_1_1.pdf	11/7/2012 12:20 PM	Adobe Acrobat Docu...	2,656 KB
 The Quitman Free Press_05-1918_1_2.pdf	10/27/2014 11:24 AM	Adobe Acrobat Docu...	557 KB
 The State_05-23-1918_1.pdf	2/25/2013 9:09 PM	Adobe Acrobat Docu...	20 KB
 The Washington Post_05-23-1918_1.pdf	2/25/2013 9:09 PM	Adobe Acrobat Docu...	22 KB

Fig. 3 List of articles in a specific subdirectory

4 Find the character

4.1 The problem

PEA/QNA projects have shown that manual coding of hundreds or thousands of documents (e.g., newspaper articles) into a coding scheme based on the 5 Ws+H narrative structure (Who, What, When, Where, Why, and How, or “story grammar” made up of SVOs, or Subject-Verb-Object and some modifiers) leads to highly disaggregated textual data made up of thousands of distinct words/textual expressions.⁹ Most subjects in such a relational database are social actors, i.e., individuals, groups, organizations, but the objects of the SVO may contain a great deal of “things/objects” (e.g., “incendiary burned down the barn,” where “barn” is not a social actor but a thing/object). Unless the coding scheme explicitly provides different coding categories for people, organizations, and “things/objects” for the Object in the SVO structure, we would not know which word/textual expression is which.

But while social actors, i.e., human agents, are the main characters in social movement research, in the study of folktales, animals may be the main characters (e.g., the wolf, the fox, the pigs):

Presently came along a wolf, and knocked at the door, and said: “Little pig, little pig, let me come in.” To which the pig answered: “No, no, by the hair of my chiny chin chin.” The wolf then answered to that: “Then I’ll huff, and I’ll puff, and I’ll blow your house in.” (*The Three Little Pigs*, from Jacob’s collection of English folktales, 1890)

Not just wolfs and pigs can talk. Things talk in fairy tales.

A year later the king took himself another wife. She was a beautiful woman, but she was proud and arrogant, and she could not stand it if anyone might surpass her in beauty. She had a magic mirror. Every morning she stood before it, looked at herself, and said: “Mirror, mirror, on the wall, who in this land is fairest of all?” To this the mirror answered: “You, my queen, are fairest of all.” (*Little Snow White*, Grimm)

⁹ Franzosi reports 1,600 distinct entries for subjects and objects and 7,000 for verbs for one of his projects (Franzosi 2010: 93); similar figures are reported by Ericsson and Simon (1996: 265–266) and Tilly (1995: 414–415).

And things even hop and dance:

Then Tatty sat down and wept; then a three-legged stool said: “Tatty, why do you weep?” “Titty’s dead,” said Tatty, and so I weep;” “then,” said the stool, “I’ll hop,” so the stool hopped. (*Titty Mouse and Tatty Mouse*, from Jacob’s collection of English folktales, 1890)

Do current computational linguistic tools provide a way to filter social actors, or animals for that matter, from a long list of words?

4.2 The computational solution

To answer the question, we developed a set of algorithms that rely on WordNet, an on-line, freeware lexical database developed at Princeton University (<https://wordnet.princeton.edu/>). In WordNet English nouns, verbs, adjectives, and adverbs are organized into synonym sets (called synsets, some 117,000 of them¹⁰), each representing one underlying lexicalized concept (Miller 1995; Fellbaum 1998). WordNet ignores prepositions, determiners and other function words. Our implementation further restricts the use of WordNet to nouns and verbs ignoring adjectives and adverbs.¹¹

For each synset, WordNet provides a brief definition (“gloss”) and, in most cases, one or more short sentences illustrating the use of the synset members. Synsets, in turn, are organized hierarchically into superordinate/subordinate relations (hyponym, and the relation also called hyperonymy, hyponymy or ISA relation) that link more general synsets to increasingly specific synsets. Thus, in WordNet, the synset *furniture* includes *bed*, which in turn includes *bunkbed*; conversely, concepts such as *bed* and *bunkbed* make up the category *furniture*. English words that have different meanings belong to different synsets and hyponyms, different for each meaning.

We developed a Java tool based on MIT JWI (Java Wordnet Interface).¹² We imported MIT JWI and implemented a depth-first search (DFS) in Java to construct comprehensive lists (e.g., of social actors). We further developed a user interface in Python 3 (see Fig. 24) that connects to the Java script via the Zoom IN/DOWN option. In our approach, we allow users to search the WordNet database either by top synsets (25 Nouns and 15 Verbs) or by one or more words (or collocations, i.e., combinations of multiple words¹³) of their choice. If the user selects a top synset, the Java algorithm uses the MIT JWI library to go through every word in the WordNet database and filter out all the words that are under a chosen

¹⁰ The most up-to-date numbers of terms are given in <https://wordnet.princeton.edu/documentation/wstats7wn>.

¹¹ A common critique of WordNet is that WordNet is better suited to account for concrete concepts than for abstract concepts. It is much easier to create hyponyms/hypernym relationships between “conifer” as a type of “tree”, a “tree” as a type of “plant”, and a “plant” as a type of “organism”. Not so easy to classify emotions like “fear” or “happiness” into hyponyms/hypernym relationships.

¹² <https://projects.csail.mit.edu/jwi/>

¹³ The WordNet databases comprises both single words or combinations of two or more words that typically come together with a specific meaning (collocations, e.g., coming out, shut down, thumbs up, stand in line, customs duty). Over 80% of terms in the WordNet database are collocations, at least at the time of Miller et al.’s Introduction to WordNet manual (1993, p. 2). For the English language (but WordNet is available for some 200 languages) the database contains a very large set of terms. The most up-to-date numbers of terms are given in <https://wordnet.princeton.edu/documentation/wstats7wn>.

category. Alternatively, if the user inputs several search words, the Java algorithm uses DFS to recursively find all hyponyms of the search words. In DFS, we first push into a stack all source words (i.e., we insert each source word at the top of the stack). We then pop a word for searching (i.e., we remove the word from the top of the stack) and add all its hyponyms into the stack. Since a word can have multiple meanings in the WordNet database, including quite rare meanings, we include only the 3 most frequent meanings and the hyponyms under these frequent meanings. We repeat this process until there are no more words in the stack, i.e., we have searched all possible hyponyms. Thus, a search based on the top noun synset ‘person’ will export a list of some 19,000 proper and improper names of individuals, groups, and organizations (see Fig. 4). Searching for ‘animal’ will lead to a similar list of over 14,000 entries. Since proper nouns have an upper-case first letter and improper nouns have a lower-case first letter, we can use these features to export proper nouns or improper nouns selectively.

5 Find the character’s ancestor

5.1 The problem

The lynching database contains over 19,000 distinct words/textual expressions (e.g., sheriff, deputy sheriff, mob) in nearly 200 coding categories: 150 distinct individual actors, 124 collective actors (e.g., mob), 50 organizations (e.g., National Guard), some 500 things (e.g., a barn, jail), and, especially nearly 1400 verbs. Even a few hundred words become intractable if we want to display relationships between social actors in a network graph. The graph would simply be too cluttered. We would need to aggregate different words into common ancestors, i.e., into higher-level aggregates.¹⁴ Thus, “talk,” “say,” “write” are all verbs of “communication,” and “run,” “flee,” and “walk” are all verbs of “movement.” Traditionally, this type of painful process of data reduction has been done by hand. Developments in NLP in recent years allow us to perform this aggregation task automatically.

5.2 The computational solution

We developed a tool that uses the same WordNet lexicon database as *Find the Character* tool, but in the opposite direction.¹⁵ We start at the lowest level of a single word or collocation¹⁶ of either nouns or verbs and move up the hierarchy of synsets all the way to the top synset to extract one of 25 top synsets for nouns or 15 top synsets for verbs (from parent, to

¹⁴ Data aggregation is often referred to as “data reduction” in the social sciences and as “linguistic categorization” in linguistics (on linguistic categorization, see Taylor 2004; on verbs classification, Levin 1993; see also Franzosi 2010: 61).

¹⁵ On the way up through the hierarchy, the script relies on the WordNet concepts of *hypernym* – the generic term used to designate a whole class of specific instances (Y is a hypernym of X if X is a (kind of) Y) – and *holonym* – the name of the whole of which the meronym names is a part. Y is a holonym of X if X is a part of Y.

¹⁶ Collocations are sets of two or more words that are usually together for a complete meaning, e.g., “coming out,” “sunny side up”. Over 80% of terms in the WordNet database are collocations, at least at the time of Miller et al.’s *Introduction to WordNet manual* (1993, p. 2). For the English language (but WordNet is available for some 200 languages) the database contains a very large set of terms. The most up-to-date numbers of terms in each category are given in <https://wordnet.princeton.edu/documentation/wstats7wn>

grandparent, great grandparent, ...—the “ancestors”; for a list of top synsets¹⁷). The automatic data-aggregation algorithm (the Zoom OUT/UP widget of Fig. 24) only aggregates words at the WordNet top synsets levels, rather than at intermediate synsets levels;¹⁸ as a result, different types of aggregation required by domain-specific or research-specific questions would require a combination of automatic tasks (via the Zoom IN/DOWN widget in Fig. 24) and manual tasks.¹⁹ The tool can be used effectively to get a distribution of the aggregate categories for verbs and nouns in a text or set of texts. In input the script takes a csv file with a list of single words or collocations and in output it creates a two-columns csv file that contains the original nouns/verbs and their aggregate WordNet values, as shown in Fig. 5 below.

We can visualize that information in an Excel bar chart of the WordNet verb categories (Fig. 6).

6 Find the missing character

6.1 The problem

In our collection, each lynching event is typically described in multiple newspaper sources. The different sources often provide different details, different spellings of names, etc. It would be useful to rely on a single summary of an event that takes into account contradictory details; this would allow us to grasp a story in a single reading, and in all its different

¹⁷ The 25 top noun synsets are: act, animal, artifact, attribute, body, cognition, communication, event, feeling, food, group, location, motive, object, person, phenomenon, plant, possession, process, quantity, relation, shape, state, substance, time.

The 15 top verb synsets are: body, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social, stative, weather.

¹⁸ Unfortunately, there is no easy way to aggregate at levels lower than the top synsets. Wordnet is a linked graph where each node is a synset and synsets are interlinked by means of conceptual-semantic and lexical relations. In other words, it is not a simple tree structure: there is no way to tell at which level the synset is located at. For example, the synset “anger” can be traced from top level synset “feeling” and follows the path: feeling—>emotion—>anger. But it can also be traced from top level synset “state” and follows the path: state—>condition—>physiological condition—>arousal—>emotional arousal—>anger. In the first case, “anger” is at level 3 (assuming “feeling” and or other top synsets are level 1). In the second case, “anger” is at level 6. Programmatically, if one gives users more freedom to control the level of aggregating up, it is hard to build a user-friendly communication protocol. If the user wants to aggregate up to level 3 (two levels below the top synset), then should “anger” be considered as a level 3 synset? Does the user want “anger” to be considered as a level 3 synset? Since there is no clear definition of how far away a synset is from the root (top synsets), our algorithm aggregates all the way up to root.

¹⁹ Suppose that you wish to aggregate the verbs in your corpus under the label “violence.” WordNet top synsets for verbs do not include “violence” as a class. Verbs of violence may be listed under body, contact, social. You could use the Zoom IN/DOWN widget of Figure 24 to get a list of verbs in these top synsets, then manually go through the list to select only the verbs of violence of interest. That would mean go through manually the list of 956 verbs in the *body* class (e.g., to find there the verb “attack,” among others), the 2515 verbs of *contact* (e.g., to find there the verb “wrestle”), and the 1688 verbs of *social* (e.g., to find there the verb “abuse”). In total, 5159 distinct verbs. A restricted domain, for example newspaper articles of lynching, may have many fewer distinct verbs, indeed 2027, extracted using the lemma of the POS annotator for all the VB* tags. Whether using the WordNet dictionary (a better solution if the list of verbs of violence has to be used across different corpora) or the POS distinct verb tags, the dictionary list can then be used to annotate the documents in the corpus via the NLP Suite dictionary annotator GUI.

1	Term	WordNet Category	Synset Definition
2	1st_Baron_Beaverbrook	person	British newspaper publisher and politician (born in Canada)
3	1st_Baron_Versulam	person	English statesman and philosopher
4	1st_Earl_Attlee	person	British statesman and leader of the Labour Party who instituted the welfare state in Britain (1883-1967)
5	1st_Earl_Baldwin_of_Bewdley	person	English statesman
6	1st_Earl_of_Balfour	person	English statesman
7	1st_Lieutenant	person	a commissioned officer in the Army or Air Force or Marines ranking above a 2nd lieutenant and below a captain
8	1st_Viscount_Montgomery_of_Alamein	person	English general during World War II
9	2nd_Lieutenant	person	a commissioned officer in the Army or Air Force or Marine Corps holding the lowest rank
10	A._A._Michelson	person	United States physicist (born in Germany) who collaborated with Morley in the Michelson-Morley experiment (1852-1931)
11	A._A._Allie	person	English writer of stories for children (1882-1956)
12	A._Coman_Doyle	person	British author who created Sherlock Holmes (1859-1930)
13	A._E._Burnside	person	United States general in the American Civil War who was defeated by Robert E. Lee at the Battle of Fredericksburg (1824-1881)
14	A._E._Housman	person	English poet (1859-1936)
15	A._E._Kennelly	person	United States electrical engineer noted for his work on the theory of alternating currents
16	A._E._W._Mason	person	English writer (1865-1946)
17	A._Noam_Chomsky	person	United States linguist whose theory of generative grammar redefined the field of linguistics (born 1928)
18	A.E.	person	Irish writer whose pen name was A.E. (1867-1935)
19	a_Kempis	person	German ecclesiastic (1380-1471)
20	Aalto	person	Finnish architect and designer of furniture (1898-1976)
21	Aaron	person	United States professional baseball player who hit more home runs than Babe Ruth (born in 1934)
22	Aaron_Burr	person	United States politician who served as vice president under Jefferson
23	Aaron_Copland	person	United States composer who developed a distinctly American music (1900-1990)
24	Aaron_Montgomery_Ward	person	United States businessman who in 1872 established a successful mail-order business (1843-1913)
25	abandoned_infant	person	a child who has been abandoned and whose parents are unknown
26	abandoned_person	person	someone for whom hope has been abandoned
27	abator	person	a person who abates a nuisance
28	abbe	person	a French abbot
29	abbess	person	the superior of a group of nuns
30	abbot	person	the superior of an abbey of monks
31	Abbott_Lawrence_Lowell	person	United States educator and president of Harvard University (1856-1943)
32	abbreviator	person	one who shortens or abridges or condenses a written work
33	Abdias	person	a Hebrew minor prophet
34	abdicator	person	one who formally relinquishes an office or responsibility
35	abductor	person	someone who unlawfully seizes and detains a victim (usually for ransom)
36	abecedarian	person	a novice learning the rudiments of some subject

Fig. 4 List of entries for top WordNet synset for the noun “person”

nuances. We used the collection of newspaper articles on each event to compile by hand²⁰ often contradictory narratives into a single narrative “compilation”.²¹ Different spellings of names of individuals and locations, different dates and times, different actors and actions were weaved into a single narrative with attribution to a specific source. Thus, for Event ID 261 (the lynching of Grant Welley), all five available newspaper articles (e.g., *The Atlanta Constitution* of 9-4-1900) contain the person’s name as Fleming but the compilation has the name Flemming (2 m’s); the *Thomasville Weekly Times-Enterprise* of 9-8-1900 reports the person’s name as Welly, contrary to all newspaper articles and to the compilation; the same newspaper article of the *Thomasville Weekly Times-Enterprise* reports the location Coolidge, contrary to the location Cooledge found in the other four articles and in the compilation.

The same problem arises for compilers of folk tales, as many different versions of the same traditional tale (e.g., Jacobs’ compilation of English folk tales, 1890 or the Grimm brothers’ collection of German folktales, 2014 [1812, 1857]). Which characters, which characters’ attributes or actions are spurious? The Grimm brothers agonized over these questions for years about each of the tales they had collected. But, returning to our corpus of newspaper articles, we used a basic rule in manually compiling event narratives: all different social actors (individuals, groups, organizations), and all their different actions mentioned in any of the available newspaper sources must be inserted into the compilation. Upon close inspection, though, compilations typically missed some actors and the role they played because of their infrequent occurrence (e.g., church ministers, souvenir hunters, a judge or a white friend of the negro, using the newspaper language of the time, who would address the mob). The problem persisted even after all compilations were inspected and

²⁰ Current computational technology makes available a different approach to creating summaries: an automatic approach where summaries are generated automatically by a computer algorithm, rather than a human (Gambhir and Gupta 2017; Lloret and Palomar 2012; Nenkova and McKeown 2012).

²¹ We use the word “compilation”, rather than “summary”, since, by and large, we maintained the original newspaper language (e.g., the word “negro”, rather than “African American”) and original story line, however contrived the story may have appeared to be.

Fig. 5 The csv output with a list of words categorized in WordNet top 15 verb synsets

	A	B
1	Word	WordNet Category
2	abandon	possession
3	abate	change
4	abdicate	social
5	abduct	contact
6	abet	social
7	abhor	emotion
8	abide	stative
9	absorb	contact
10	abuse	social
11	accede	communication
12	accept	cognition
13	accommodate	stative
14	accompany	stative
15	accomplish	creation
16	accord	stative
17	accost	communication
18	account	stative
19	accrue	change
20	accumulate	possession
21	accuse	communication
22	accustom	change
23	acknowledge	communication
24	acquaint	communication
25	acquiesce	communication
26	acquire	possession
27	acquit	communication
28	act	social
29	actuate	creation
30	adapt	change
31	add	change
32	addict	consumption
33	address	communication
34	adduce	communication
35	adduct	contact
36	adhere	stative

edited by different researchers, multiple times. It became clear that with a few hundred events and several thousand newspaper articles an automatic strategy for checking the reliability of compiled stories was required.

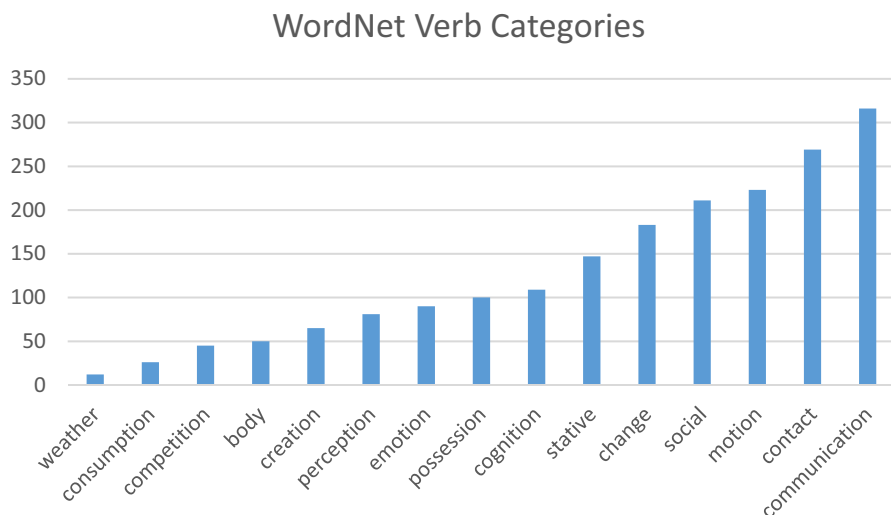


Fig. 6 Excel bar chart of WordNet aggregate verbs in the lynching newspaper articles

6.2 The computational solution

We developed a Python 3 script that takes the list of social actors obtained from WordNet (*Find the character* tool) and checks the values in the list against a list of words found in both summaries and original sources. To obtain the latter list, we use the Stanford CoreNLP²² POS tag annotator to tag each word (token) in both source document and compilation for its Part of Speech value (POSTAG, for nouns, verbs, adjectives, adverbs, etc.). We focus on POS tag values for nouns: NN, noun singular, and NNS, noun plural. For each of these nouns we check whether they are in the WordNet social actor list. Any social actor in a newspaper source not detected in the compilation's social actor list, is flagged as "missing."

We then use another Stanford CoreNLP annotator to tag each word for its NER value (Named Entity Recognition). Of particular interest here are the NER values of Location, Date, Person, Organization.²³ We check each of these NER values in the different sources for each event against the NER values of the compilation. Again, any NER value in a newspaper source not detected in the compilation, is flagged as "missing." The output in csv format lists all cases of error, organized in five rows (one for each type of error: social actor, NER person, organization, date, and location) and six columns: Type of Error, Frequency of Summaries in Error, Percentage of Summaries in Error, Frequency of Error, List of Summary Filenames for Type of Error, List of Documents for Type of Error (see Figs. 7; 8 for a bar chart of the frequency of types of error; the hover-over effects would display all the documents at fault in a specific bar when hovering the bar).²⁴

²² <https://stanfordnlp.github.io/CoreNLP/> Manning et al. (2014).

²³ More specifically, for locations, the NER tags used are: City, State_or_Province, Country. Several other NER values are also recognized and tagged (e.g., Numbers, Percentages, Money, Religion), but they are irrelevant in this context.

²⁴ The column "List of Documents for Type of Error" may be split in several columns depending upon the number of documents found in error.

7 Check the character's name tag

7.1 The problem

When using documents for historical research, whether newspaper articles, police reports, or official institutional documents (e.g., government or business organizations) you can expect misspellings (or, at least, different spellings of the same names but with no *true* yardstick of comparison); misspellings of people's and organizations' names, misspellings of locations. You may encounter a similar problem as a qualitative scholar conducting in depth interviews in a community, accidentally misspelling a name brought up by different individuals. As a literary scholar, you may be interested in checking the spelling of various characters' names in different English translations of the short story "Gentle Breath" by the Soviet writer Ivan Bunin, 1933 Nobel Prize winner in Literature. We already met this problem in finding the missing character in event summaries. Computer algorithms do not deal well with uncertainty and imprecise information (MacEachren et al. 2012; Murchú and Lawless 2014). Whether you have thousands of documents or just a handful, what is the computational solution to make our life easier?

7.2 The computational solution

We take a Python 3 computational approach to misspellings based on Levenshtein's word distance, also known as edit distance (Levenshtein 1966). The algorithm takes as input the NER values of Location, Person, Organization, as computed by Stanford CoreNLP.²⁵ In output, the script produces a csv file that identifies potential typos. Thus, for event ID 2, 'Zed' is identified as a potentially typo, given that it only appears once and in only one document (in the second sentence), compared to 3 times in 2 other documents (Fig. 9, Fig. 10).

8 Find the intruder

8.1 The problem

The over 1,000 newspaper articles on Georgia lynchings were manually grouped together into events, each event characterized by a specific event ID, with each event as a separate folder containing *only* its relevant articles (Figs. 2 and 3 above). Upon close inspection of the articles by lynching events, several errors were discovered, with some newspaper articles "intruding" into the wrong story (i.e., lynching event); put another way, some newspaper articles were stored in the wrong event folder (cross-referencing errors). As a humanist, you are working on the epistolary of an author you are studying. Although the letters are in digitized form, you work by "close reading," reading each letter and classifying and storing the letters in folders by addressee name. But some letters end up in the wrong folder. Is there a way to check automatically for this type of errors; a way to obtain a list of all the newspaper articles that have nothing to do with

²⁵ The algorithm can process all or selected NER values, comparing the associated word values either within a single event subdirectory or across all subdirectories (or all the files listed in a directory, for that matter).

the event to which they are cross-referenced as data sources for the event? A way to find those letters that have nothing to do with the addressee of the folder?

8.2 The computational solution

To understand a possible solution, let's go back to our m subfolders, each subfolder Z containing a variable set j of documents that talk about a specific lynching event. We want to search through all the j documents in each Z to see whether they talk about the same event using a combination of POS tags (NN, noun singular, and NNS, noun plural) and NER tags (Location, Date, Person, and Organization). The logic is similar to the compilation checker (the *Find the Missing Character* tool, except that in this case the script does not check each document against a standard, yardstick, document (the "compilation"). Rather, to identify the intruder, the script checks each of the j documents in an event against each of all other $j-1$ documents in the event. And for each document it builds an index of relativity, based on the cosine similarity formula, to be used comparatively with all the documents in the subfolder (or event cluster).

Two different types of output in csv format are generated: 1. The list of intruders.csv file that lists intruder documents (Fig. 11); 2. The frequency of intruders.csv file lists all documents that do not belong to the group (Fig. 12).

9 Find the character's home

9.1 The problem

Researchers working with documents will no doubt be familiar with the horror stories of misplaced files: fieldnotes from specific informants, in-depth interviews about specific topics, blogs about specific diseases or travel locations, newspaper articles about specific events but where the "specific" is no longer known because the files were accidentally moved from their folders and the cross-reference between document and informant, topic, etc. was lost. With a handful of documents at hand it is certainly easy to read the documents and re-establish the cross-references. But with hundreds or thousands of documents?

9.2 The computational solutions

We propose two solutions to the problem, a naïve solution and a more computing-intensive solution. In both cases, the basic file structure is the same as illustrated in Figs. 1, 2 and 3 above.

1. We have a folder Y containing m subfolders (Z), each subfolder containing a variable set of j documents that talk about event Z ;
2. We also have a folder X containing a list of n unrelated documents each one of which should be in any of the m Z subfolders.

1	*****				
2	Group Identifier: 100				
3	Compilation Filename: 100.txt				
4	Input Documents:				
5	Ashburn Turner County Banner_04-22-1910_1_1.txt				
6	Montgomery Monitor_04-28-1910_3_1.txt				
7	The Atlanta Constitution_04-16-1910_1_1.txt				
8	The Eastman Times-Journal_04-21-1910_1_1.txt				
9	Missing Words	Input Documents			
10	amboy	The Atlanta Constitution_04-16-1910_1_1.txt			
11	body	The Atlanta Constitution_04-16-1910_1_1.txt			
12	charley	Montgomery Monitor_04-28-1910_3_1.txt			
13	citizen	Montgomery Monitor_04-28-1910_3_1.txt			
14	edenfield	The Eastman Times-Journal_04-21-1910_1_1.txt			
15	few	The Atlanta Constitution_04-16-1910_1_1.txt			
16	head	The Eastman Times-Journal_04-21-1910_1_1.txt			
17	hour	Ashburn Turner County Banner_04-22-1910_1_1.txt			
18	life	The Eastman Times-Journal_04-21-1910_1_1.txt			
19	minister	The Eastman Times-Journal_04-21-1910_1_1.txt			
20	murderer	The Eastman Times-Journal_04-21-1910_1_1.txt			
21	official	The Eastman Times-Journal_04-21-1910_1_1.txt			
22	presence	The Eastman Times-Journal_04-21-1910_1_1.txt			
23	saturday	The Atlanta Constitution_04-16-1910_1_1.txt			
24	second	The Eastman Times-Journal_04-21-1910_1_1.txt			
25	stepson	The Eastman Times-Journal_04-21-1910_1_1.txt			
26	thursday	The Atlanta Constitution_04-16-1910_1_1.txt			
27	today	The Eastman Times-Journal_04-21-1910_1_1.txt			
28	tuesday	Ashburn Turner County Banner_04-22-1910_1_1.txt			
29	year	The Atlanta Constitution_04-16-1910_1_1.txt			
30	yesterday	The Atlanta Constitution_04-16-1910_1_1.txt			

Fig. 7 validity_freq.csv file output of frequencies of compilation errors by error type (missing social actors and missing NER values)

9.2.1 Naïve solution

We use the dates embedded in filenames: for each of the n files in X we extract the embedded date (let's call this *source date*) and we then traverse each of the m subfolders Z in Y , extract the date from each of the filenames contained in each subfolder (the *target date*), and check that source and target dates fall within a user-specified range (e.g., 5 days, 7 months, depending upon the temporal nature of the project). If the date comparison fails, we move to the next subfolder until the date comparison passes and we move the file in X currently being processed and containing the source date from X to the subfolder Z . The approach is quite fast and simple but is based on two assumptions: the filenames embed a date (e.g., The New York Times_12-01-1949_2_1); events in each of the m subfolder Z are temporally spaced out (i.e., the dates of the filenames in each of the m Z s are unlikely to be close together).

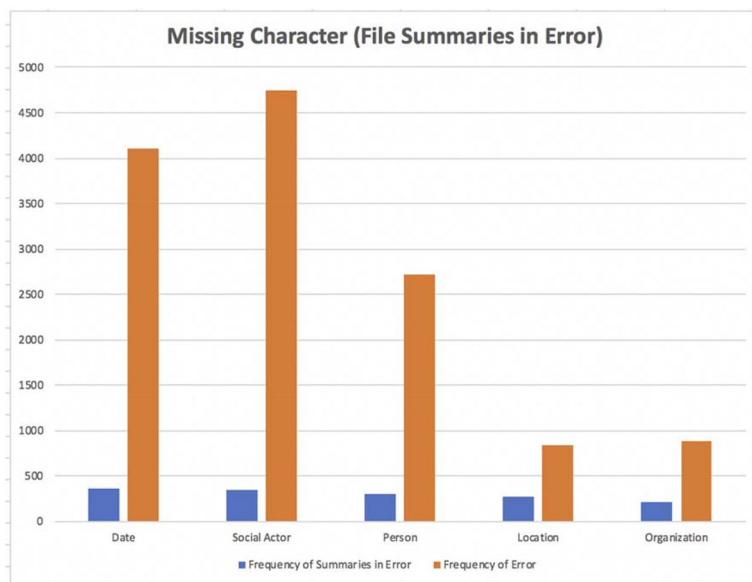


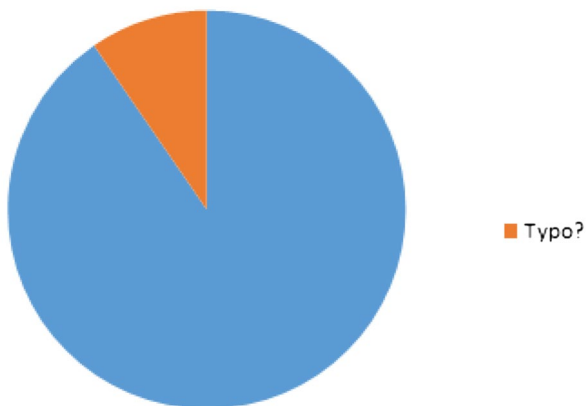
Fig. 8 Column chart of frequencies of compilation errors by error type (missing social actors and missing NER values)

1	Words	Word_fre	SentenceID	DocumentFile_Name	Named_EiSimilar_w	Similar_word_freq	Typo?	Number_of_documents_processed	Processed_dir
8	Zed	1	2	3 The Atlanta Georgian_09-24-1906_1_1.txt	PERSON	Zeb	3 Typo?	3	2
9	Zeb	3	1	1 Atlanta Evening News_09-24-1906_1_1.txt	PERSON			3	2
10	Zeb	3	2	1 Atlanta Evening News_09-24-1906_1_1.txt	PERSON			3	2
11	Zeb	3	4	2 Indiana Evening Gazette_09-24-1906_1_1.txt	PERSON			3	2
12	Zaidee	1	12	2 Montgomery Monitor_11-28-1895_2_1.txt	PERSON			1	37
13	Zack	1	27	3 The Augusta Chronicle_01-26-1910_7_1.txt	PERSON			0	33

Fig. 9 Screenshot of csv output for Levenshtein word/edit distance

Fig. 10 Pie chart of predicted typos

Percentage of Predicted Typos in Corpus



Group identifier (folder name)	Documents in group	Intruder document	Group path	Document path	Similarity index (< 2)	
2	100	4 The Eastman Times Journal, 04-23-1902_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.13217913	*****	
3	101	2 The Eastman Times Journal, 05-17-1904_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.18947664	*****	
4	102	No document in the input folder				
5	104	3 The Mason Telegraph, 06-18-1895_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.14543108	*****	
6	106	3 The Eastman Times Journal, 07-22-1895_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.15512944	*****	
7	108	3 The Mason Telegraph, 08-24-1895_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.1875	*****	
8	107	4 The Eastman Times Journal, 08-28-1897_4_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0	*****	
9	109	6 The Mason Daily Telegraph, 09-26-1902_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.14860469	*****	
10	112	2 The Eastman Times Journal, 09-09-1892_2_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.13224889	*****	
11	113	3 The Savannah Morning News, 09-26-1893_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.1	*****	
12	114	4 Chattanooga Daily Times, 12-12-1895_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.0903226	*****	
13	115	3 Benton Star, 05-02-1901_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.12542373	*****	
14	122	3 The Atlanta Constitution, 07-10-1900_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.17430396	*****	
15	129	2 The Atlanta Constitution, 08-20-1914_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.07811373	*****	
16	131	4 The Atlanta Constitution, 08-24-1909_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.18344004	*****	
17	130	2 Monroe Advertiser, 09-14-1906_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.10256403	*****	
18	130	3 The Mason Daily Telegraph, 09-15-1908_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.13313333	*****	
19	134	2 The Atlanta Constitution, 04-14-1902_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.09039216	*****	
20	134978	2 Thomas Times, 06-24-1905_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.1875	*****	
21	21	2 The Mason News, 11-09-1910_3_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.10734257	*****	
22	136158	No document in the input folder				
23	137	2 The Atlanta Constitution, 12-02-1919_21_1.txt	C:\Users\Franco\Documents\My Projects\C:\Users\Franco\Documents\My Projects\Emory\I	0.18181818	*****	

Fig. 11 Screenshot of csv output listing intruder documents

1	Similarity index cutoff	Number of intruders	Percentage of intruders among all documents for all groups (folders)	Number of intruded groups (folders)	Percentage of intruded groups (folders)	List of intruded groups (folders)	List of intruded documents
2	0.2	151	13.49	118	30.73	100; 101; 104; 106; 107; 108; 11 The Eastman Times-Journal, 04-23-1902_3_1.txt; The Eastman	

Fig. 12 Screenshot of csv output listing frequency and percentage of intruder documents by group

9.2.2 Computing-intensive solution

The solution is similar to the one adopted for *Find the Intruder*. We use a combination of social actors taken from WordNet and Stanford CoreNLP POS and NER tag annotators (POS tags NN, noun singular, and NNS, noun plural; NER tags Location, Date, Person, Organization). Using these POS and NER tag values we build relativity indices based on cosine similarity.²⁶ Any of the n unrelated documents with the highest relativity index higher than the user-selected threshold with any of the m subfolders (Z) will be signalled as belonging to that subfolder: a home is found for the stray document (see Figs. 13 and 14²⁷ for the output). With a large number of unclassified source/input documents and target/output directories and subdirectories, and a default value of 0.25 for the relativity index threshold, the script may lead to a number of source documents classified in several output folders. Such cases are classified as Repeated under the Outcome column in the output csv file. Raising the threshold will reduce repeated cases but may miss genuine cases. To help the user, the output csv file will mark with an * under the column “Highest index” the highest value in the group of repeated targets (each group of repeated targets for the same source are conveniently separated by *****). Most likely, that would be the best target. Since all Source document

²⁶ We calculated the relativity index by using cosine similarity (Singhal 2001). We use the two list of NN, NNS, Location, Date, Person, and Organization from the j doc ($L1$) and from all other $j-1$ docs ($L2$) and compute cosine similarity between the two lists. We construct a vector from each list by mapping the word count onto each unique word. Then, relativity index is calculated as the cosine similarity between two vectors and n is the count of total unique words. For instance, $L1$ is [Alice: 2, doctor: 3, hospital: 1], and $L2$ is [Bob: 1, hospital: 2]. If we fix the order of all words as {Alice, doctor, hospital, Bob}, then the first vector ($V1$) is (2, 3, 1, 0), the second vector ($V2$) is (0, 0, 2, 1), and the length n of the vector is 4. The relativity is the dot product of two vectors divided by the product of two vector lengths. Documents with index of relativity significantly lower than the rest of the cluster are signalled as unlikely to belong to the cluster.

$$\text{relativity index} = \frac{\sum_{i=1}^n (V1_i V2_i)}{\sqrt{\sum_{i=1}^n V1_i^2} \sqrt{\sum_{i=1}^n V2_i^2}}$$

The relativity index ranges from 0 to 1, where 0 means two documents are totally different, and 1 means two documents have exactly the same list of NN, NNS, Location, Date, Person, and Organization.

²⁷ The bar chart displays the distribution of most frequent threshold index values as intervals, with most records in the 0.25~0.29 interval.

and Target directory entries contain hyperlinks, a user can easily check the accuracy of the script's classification.

10 Find the plagiarist

10.1 The problem

A “close reading” of at least some of the thousands of newspaper articles on Georgia lynchings makes it apparent that many of these articles published in newspapers across America were “plagiarized” copies of a single story for the same event. How many of the available newspaper articles were basic copies of each other? Was one newspaper typically more involved than others in providing the basic story line to the others? And when newspaper articles were plagiarized, was the copy exactly the same as the original or was only a part of the original plagiarized in the copy? Was there a difference in story line between newspapers published in former confederate and union states? Did the plagiarism practice decline over time?²⁸

10.2 The computational solution

To find an answer to these questions, we developed a computational tool written in Java based on the Apache Lucene library (Apache Lucene 7.7.1, released on March 1st, 2019).²⁹

³⁰ Apache Lucene is a Java open source software library for indexing and searching text by the Apache Software Foundation. As we read in McCandless et al. (2010) “When Lucene first hit the scene five years ago, it was nothing short of amazing,” as an incredibly powerful search engine. Ten years on and Lucene is now used “in diverse companies including

²⁸ It should be noted that the use of the words *plagiarism* and *plagiarist* in this context should be taken with a grain of salt. First, the data do not tell us anything about who copied whom, but only that the two different newspapers shared content, wholly or in part; furthermore, the shared content may well have come from an unacknowledged wire service (on the development and spread of news wire services in the United States during the second half of the nineteenth century, see Brooker-Gross 1981; on computational tools for plagiarism and authorship attribution, see, for instance, Stein et al. 2011).

²⁹ <http://lucene.apache.org/core/downloads.html>. For a summary of approaches to document similarities, see Forsyth and Sharoff (2014).

³⁰ Other approaches are also available. After all, determining document similarity has been a major research area due to its wide application in information retrieval, document clustering, machine translation, etc. Existing approaches to determine document similarity can be grouped into two categories: knowledge-based similarity and content-based similarity (Benedetti et al., 2019).

Knowledge-based similarity approaches extract information from other sources to supplement the corpus, so as to draw from more document features to analyze. For example, Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch 2007) represents documents in high dimensional vectors based on the features extracted from both original articles and Wikipedia articles. Then, similarity of documents is calculated using vector space comparison algorithm. Since our main focus in this work is to detect plagiarism among texts in the same corpus, knowledge-based similarity approaches are not very fruitful.

Content-based similarity approaches focus on using only textual information contained in documents. Popular proposed techniques in this fields are Vector Space Models (Turney and Pantel 2010), probabilistic models such as Okapi BM-25 (Robertson and Zaragoza 2009). These methods all transform documents into some form of representations, and then either do a vector space comparison or query search match on the constructed representations.

1	Source document	Target directory	Highest index	Relativity index (>0.25)	Outcome
2	C:/Users/AAlbany Weekly Herald_03-16-1901_1_1.word.txt	C:/Users/Lynching/Georgia/151		0.26 Repeated	*****
3	C:/Users/txt files/Albany Weekly Herald_03-16-1901_1_1.word.txt	C:/Users/Lynching/Georgia/173		0.3 Repeated	
4	C:/Users/txt files/Albany Weekly Herald_03-16-1901_1_1.word.txt	C:/Users/Lynching/Georgia/2		0.25 Repeated	
5	C:/Users/txt files/Albany Weekly Herald_03-16-1901_1_1.word.txt	C:/Users/Lynching/Georgia/239		0.26 Repeated	
6	C:/Users/txt files/Albany Weekly Herald_03-16-1901_1_1.word.txt	C:/Users/Lynching/Georgia/330		0.27 Repeated	
7	C:/Users/txt files/Albany Weekly Herald_03-16-1901_1_1.word.txt	C:/Users/Lynching/Georgia/348		0.35 Repeated	
8	C:/Users/txt files/Albany Weekly Herald_03-16-1901_1_1.word.txt	C:/Users/Lynching/Georgia/8	*	0.82 Repeated	
9	C:/Users/txt files/Athens Banner_04-22-1921_1_1.txt	C:/Users/Lynching/Georgia/250		0.27 Repeated	*****
10	C:/Users/txt files/Athens Banner_04-22-1921_1_1.txt	C:/Users/Lynching/Georgia/73	*	0.58 Repeated	
11	C:/Users/txt files/Athens Weekly Banner_03-25-1898_1_4.txt	C:/Users/Lynching/Georgia/136122	*	0.81 Repeated	*****
12	C:/Users/txt files/Athens Weekly Banner_03-25-1898_1_4.txt	C:/Users/Lynching/Georgia/216		0.25 Repeated	
13	C:/Users/txt files/Athens Weekly Banner_03-25-1898_1_4.txt	C:/Users/Lynching/Georgia/225		0.28 Repeated	
14	C:/Users/txt files/Athens Weekly Banner_03-25-1898_1_4.txt	C:/Users/Lynching/Georgia/371		0.29 Repeated	

Fig. 13 Screenshot of csv output listing documents classified by NER values

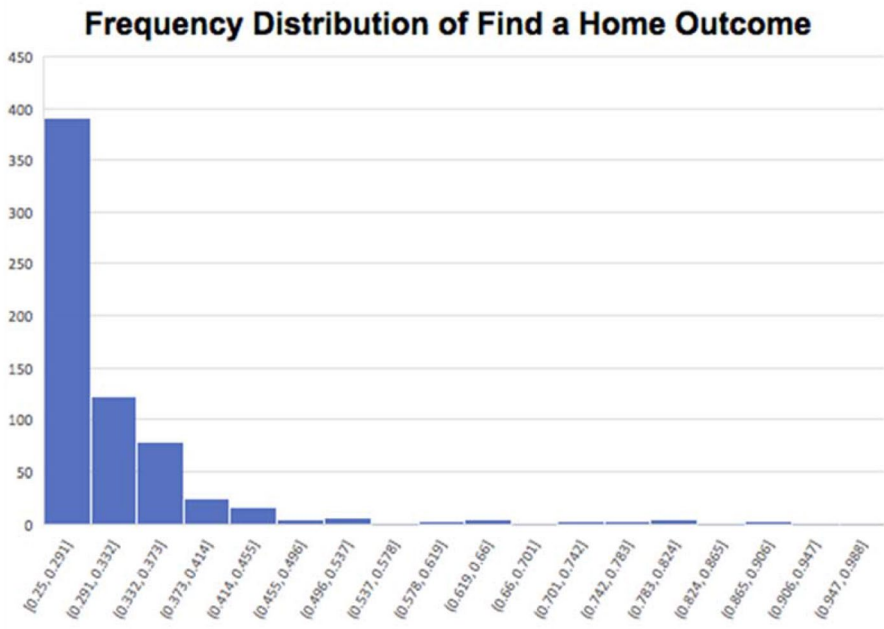


Fig. 14 Bar chart of document classification by NER values in intervals of most frequent threshold index values

Akamai, Netflix, LinkedIn, Technorati, HotJobs, Epiphany, FedEx, Mayo Clinic, MIT, New Scientist Magazine, and many others.”

Lucene has been most used as a search engine for “performing common search and search related tasks like indexing, querying, highlighting, language analysis and many others” (Bialecki et al. 2012). As such, Lucene is “an excellent full-text search engine tool package” that can be used to do text analysis (Zhang and Li 2009). In our application of Lucene, we extend Lucene’s indexing and querying functions to find representative terms of each document. This is in line with other similar uses of Lucene. Cooper et al. (2002) write: “We obtain a ranked list of the most important terms in each document using a rapid phrase recognizer system. We store these in a database and compute document similarity using a simple database query” (Cooper et al. 2002). Similarly, Borg et al. (2014) used Lucene to create “a document which consists of fields, that represent the defect report” and

used query library to get a “ranked list of potential duplicate defect reports” (Borg et al. 2014).

The tool has two components: first, we index all files in an input directory; then, we loop through the files in that directory to find duplicates in file content, in at least some part of their content. The user can vary the similarity index. We use a similarity index of 0.8. In output, the routine produces several files. A text file³¹ (Lucene_document_duplicates.txt) contains information about duplicates: the document with duplicates in the directory, listing representative terms, i.e., those words that best distinguish document (e.g., The Savannah Morning News_04_27_1901_1_1.txt) from other documents (e.g., The Dublin Courier-Dispatch_05_02_1901_1_1.txt) (see Fig. 15).

Four further csv files are exported in output to provide frequency measures of the phenomenon of “plagiarism” by classes of involvement (0–10%, 10–20%, 20–30%,...) with a listing of documents in each class (File 1, Lucene_classes_freq.csv), by classes of involvement over time³² (if filenames embed dates) (File 2, Lucene_classes_time_freq.csv), by extent of involvement for each specific document (File 3, Lucene_document_instance_classes_freq.csv), and by classes of involvement for every document (File 4, Lucene_document_name_classes_freq.csv).³³

Figure 16 provides an example of File 1. The frequencies refer to the number of pairs of documents that fall within the specified percentage duplication. Thus, 857,592 pairs of documents are within 0–10% duplication. The List of Documents in Category column displays all such pairs. Thus, “The Atlanta Constitution_09-29-1922_7_1.txt” and “The Macon Telegraph_02-11-1899_1_1.txt” are within 0–10% duplicate of each other.

File 2 provides the distribution by classes of percentage duplication by year (in this specific project, between 1875 and 1935) (Fig. 17).

Percentages in each class are computed with the numerator as the number of documents in the year that fall within the specified duplication range and with the denominator as the total number of documents for the specific year. Thus, for 1876, 96.59% of our documents have less than 10% duplication with other documents, and the rest of them have under 20% duplication with other documents (percentages in each row add up to 1).

File 3 provides, for each specific document, the frequency of other documents that fall within the specified duplication range (Fig. 18).

Thus, “The Atlanta Constitution_10-12-1908_1_1.txt” has under 10% duplication score with 1007 other documents, 10–20% duplication score with 37 other documents and under 20–30% duplication score with 1 other document (Fig. 19).³⁴

Output data are also visualized in Excel pie charts and line charts (if filenames embed dates). The pie chart of Fig. 20 shows that the 0–10% plagiarism class takes up ~96% of document pairs, i.e., most articles do not plagiarize each other, the 10–20% plagiarism

³¹ document_duplicates.txt.

³² Users can specify different spans of temporal aggregation (e.g., year, quarter/year, month/year).

³³ In this specific application, documents are newspapers where *document name* refers to the name of the paper (e.g., The New York Times) and *document instance* refers to a specific newspaper article (e.g., The New York Times_12-11-1912_1, referring to a The New York Times of December 11, 1912 on page 1). But the *document name* could refer to an ethnographic interview with *document instance* referring to an interviewer’s ID (by name or number), an interview’s location, time, or interviewee (by name or ID number).

³⁴ The numbers in each row of the table add up to *approximately* the total number of newspaper articles in the corpus. This number is not exact due to the way the Lucene function “find top similar documents” computes similar documents with discrepancies numbering in the teens.

class takes up ~4% of document pairs. The remaining are all very small numbers that round to ~0% as shown in the zoom in display.

The line chart of Fig. 21 shows the temporal dynamics of “plagiarism” in the 0–10% frequency class.

The bar chart of Fig. 22 visualizes a subset of 50 documents for better readability. For each document, it presents 10 bars corresponding to 10 frequency classes. It shows that different newspapers have different involvement in “plagiarism”. For most newspapers, the 0–10% “plagiarism” rate class (blue bar) dominates; the 10–20% “plagiarism” rate class (red bar) is comparatively much smaller; and all the other 8 bars for each newspaper are not even visible in the chart as the frequencies are too close to zero.

10.2.1 Beyond social movements research

Relying on a large set of newspaper documents we have illustrated the power of a set of computational tools that can be profitably used by scholars in the humanities and social sciences. The documents describe lynching events in Georgia (1875–1935). Both data source—newspapers—and topic—group violence—have traditionally come under the purview of social movement, protest event, and violence research by both qualitative and quantitative historians and social scientists.³⁵ But the tools have much wider applications. In describing each tool, we have highlighted how a qualitative ethnographer or literary critic may profit from the NLP tools for their specific needs. For as long as you work with text as data, the algorithms may have something useful for you. If filenames embed meta-data (e.g., a date, location, interviewee name in interviews) checking the consistent use of metadata fields is a general problem beyond social movement research; and so is the algorithm that checks minor spelling differences of the same names across different documents. If as a literary critic you are working on the epistolary of an author, in how many letters does your author copy and paste text, repeating the same information for different addressees? The plagiarism tool is similarly independent of topic.

But regardless of methodological/disciplinary identity and documents content—it may be daunting to approach a methodology that comes with such keywords as computational linguistics or Natural Language Processing. You have less than a hundred documents, you might as well analyze them by hand (“close reading”), regardless of problem. Fair decision. Nearly all computational algorithms described in scholarly journals do indeed require a computer science background to run them even when they are open-source and freely available. And when tools are used in a “pipeline”—the output of one tool becoming the input to the next tool in a pipeline—using the algorithms present serious challenges. But the tools described here are part of a set of NLP tools, freely available on GitHub (<https://github.com/NLP-Suite/NLP-Suite/wiki>), designed for minimal start-up costs (e.g., auto-installation of all the required software) and for easiness of use (e.g., algorithms come with user friendly Graphical User Interface (GUI), automatic reminders, TIPS files, HELP buttons).

³⁵ On the specific topic of lynching, see, for instance, the quantitative work by Beck and Tolnay (1990) or Franzosi et al. (2012) and the more qualitative work by Brundage (1993).


```

1 The document that is copied most is: The Savannah Morning News_04-27-1901_1_1.txt. It is copied 2 times
2 46 articles have copies
3
4 The Savannah Morning News_04-27-1901_1_1.txt has 2 copy(s):
5 Representative Terms: set, offered, months, april, william, citizens, developed,
6 scoundrel, free, premises, elberton, widowed, elbert, reported, fishing, wor
7 kman, rhoda, miss, thrown, mother, appeared, goolsby, savannah, alexander, ri
8 ver
9
10 1. Title: Dublin Courier-Dispatch_05-02-1901_1_1.txt
11 Score (% match with representative terms): 88.3%
12 Matching Terms: set, offered, months, william, citizens, developed, free, premises,
13 elberton, widowed, reported, fishing, workman, rhoda, miss, thrown,
14 mother, appeared, goolsby, savannah, alexander, river
15
16 2. Title: The Atlanta Constitution_04-27-1901_2_1.txt
17 Score (% match with representative terms): 86.27%
18 Matching Terms: offered, months, april, william, citizens, developed, premises,
19 elberton, widowed, elbert, reported, fishing, workman, rhoda, miss, thrown,
20 mother, appeared, goolsby, savannah, alexander, river
21
22
23 The Atlanta Constitution_06-10-1890_1_1.txt has 1 copy(s):
24 Representative Terms: sunday, upstairs, found, security, drowning, complained, default,
25 tied, masked, body, evening, unsatisfactory, schevenel, decayed, forty,
26 deceased, hall, bond, resides, river, hall's, evidence, pound, elberton
27
28 1. Title: Hartwell Sun_06-13-1890_3_1.txt
29 Score (% match with representative terms): 90.2%
30 Matching Terms: sunday, upstairs, found, security, drowning, complained, default,
31 tied, masked, body, evening, schevenel, forty, deceased, hall, bond, resides,
32 river, hall's, evidence, pound, elberton

```

Fig. 15 Screenshot of the txt output file listing plagiarized documents

1	Classes of Percentage Duplication	Frequency	List of Documents in Category
2	0-10%	857592	The Atlanta Constitution_09-29-1922_7_1.txt - The Macon Telegraph_02-11-1899_1_1.txt, The.
3	10-20%	29861	The Frederick News_09-02-1902_2_1.txt - The Bainbridge Democrat_09-06-1888_3_1.txt, Sprin
4	20-30%	1612	Vidalia Advance_07-31-1930_10_1.txt - Vidalia Advance_08-07-1930_4_1.txt, Brunswick Times,
5	30-40%	511	The Augusta Chronicle_03-05-1909_1_1.txt - The Atlanta Constitution_03-05-1909_11_1.txt, Th
6	40-50%	277	The Macon News_05-18-1918_1_1.txt - Cordele Dispatch_05-19-1918_1_1.txt, The Atlanta Con:
7	50-60%	149	The Atlanta Constitution_07-28-1913_1_1.txt - The Abbeville Chronicle_07-31-1913_1_1.txt, Th
8	60-70%	78	The Atlanta Constitution_07-07-1915_3_1.txt - The Albany Herald_07-07-1915_1_1.txt, The Atl
9	70-80%	46	The Atlanta Constitution_05-16-1919_16_1.txt - The Macon Daily Telegraph_05-16-1919_1_1.tx
10	80-90%	44	The Savannah Morning News_07-01-1896_1_1.txt - The Quitman Free Press_07-04-1896_1_1.tx
11	90-100%	42	The Savannah Morning News_03-01-1901_10_1.txt - The Atlanta Constitution_03-01-1901_2_1.

Fig. 16 Screenshot of the csv output file (File 1) listing plagiarized documents by classes of involvement

1	Year	0-10%	10-20%	20-30%	30-40%	40-50%	50-60%	60-70%	70-80%	80-90%	90-100%
2	1876	0.97	0.03	0	0	0	0	0	0	0	0
3	1877	0.93	0.06	0	0	0	0	0	0	0	0
4	1878	0.99	0.01	0	0	0	0	0	0	0	0
5	1879	1	0	0	0	0	0	0	0	0	0
6	1880	0.94	0.06	0	0	0	0	0	0	0	0
7	1881	0.99	0.01	0	0	0	0	0	0	0	0
8	1882	0.97	0.03	0	0	0	0	0	0	0	0
9	1883	0.99	0.01	0	0	0	0	0	0	0	0
10	1884	0.97	0.03	0	0	0	0	0	0	0	0
11	1885	0.94	0.06	0	0	0	0	0	0	0	0
12	1886	0.97	0.03	0	0	0	0	0	0	0	0
13	1887	0.95	0.05	0	0	0	0	0	0	0	0

Fig. 17 Screenshot of the csv output file (File 2) of plagiarism by classes of involvement over time

1	File Name	0-10%	10-20%	20-30%	30-40%	40-50%	50-60%	60-70%	70-80%	80-90%	90-100%
2	The Atlanta Constitution_10-12-1908_1_1.txt	1007	37	1	0	0	0	0	0	0	0
3	The Atlanta Constitution_05-08-1914_9_1.txt	987	7	1	0	0	0	0	0	0	0
4	Dawson News_11-06-1901_2_1.txt	854	15	0	2	0	0	0	0	0	0
5	The Atlanta Constitution_06-17-1899_2_1.txt	993	8	0	0	0	1	0	0	0	0
6	Americus Times-Recorder_08-10-1898_1_1.txt	822	29	2	0	0	2	0	0	0	0
7	The Atlanta Constitution_08-29-1909_5_1.txt	971	79	4	0	0	0	0	0	0	0
8	The Moultrie Observer_07-01-1922_1_1.txt	942	14	0	1	0	1	0	0	0	0
9	Monroe Advertiser_09-14-1906_1_1.txt	910	8	0	0	0	0	0	0	0	0
10	The Atlanta Constitution_05-19-1922_20_1.txt	1024	15	0	0	1	1	0	0	0	0
11	Columbus Daily Enquirer-Sun_07-23-1884_4_1.txt	547	3	0	0	0	0	0	0	0	0
12	The Athens Weekly Banner_06-17-1890_6_1.txt	991	3	1	0	0	0	0	0	0	0
13	The Atlanta Constitution_02-23-1916_11_1.txt	850	1	2	1	0	0	0	0	0	0
14	Early County News_06-16-1921_1_1.txt	983	32	4	1	0	1	0	0	0	0
15	The Quitman Free Press_06-17-1921_1_1.txt	840	109	4	3	0	1	0	0	0	0
16	The Atlanta Constitution_10-20-1888_2_1.txt	778	71	2	0	0	0	0	0	0	0
17	Moultrie Daily Observer_07-02-1909_1_1.txt	895	8	1	2	0	0	0	0	0	0

Fig. 18 Screenshot of the csv output file (File 3) of plagiarism by classes of involvement for every specific document

11 Limitations

The set of tools described here relies on a number of external freeware packages that we adapted to help non-expert users with issues of data reliability. None of these packages are 100% accurate, from Stanford CoreNLP POS and NER annotators to Lucene or Levenshtein's word/edit distance. We tested each tool extensively, in an iterative process, at each new iteration manually inspecting the output against input and making changes to the algorithm until finally satisfied with performance. We should, nonetheless, point out some of the limitations of the tools.

First, in designing the tools, we did not aim for scalability and speed. We designed the algorithms for the typical humanity or social science use with small/medium size datasets, typically hand coded, rather than big data, where documents number in the millions. Speed is less of an issue in such cases, particularly when considering that these tools would be used only once or twice in a project.

Second, the applications of the tools assume that the documents have been assembled into clusters, each cluster describing an event, a story, or a unique entity, and where these clusters in a computer environment correspond to different folders (see Figs. 2 and 3 above). That classification of documents into larger clusters can take different forms, from a cluster ID number or topic name embedded in the filename and separated by __ (e.g., The Atlanta Constitution_05-24-1918_2_1__3, The Atlanta Constitution_05-24-1918_2_1__Jim Cobb). Then, all documents describing event 3 or Jim Cobb would all share that same unique identifier (__3 or __Jim Cobb). Alternatively, those same documents describing the same event could be grouped together under different subfolders, regardless of file name embeddings. For simplicity, we assume the second type of approach with events embedded in the directory structure rather than filename. The tools are designed to deal with possible mishandling of documents *within* that type of research design (e.g., documents downloaded or moved through cut and paste to the wrong subfolder, i.e., event cluster). A different problem would be if we had thousands of documents that need to be classified into different events. There are many open source algorithms that deal with problems of text classification, cutting-edge algorithms typically based on neural network approaches (for recent surveys, see Aggarwal and Zhai 2012; Kowsari et al. 2019).

Third, the tools work for documents in the English language; our reliance on WordNet and Stanford CoreNLP imposes that limitation, in particular WordNet which is solely based on the English language while Stanford CoreNLP supports several different languages,

	Newspaper Name	0-10%	10-20%	20-30%	30-40%	40-50%	50-60%	60-70%	70-80%	80-90%	90-100%
1	Albany Daily Herald	953	154	2	0	0	0	0	0	0	0
2	Albany Weekly Herald	671	7	0	0	0	0	0	0	0	0
3	Americus Times-Recorder	748	22	1	0	0	0	0	0	0	0
4	Americus Weekly Times-Recorder	950	135	3	0	0	0	0	0	0	0
5	Ashburn Turner County Banner	758	1	0	0	0	0	0	0	0	0
6	Athens Weekly Banner	543	8	0	0	0	0	0	0	0	0
7	Atlanta Daily World	806	21	1	0	0	0	0	0	0	0
8	Atlanta Evening News	853	4	1	0	0	0	0	0	0	0
9	Augusta Chronicle	889	50	1	0	0	0	0	0	0	0
10	Augusta Chronicle and Constitutionalist	642	15	0	0	0	0	0	0	0	0
11	Bainbridge Search Light	864	4	0	0	0	0	0	0	0	0
12	Bainbridge Weekly Argus	1049	10	0	1	0	0	0	0	0	0
13	Bainbridge Weekly Democrat	539	4	0	1	0	1	0	0	0	0
14	Baltimore Afro-American	886	7	0	0	0	0	0	0	0	0

Fig. 19 Screenshot of the csv output file (File 4) of plagiarism by classes of involvement for every document

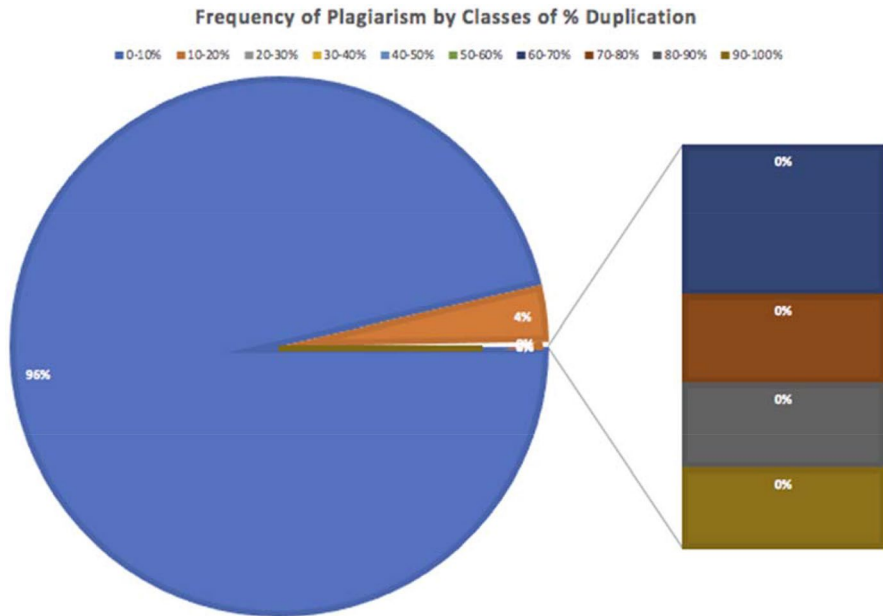


Fig. 20 Pie chart of frequencies of “plagiarism”

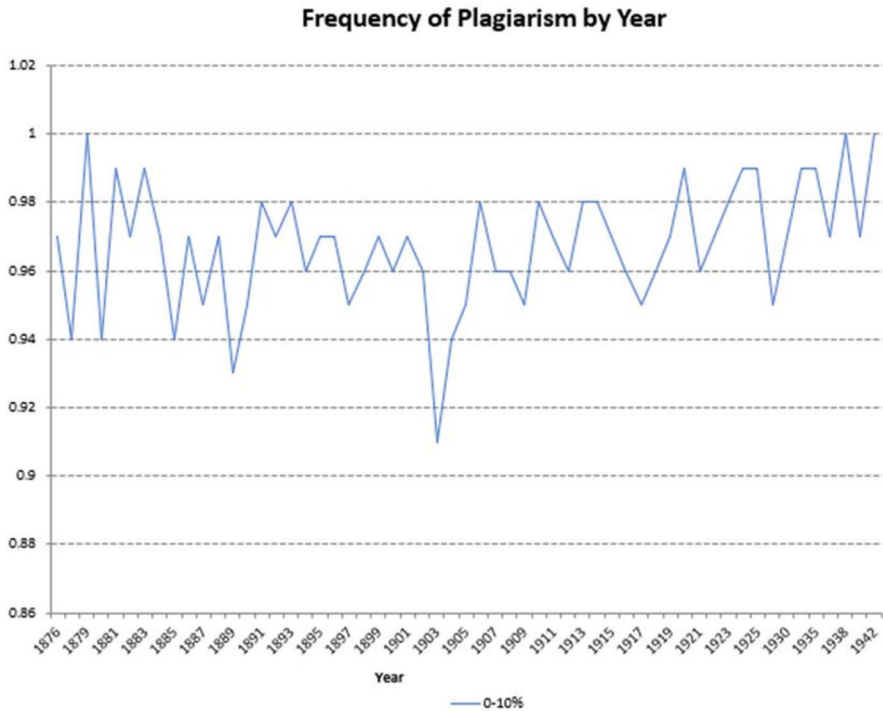


Fig. 21 Line chart of frequencies of “plagiarism” by year in the 0–10% class

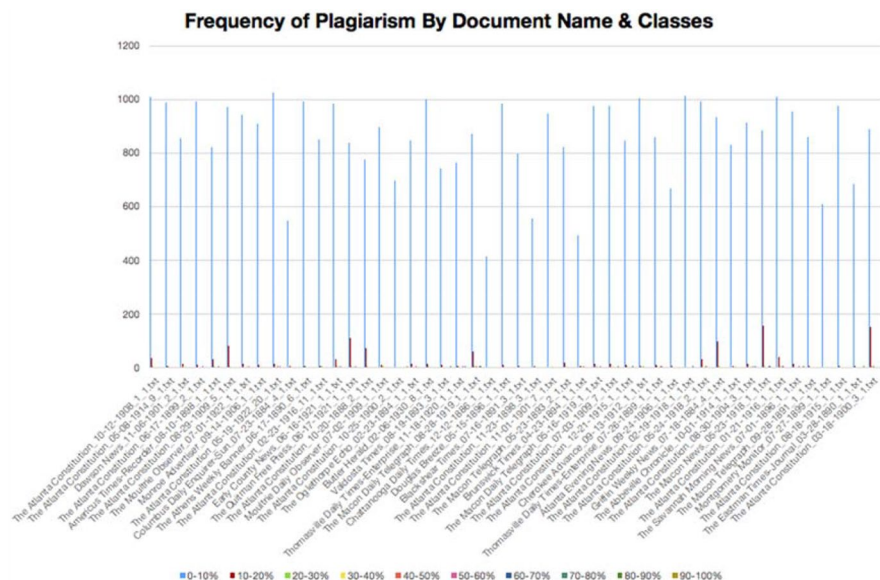


Fig. 22 Bar charts of frequencies of “plagiarism” for a sample of 50 newspapers by classes of plagiarism

although appropriate language packs must be downloaded (Arabic, Chinese, French, German, Spanish). To the extent that our tools are freeware and opensource, they can be potentially modified for other languages.

Fourth, WordNet does not provide easy and consistent ways of aggregating nouns and verbs at levels other than the top synsets, given the linked graph design of the database. Although some solutions for alternative ways of aggregating are available, as discussed in the paper, these involve a combination of automatic and manual work.

Finally, the tools focus on factual aspects of language use: social actors (or agents, more generally), date/time, location, names of individuals and of organizations (with an implicit assumption that the documents tell stories: Who (social actors, individuals and organizations), What, When, and Where. The tools are ill suited to identify the evaluative aspect of language (see Labov 1972: 366–375). Such expressions found in many newspaper articles on lynching would escape detection: “*All negros who behave this way deserve to be lynched;*” “*He met his just death;*” “*While one should let the law take its course, lynching is necessary to deal with this unspeakable crime.*”

12 Conclusions

In this paper, we have described a set of computational tools, written in both Java and Python 3, that can be profitably used for *automatic* checking of data reliability. The tools are particularly useful in social movements research where multiple descriptions of the same events are often available from different sources (e.g., newspapers, police, NGOs). But field researchers could also benefit from using these tools to the extent that different informants may provide different accounts of the same event. More specifically, the following set of different tools can be used for different social-science purposes.

1. *Check the filename well-formedness*—The different Python functions provide several ways of checking the well-formedness of filenames when filenames embed structured data (e.g., the newspaper name, date, page and column numbers, and perhaps event ID or event identifying name, e.g., The Atlanta Constitution_05-24-1918_2_1__3, The Atlanta Constitution_05-24-1918_2_1__Jim Cobb).
2. *Find the character and the ancestor* (via WordNet)—The tool relies on the lexicon database WordNet in order to
 - a. get lists of terms found in the English language and belonging to specific word classes (e.g., all animals, all persons);
 - b. aggregate nouns or verbs into higher-level aggregates (e.g., such verbs as ‘run,’ ‘walk,’ ‘strut,’ ‘swagger’ as verbs of ‘motion’).
3. *Find the missing character*—The script will check the reliability of the summary of an event obtained from different source documents; it addresses such questions as: does the summary correctly reflect what everyone did or said? Does it contain all the different spellings of the names of individuals, organizations, or locations found in the various source documents? Does it contain a list of all the filenames of the document used for the summary?
4. *Find the character’s name tag*—Is the name of Peter Stamps always spelled that way in all the articles dealing with the case (e.g., The Atlanta Constitution_05-24-1918_2_1__Jim Cobb)? Or is Stamps sometimes spelled Stamp, Stumps? The algorithm relies on Levenshtein’s word/edit distance to find word similarities and potential misspellings as present in the original document or due to transcription errors.
5. *Find the intruder*—You have organized your documents by event, each event with its own set of documents; and yet, some documents may have nothing to do with the event, the result of a wrong file naming when filenames embed the event ID number or name or a file download (or cut and paste) in the wrong folder. With thousands of documents and events, a tool that allows you to automatically check whether a document belongs to the correct event (or topic) would be of great help. Such is the Find the intruder tool.
6. *Find the character’s home*—When a set of files are listed in a directory without knowing which event cluster they belong to either via an event ID number or an event identifying name embedded in the filename (e.g., The Atlanta Constitution_05-24-1918_2_1__3, The Atlanta Constitution_05-24-1918_2_1__Jim Cobb), the tools provide different ways of classifying documents in the most appropriate cluster.
7. *Find the plagiarist*—The tool can be used to find an answer to the following question: how many documents are copies of each other and to what extent? You may think that you are relying on multiple sources of evidence for your analyses, lending greater credibility to your conclusions, but are these conclusions based on one single document whose content was copied many times over in other documents?

Appendix

See Figs. 23, 24 and 25

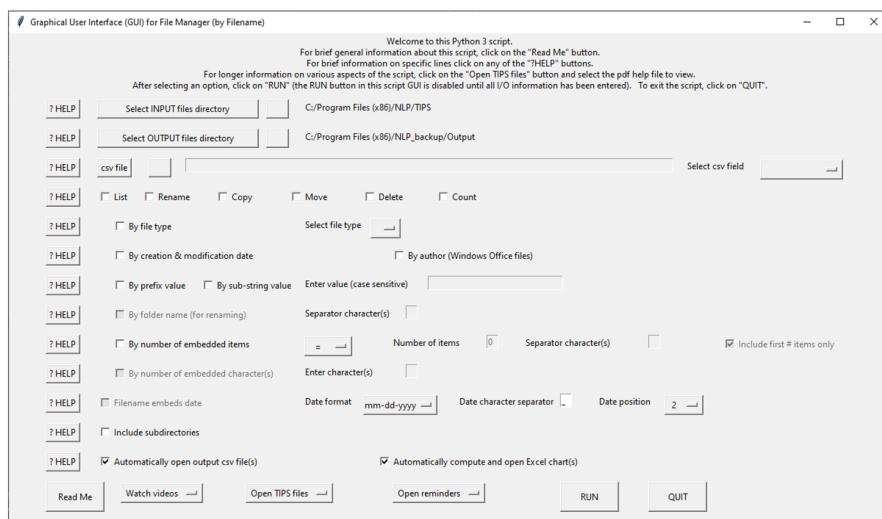


Fig. 23 Screenshot of the Graphical User Interface (GUI) for the filename checker

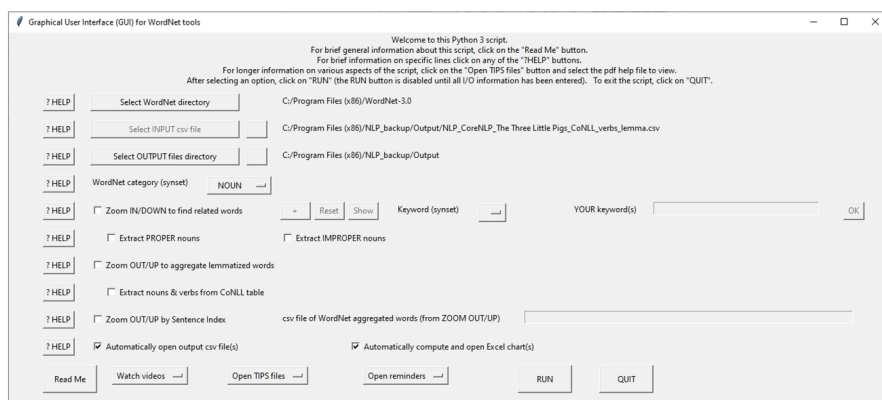


Fig. 24 Graphical User Interface (GUI) for WordNet options

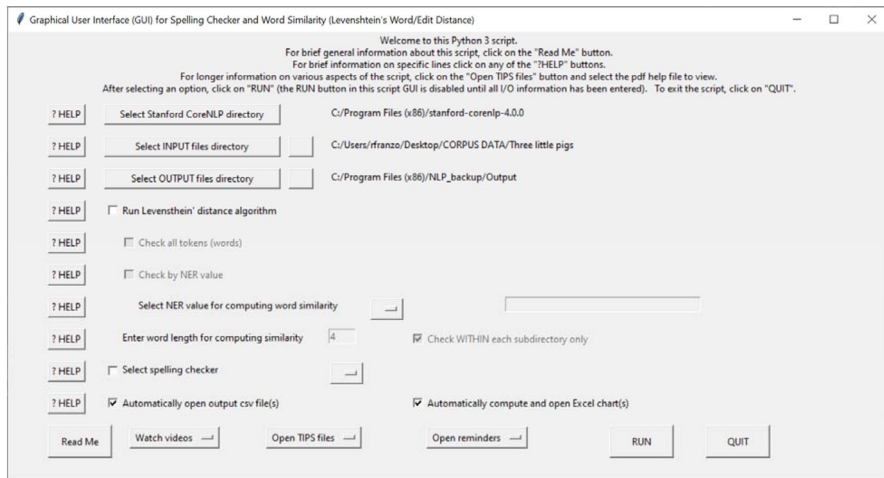


Fig. 25 Graphical User Interface (GUI) for Word Similarities

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

- Aggarwal, C.C., Zhai, C.: A survey of text classification algorithms. In: Aggarwal, C.C., Zhai, C. (eds.) *Mining Text Data*, pp. 163–222. Springer, Boston (2012)
- Beck, E.M., Tolnay, S.: 'The killing fields of the deep south: the market for cotton and the lynching of blacks, 1882–1930.' *Am. Sociol. Rev.* **55**, 526–539 (1990)
- Beck, E.M., Tolnay, S.E.: Confirmed inventory of southern lynch victims, 1882–1930. Data file available from authors (2004).
- Benedetti, F., Beneventano, D., Bergamaschi, S., Simonini, G.: Computing inter document similarity with Context Semantic Analysis. *Inf. Syst.* **80**, 136–147 (2019). <https://doi.org/10.1016/j.is.2018.02.009>
- Bialecki, A., Muir, R., & Ingersoll, G.: "Apache Lucene 4." *SIGIR 2012 Workshop on Open Source Information Retrieval*. August 16, 2012, Portland, OR, USA (2012).
- Brundage, F.: *Lynching in the New South: Georgia and Virginia, 1880–1930*. University of Illinois Press, Urbana (1993)
- Johansson, J., Borg, M., Runeson, P., Mäntylä, M.V.: A replicated study on duplicate detection: using Apache Lucene to search among android defects. In: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 8. ACM (2014)
- Brooker-Gross, S.R.: News wire services in the nineteenth-century United States. *J. Hist. Geogr.* **7**(2), 167–179 (1981)
- Cooper, J.W., Coden, A.R., Brown, E.W.: Detecting similar documents using salient terms. In: *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, 245–251 (2002)
- Edelmann, A., Wolff, T., Montagne, D., Bail, C.A.: Computational social science and sociology. *Ann. Rev. Sociol.* **46**, 61–81 (2020)
- Ericsson, K.A., Herbert, S.: *Protocol Analysis: Verbal Reports as Data*, 2nd edn. MIT Press, Cambridge, MA (1996)
- Evans, J.A., Aceves, P.: Machine translation: mining text for social theory. *Ann. Rev. Sociol.* **42**, 21–50 (2016)

- Fellbaum, C. (ed.): WordNet. An Electronic Lexical Database. MIT Press, Cambridge, MA (1998)
- Forsyth, R.S., Sharoff, S.: Document dissimilarity within and across languages: a benchmarking study. *Liter. Linguistic Comput* **29**(1), 6–22 (2014)
- Franzosi, R.: Quantitative Narrative Analysis, vol. 162. Sage, Thousand Oaks, CA (2010)
- Franzosi, R., De Fazio, G., Vicari, S.: Ways of measuring agency: an application of quantitative narrative analysis to lynchings in Georgia (1875–1930). *Sociol. Methodol.* **42**(1), 1–42 (2012)
- Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. *IJcAI* **7**, 1606–1611 (2007)
- Gambhir, M., Gupta, V.: Recent automatic text summarization techniques: a survey. *Artif. Intell. Rev.* **47**, 1–66 (2017)
- Grimm, J., Grimm, W.: [1812, 1857]. *The original folk and fairy tales of the brothers Grimm: The Complete First Edition*. [Kinder- und Hausmärchen. Children's and Household Tales]. Translated and Edited by Jack Zipes. Princeton, NJ: Princeton University Press (2014)
- Hutter, S.: Protest event analysis and its offspring. In: Donatella della Porta (ed.) *Methodological Practices in Social Movement Research*. Oxford: Oxford University Press, pp. 335–367 (2014)
- Jacobs, J.: *English fairy tales (Collected by Joseph Jacobs, Illustrated by John D. Batten)*. London: David Nutt (1890)
- Klandermans, B., Staggenborg, S. (eds.): *Methods of Social Movement Research*. University of Minnesota Press, Minneapolis (2002)
- Koopmans, R., Rucht, D.: Protest event analysis. In: Klandermans, Bert, Staggenborg, Suzanne (eds.) *Methods of Social Movement Research*, pp. 231–59. University of Minnesota Press, Minneapolis (2002)
- Kowsari, K., Meimandi, K.J., Heidarysafa, M., Mendu, S., Barnes, L., Brown, D.: Text classification algorithms: a survey. *Information* **2019**(10), 150 (2019)
- Labov, W.: *Language in the Inner City*. University of Pennsylvania Press, Philadelphia (1972)
- Lansdall-Welfare, T., Sudhahar, S., Thompson, J., Lewis, J., FindMyPast Newspaper Team, and Cristianini, N.: Content analysis of 150 years of british periodicals. *Proceedings of the National Academy of Sciences (PNAS)*, PNAS, Published online January 9, 2017 E457–E465 (2017)
- Lansdall-Welfare, T., Cristianini, N.: History playground: a tool for discovering temporal trends in massive textual corpora. *Digit. Scholar. Human.* **35**(2), 328–341 (2020)
- Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965 (Russian). English translation in *Soviet Physics Doklady*, 10(8):707–710, 1966. (Doklady is Russian for "Report". Sometimes transliterated in English as Doclady or Dokladi.) (1966)
- Levin, B.: *English Verb Classes and Alternations*. The University of Chicago Press, Chicago (1993)
- Lloret, E., Palomar, M.: Text summarisation in progress: a literature review. *Artif. Intell. Rev.* **37**, 1–41 (2012)
- MacEachren, A.M., Roth, R.E., O'Brien, J., Li, B., Swingle, D., and Gahegan, M.: Visual semiotics & uncertainty visualization: an empirical study. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 18, No. 12, December 2012 (2012)
- Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J. and McClosky, D.: The stanford CoreNLP natural language processing toolkit. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60 (2014)
- McAdam, D., Yang, Su.: The war at home: antiwar protests and congressional voting, 1965–1973. *Am. Sociol. Rev.* **67**(5), 696–721 (2002)
- McCandless, M., Hatcher, E., Gospodnetic, O.: *Lucene in Action*, Second Edition Covers Apache Lucene 3.0. Manning Publications Co, Greenwich, CT (2010)
- Miller, G.A.: WordNet: a lexical database for english. *Commun. ACM* **38**(11), 39–41 (1995)
- Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J.: Introduction to WordNet: an on-line lexical database. *Int. J. Lexicogr.* **3**(4), 235–244 (1990)
- Nenkova, A., McKeown, K.: A survey of text summarization techniques. In: Aggarwal, C.C., Cheng, X.Z. (eds.) *Mining Text Data*, pp. 43–76. Springer, Boston (2012)
- Murchú, T.Ó., Lawless, S.: The problem of time and space: the difficulties in visualising spatiotemporal change in historical data. *Proc. Dig. Human.* **7**(8), 12 (2014)
- Panitch, L.: Corporatism: a growth industry reaches the monopoly stage. *Can. J. Polit. Sci.* **21**(4), 813–818 (1988)
- Robertson, S., Zaragoza, H.: The probabilistic relevance framework BM25 and beyond. *Found. Trends® Inf Retr.* **3**(4), 333–389 (2009).
- Singhal, A.: Modern information retrieval: a brief overview. *Bull. IEEE Comput. Soc. Tech. Comm. Data Eng.* **24**(4), 35–43 (2001)

- Stein, B., Lipka, N., Prettenhofer, P.: Plagiarism and authorship analysis. *Lang. Resour. Eval.* **45**(1), 63–82 (2011)
- Taylor, J.R.: *Linguistic Categorization*. Oxford University Press, Oxford (2004)
- Tilly, C.: *Popular Contention in Great Britain, 1758–1834*. Harvard University Press, Cambridge, MA (1995)
- Turney, P.D., Pantel, P.: From frequency to meaning: vector space models of semantics. *J. Artif. Int. Res.* **37**(1), 141–188 (2010)
- Zhang, H., Pan, J.: CASM: a deep-learning approach for identifying collective action events with text and image data from social media. *Sociol. Methodol.* **49**(1), 1–57 (2019)
- Zhang, Y., Li, J.L.: Research and improvement of search engine based on Lucene. *Int. Conf. Intell. Human-Mach. Syst. Cybern.* **2**, 270–273 (2009)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.