Digitising Greek texts: Named entity recognition in graph-based database models

Acknowledgements

I am grateful to Prof. Margherita Fantoli and Frédéric Pietowski, who guided me throughout this project, and the Gerda Henkel Foundation, Düsseldorf, which supported its implementation (grant no. 16/F/22).

Introduction

Working with digital data has become a common fact in the field of Humanities for many years. Nowadays, scholars, specialists, and even non-specialists usually apply digital methods such as text analysis, data mining, topic modelling, data visualisation, digital mapping, and online publishing in their regular activities.

For historians, data focusing on past events are crucial. Digitised texts and information can now offer what old indexes, dictionaries, and encyclopaedias provided for generations, suggest new interpretations and methods of research, allow interoperability, and deliver to the public easy consuming outcomes.

On the one hand, for extracting (digital) data from ancient texts, the commonly used method is annotation. Through this operation, persons, places, events, trends, categories, and any other helpful information become digital entities that are easy to interpret and connect. Concerning people and places, the annotation can me made manually or automatic. On the other hand, data need specific formats to ensure interoperability. Relational databases are now challenged by graph-based models, which offer new increasing functionalities. Moreover, to be freely accessed, data can be shared in an open-linked semantic web.

In this context, this project focuses on those two aspects: extracting and linking data from ancient texts. After presenting the theoretical context, I explored, in the first part, the functionalities of a tool (Recogito) that allows manual annotation. As sources, I used some Greek texts of the Byzantine period. I extracted use case entities in tabular format and, to connect them with reference/authority control systems (like VIAF and Wikidata), I added (also manually) some standard identifiers. In the second part, I supplemented this traditional method with exploring the previous data in a graph-based environment (Neo4j). I simulated scenarios for (automatic) recognition of named entities and texts and assigning Wikidata identifiers.

1. Context: Entity tagging/recognition and graph-based formats

1.1. Text annotation

Broadly speaking, text annotation is a technique that allows adding supplementary information to texts. It refers to an old method that people have always used to mark the appearance of a specific element in the text, assign labels to words, explain certain parts of the text, summarise its content, or argue about the ideas included. Notes and comments placed into or around the text are standard annotations, which can concern a single word as well as phrases, sentences, paragraphs, or any other text sequence. In the past, humans annotated manuscripts and printed books. Now they can highlight points of interest in digital documents and web pages.[1]

Any annotation, not only the textual ones, contains a related body (Annotation – body – Body) and a related target (Annotation – target – Target) that is described or commented on in the body:[2]

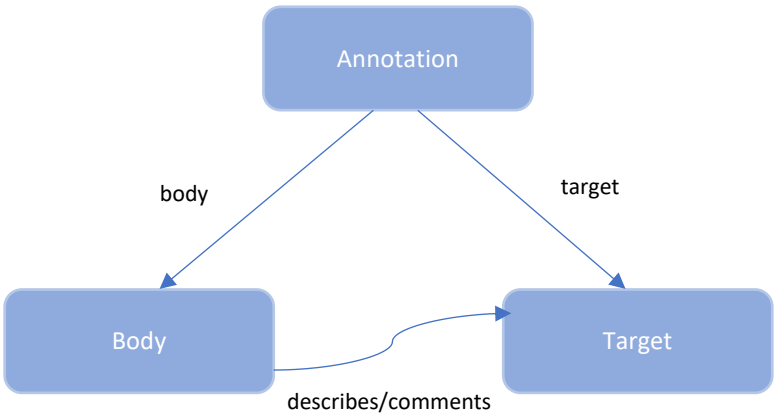


Fig. 1: Annotation model

In a more specific sense, text annotation points out the linguistic and semantic elements of a text. One the one hand, the linguistic or "structural" layer of annotation highlights the components of texts in their grammatical and syntactic context. The old indexes of names and places we find at the end of a book are among possible representations of this technique. On the other hand, "semantic" annotation focuses on the meaning and semantic role of words, phrases, and sentences. It collects and organises topics and passages relevant to specific research topics. A thematic index is an imperfect example in this respect. Those two elements are not exclusive. Many

[1] See tools like Annotator (https://annotatorjs.org/), Hypothesis (https://web.hypothes.is/), and Genius (https://genius.com/web-annotator), which add comments, tags, links, users, etc., to any webpage.

[2] It is the main structure of the Web Annotation Data Model recommended by W3C consortium, https://www.w3.org/TR/annotation-model/.

annotation techniques use both to increase the accuracy of the analysis and emphasise points of interest in the text.

It is possible to annotate a text either manually or automatically. The first method is the classical one; it implies human identification of and judgment upon every annotated entity (word, phrase, etc.). The (modern) automatic process is less time-consuming and requires little human intervention. In that later case, the term “text annotation” is sometimes replaced by “entity recognition,” even if the two expressions are also commonly used in different contexts.

The aim of text annotation/entity recognition is not only to provide additional information, metadata, or statistics but also to encourage the following applications. Annotation techniques are crucial in linguistic and corpus linguistics, natural language processing (NLP), machine learning, data science, and digital humanities.

Both manual and automatic text annotation includes several components, among which the most important are the following:

a. text segmentation; this process refers to dividing the text into logical units, such as sentences and words (tokenisation);
b. part-of-speech (POS) tagging is one of the most important parts of annotation, which concerns words’ labelling with part-of-speech tags, such as nouns, verbs, adjectives, adverbs, etc.; in English, the Universal Dependencies tagset lists 17 parts of speech, including ADJ (adjective), ADV (adverb), INTJ (interjection), NOUN, NUM (numeral), PART (particle), PRON (pronoun), PROPN (proper noun), PUNCT (punctuation), and VERB;[3] for a sentence like:

   Edith Borremans lives in Leuven, the capital of the province of Flemish Brabant.

   the following part-of-speech tags are identifiable:

Words	*Edith*	*Borremans*	*lives*	*in*	*Leuven*	*,*	*the*
Tag	PROPN	PROPN	VERB	ADP	PROPN	PUNCT	DET
Words	*capital*	*of*	*the*	*province*	*Flemish*	*Brabant*	*.*
Tag	NOUN	ADP	DET	NOUN	ADJ	PROPN	PUNCT

[3] De Marneffe, Manning, Nivre, and Zeman, 2021.

where ADP means apposition (preposition/postposition) and marks a noun's spatial, temporal, or other relation; and DET means determiner and marks noun phrase properties;

c. syntactic parsing; this process involves the identification of the constituent phrases into sentences, such as noun phrases, verb phrases, infinitive phrases, etc.; in the previous sentence, *the capital of the province of Flemish Brabant* is a noun phrase related to *Leuven*; in the sentence: *Today she travels from Leuven to Antwerp to see a friend*, the expression *to see a friend* is an infinitive phrase;
d. named entity tagging/recognition (NER) concerns the identification of the named entities such as persons, places, and organisations; there are four common tags: PER for people, characters, etc.; LOC for places, regions, mountains, etc.; ORG for companies, sports teams, etc.; and GPE for geo-political entities such as states and parties;[4] I described this technique in the next section;
e. entity/concept tagging/extraction; this process refers to identifying and tagging other entities/concepts than proper names relevant to the text;
f. coreference resolution means linking words/phrases that refer to the same entity; in the previous examples, *Leuven* and *the capital of the province of Flemish Brabant*, as well as *Edith Borremans* and *she* refer to same entities;
g. semantic role labelling is the process of identifying the semantic roles played by words/phrases in sentences in relation to a given predicate, such as agent, effect, condition, time, and place; the are many proposals for a standard list of thematic roles but no universally agreed-upon one; the oldest belongs to the Sanskrit philologist Pāṇini (5th c. BCE) in his sutra-style treatise Aṣṭādhyāyī (The book of eight chapters); the modern lists includes models such as FrameNet, VerbNet, and PropBank;[5] in the latter, there are six core semantic roles, usually A(RG)0-5: A0 is the agent of the predicate; A1, the effect/influence of the predicate; A2-5 have different meanings depending on the predicate; there are also 15 additional semantic roles, such as BNE beneficiary; CND, condition; DIR, direction; DGR, degree; EXT, extent; FRQ, frequency; LOC, location; TMP, time; etc.;[6] in one of the previous examples, the following semantic roles could be identified:

Words	*Today*	*she*	*travels*	*from*	*Leuven*	*to*	*Antwerp*
Role	TMP	A0	predicate	LOC		LOC	

[4] Jurafsky and Martin, 2023 (ch. 8).
[5] https://framenet.icsi.berkeley.edu/; https://verbs.colorado.edu/verbnet/; https://propbank.github.io/.
[6] Palmer, Gildea, and Kingsbury, 2005; Jurafsky and Martin, 2023 (ch. 22).

h. relationship extraction refers to identifying relationships between extracted entities and tagging them with related external or internal domain knowledge; for example, the entities *Leuven* and *capital of the Flemish Brabant* could be linked with a relationship of type *IsCapitalOf*;
i. entity linking/disambiguation (NEL – Named entity linking, in the case of named entities) refers to identifying stable identifiers for text entities in standard databases or ontologies (such as Wikidata);[7]
j. finally, stance detection means understanding whether the text's author is neutral or subjective towards the labelled entity; in the latter case, the polar orientation (i.e., favour or against) must also be defined.

Nowadays, automatic annotation techniques have good results with modern datasets. However, the situation is a little different in the field of Humanities, with historical corpora, for example. Lack of or poor existing annotated texts, inconsistent or erroneous spelling (sometimes generated by OCR tools, not customary in creating contemporary datasets), use of several languages in the same corpus, and especially the linguistic feature (not as anglophone as in the modern world) of those corpora are among the causes of this retard.[8]

This last aspect is even more complicated due to the limited number of specialists. In its whole complexity, the treatment of non-Latin texts, for example, could be understood in this context:

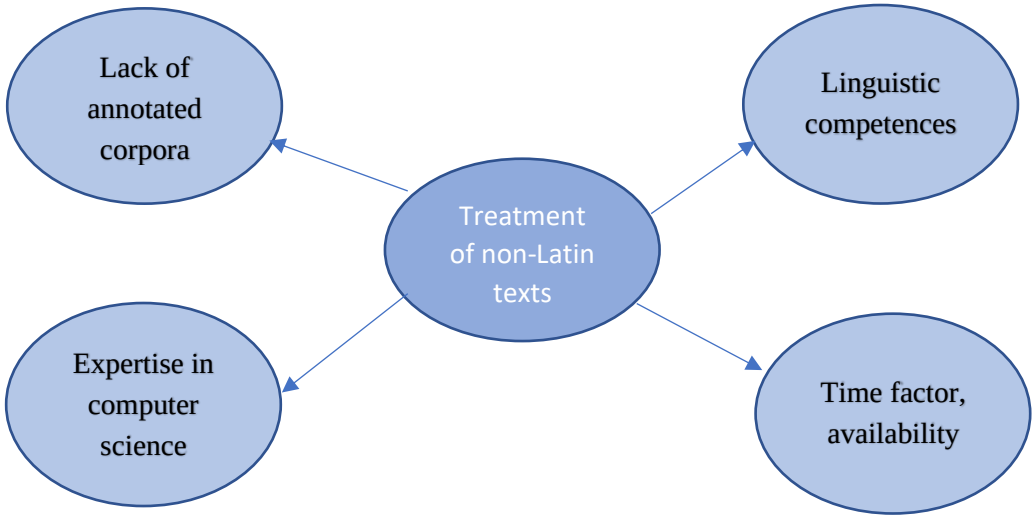


Fig. 2: Treatment of non-Latin texts

[7] Pontes et al., 2020.
[8] Ehrmann et al., 2016; Humbel et al., 2021.

Despite those difficulties, however, there are already some remarkable projects in this regard:

- Trismegistos (KU Leuven), a metadata platform that uses NER procedures for extracting Greek and Coptic names from ancient texts;[9]
- Coptic Scriptorium, a research portal containing tools for segmentation, normalisation, part-of-speech tagging, lemmatisation, and language of origin tagging;[10]
- Calfa, a tool that includes, among others, lemmatisation, POS tagging, and morphological engine for Eastern and Classical Armenian, Old Georgian, Syriac, and Ancient Greek;[11]
- GREgORI (University of Louvain-la-Neuve), a tool that performs the automatic (morphological) processing of Greek and other Oriental languages.[12]

During the current project, I used Greek texts and manually annotated them using the following (previously discussed) methods: d. named entity tagging (people and places); e. entity/concept tagging (time and event); h. relationship extraction; i. entity linking (Wikidata, VIAF, Trismegistos, GeoNames identifiers).

1.2. Named entity recognition

Among text annotation techniques, one of the most used is named entity tagging/recognition. Indeed, proper names structure the content of a text. They are probably the most relevant type of information it includes. In the past, a thematic index was not currently added, but many books included an index of people and places.

Generally, a named entity means an item that can be referred to with a proper name. As mentioned, there are four common entity tags (PER, LOC, ORG, GPE), but the notion is also extended to other entities, such as dates, times, currencies, and numerical expressions. As with other text annotation/entity recognition processes, named entity recognition (NER) includes two steps: named entity identification (NEI) and named entity classification (NEC). The order of those two stages is not compulsory.

One of the major problems of entity tagging is identifying words and spans of text that refer to entities. One of the solutions is the so-called BIO tagging method, which labels words of interest with “I” (from “inside”), words out of interest with “O” (from “outside”), and the beginning of a span of interest with “B.”[13] With this notation, the previous sentence concerning *Edith Borremans* is tagged as follows:

---

[9] Roux and Depauw, 2015; https://www.trismegistos.org/ref/about_naw.php.
[10] Zeldes and Schroeder, 2016; https://copticscriptorium.org/tools
[11] Vidal-Gorène et al., 2020; https://calfa.fr./textanalysis#details.
[12] Kindt et al. 2022.
[13] Ramshaw and Marcus, 1995.

Words	*Edith*	*Borremans*	*lives*	*in*	*Leuven*	*the*
BIO Label	B-PER	I-PER	O	O	I-LOC	O
Words	*capital*	*Of*	*the*	*province*	*Flemish*	*Brabant*
BIO Label	O	O	O	O	B-GPE	I-GPE

Other tagging schemes include IO tagging, which does not tag the beginning of the span, and BIOES tagging, which contains an "E" for the span's end and "S" for spans containing only one word of interest.

Another aspect of entity tagging refers to entity meaning, which can differ from one context to another. In the following examples,[14] "Washington" refers to different types of entities:

Washington [PER] was born into slavery on the farm of James Burroughs.

Blair arrived in Washington [LOC] for what may well be his last state visit.

Washington [ORG] went up 2 games to 1 in the four-game series.

In June, Washington [GPE] passed a primary seat belt law.

When tagging locations, a helpful feature is a gazetteer, i.e., a list of places that could include detailed information of geographical, historical, or social relevance. If the tagged location is identified in the gazetteer, a link between them is established. For example, GeoNames (11 million placenames)[15] and Geographic Names Server (13 million placenames)[16] provides gazetteers for all the places on the earth; The Getty Thesaurus of Geographic Names (3 million placenames) adds information to places;[17] U.S. Board on Geographic Names include a gazetteer of the domestic names.[18] Pleiades is a gazetteer of ancient places.[19]

[14] Jurafsky and Martin, 2023 (ch. 8).
[15] https://www.geonames.org/
[16] https://geonames.nga.mil/geonames/GNSHome/index.html
[17] https://www.getty.edu/research/tools/vocabularies/tgn/index.html
[18] https://www.usgs.gov/us-board-on-geographic-names/domestic-names
[19] https://pleiades.stoa.org/

The same facility could be used for names and corporations, but their efficacy is less evident than in the case of geographical gazetteers.

The utility of named entity tagging is obvious. It allows the transformation of unstructured data into information that can be reused and improved. When two persons or organisations sign a contract, the NER system can scan the document and check if the included proper names, organisation names, addresses, etc., match the same entities already existing in contractors' or external databases. This identification diminishes the manual work, especially in the case of large-scale types of checking.

For automatic recognition of named entities, NER systems (as well as POS tagging ones) use three standard types of approaches:

a. rule-based approaches, commonly used until the late 1990s, which identify entities through sets of linguistic rules manually introduced by the annotator;
b. feature-based or machine-learning-based approaches, developed since the 2000s, which allow machines to learn from already existing corpora on the basis of selected features and assign tags to words (or phrases) using statistical models (such as hidden Markov model – HMM, support vector machine – SVM, and conditional random field – CRF – model) applied to given sequences of (preceding) words; there are three types of learning: supervised, semi-supervised, and unsupervised;[20]
c. neural-based or deep learning approaches, currently the dominant ones, which use artificial neuronal networks to allow machines to automatically learn representations of data with increasing levels of abstraction without using manually crafted features.[21]

I mentioned that annotated corpora are less numerous in the field of Humanities than in domains that use contemporary data. However, named entity tagging tools become of vital importance in historical research, for example, where the "5 W's" – who did what when and where with whom – are crucial.[22] In this respect, some projects that implement named entity tagging/recognition could be cited:

- Transkribus,[23] a platform that includes the recognition of historical data in documents;
- NewEye,[24] a multilingual (French, German, Finnish, and Swedish) dataset of 19th-20th-century newspapers, created using named entity recognition and linking;

[20] Nadeau and Sekine, 2007.
[21] Ehrmann et al., 2021.
[22] Ehrmann et al., 2016.
[23] Kahle et al., 2017; https://readcoop.eu/transkribus
[24] Hamdi et al., 2021; https://github.com/newseye.

- Chinese Text Project (Durham University)[25] includes MARKUS,[26] an online tool for both manual and automatic labelling of named entities in ancient Chinese texts;
- GraphManuscribble (University of Freiburg),[27] an interface that allows manually annotating digital facsimiles of manuscripts;
- a dataset of 19th-century French printed trade directories[28] created using OCR and NER techniques;
- a dataset of Swiss newspaper archives *Le Temps*, tested with four different NER tools;[29]
- a training dataset for Dutch NER in the archaeology domain.[30]

As mentioned for text annotation, the non-Latin texts enjoyed much less attention than the English ones. However, Japanese, Chinese, Korean, and Arabic have started to receive more interest in the last few years.[31] For modern Greek, a named entity recogniser based on a corpus of 130.000 words was developed in the 2000s;[32] a BERT-based language model trained on 29 GB of Greek texts in the 2020s.[33]

For ancient languages, Classical Language Toolkit (CLTK) is probably one of the best-known libraries to perform NLP tasks, including NER.[34] Trismegistos, Calfa, and GREgORI, tools already mentioned, include automatic recognition of named entities in ancient Greek, Armenian, Georgian, and Syriac.[35] A tool for annotating ancient Greek text using INCEpTION platform was announced in 2019.[36] Recently, annotation projection (i.e., performing linguistic annotation of a text and projection of it to its translation, e.g., from English to ancient Greek) was suggested as a method to improve NER techniques for ancient languages and help the creation of trained corpora.[37]

A standard pipeline for recognition of annotations/named entities in Greek texts, based on gazetteers and other resources, would contain the following elements:

- identifying the named entities as annotations;
- selection of all distinct annotations;
- identifying the uninflected form of the words (lemmata) based on gazetteers and lexical resources;

---

[25] Sturgeon, 2021; https://ctext.org/
[26] https://dh.chinese-empires.eu/markus/
[27] Garz et al., 2016; https://diuf.unifr.ch/main/hisdoc/graphmanuscribble.html
[28] Abadie et al., 2022; https://github.com/soduco/paper-ner-bench-das22.
[29] Ehrmann et al., 2016.
[30] https://topostext.org/
[31] Nadeau and Sekine, 2007.
[32] Boutsis et al., 2000.
[33] Koutsikakis et al., 2020.
[34] Johnson et al., 2021.
[35] Vidal-Gorène et al., 2020; Kindt et al. 2022.
[36] Berti, 2019; https://inception-project.github.io.
[37] Yousef et al., 2023.

- validation of the named entities;
- linking named entities with texts and annotations.

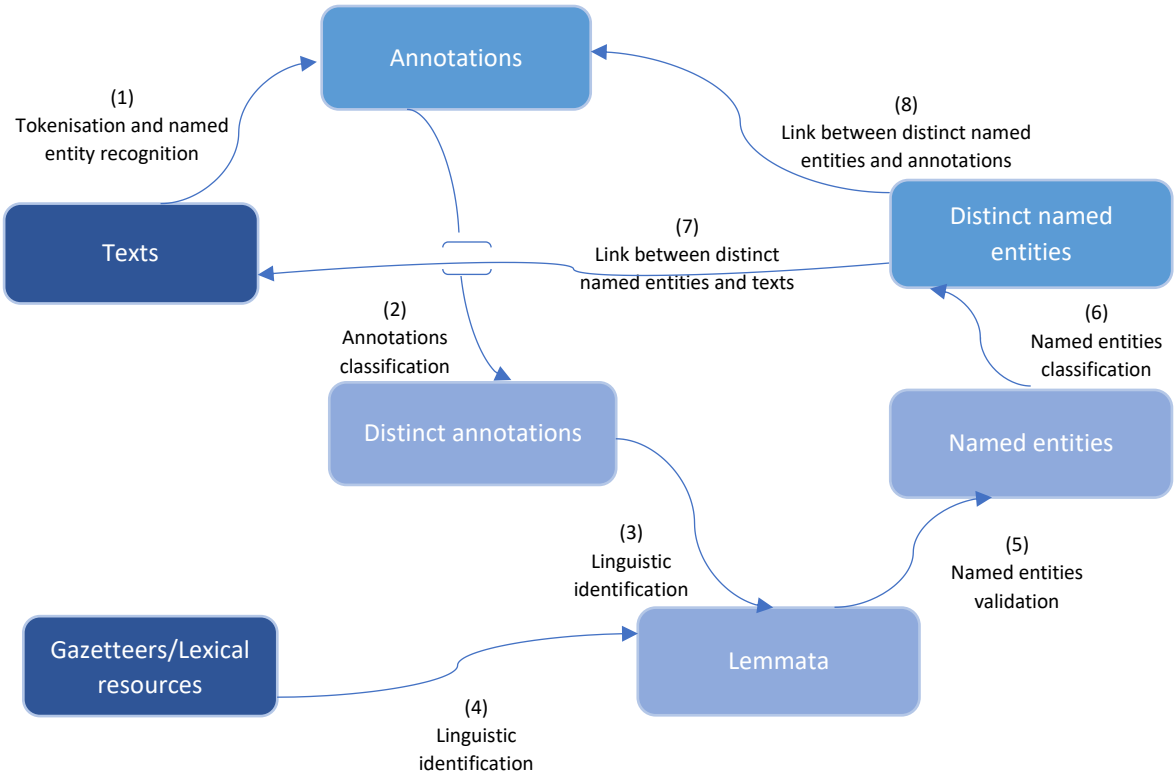


Fig. 3: Named entity recognition standard pipeline

Trismegistos platform, which divides named entities into “People” and “Places,” contained in 2022 the following number of records concerning persons:[38]

Tables	**Records**
Annotations (NamRef)	20,711
Distinct annotations (Ref)	11,017
Named entities (NamVar)	229,300
Distinct named entities (Nam)	38,833[39]

[38] https://www.trismegistos.org/ref/about_naw.php#snap-ner
[39] The number of distinct people is more significant than the number of annotations because this table, which also plays the role of the gazetteer, was enriched with external datasets, see Roux and Depauw, 2015.

For linguistic identification, Trismegistos checks every distinct annotation found in texts against all possible declined names (in my opinion, this is the great strength of this platform):

Table	Records
Declined name forms (NamVarCase)	998,637

In this general context, this project suggests methods for identifying named entities in texts and assigning them stable identifiers in a graph-based (Neo4j) context. I used a basic method that recognises words beginning with capitals and compared them with other entities in the graph or Wikidata catalogue.

1.3. Named entity linking

Once named entities are recognised, the research interest is that they can be used and understood not only in internal but also in external contexts. In this respect, named entities become accessible by assigning URIs, linking to other existing stable identifiers (also URIs), and exporting in RDF—Resource Description Framework—standard formats.

This is the trend followed today by many DH applications. Pelagios Network[40] and Linked Pasts symposia,[41] for example, place the automatic recognition of stable identifiers in the centre of their preoccupations. With Recogito platform, one of the partners of those networks, after uploading texts and manually annotating them, the users can validate them against online gazetteers thanks to advanced search engines.[42] In Trismegistos project, each named entity found in texts (also every text and author) received a stable URI (e.g., the place called Karyanda has its own URI, www.trismegistos.org/place/3409, and identifier, TM Geo ID 3409). The links with similar identifiers (extracted, in this case, from the web-based platforms Digital Atlas of the Roman Empire, Nomisma, Pleiades, and ToposText) were provided. Data can be exported in RDF/XML format.[43]

However, those aspects are not enough to define data as Linked Open Data, whose principles (and philosophy) Tim Berners-Lee explained almost twenty years ago:

[40] https://pelagios.org/about-us/
[41] https://linkedpasts.hcommons.org/
[42] Simon et al., 2017; https://recogito.pelagios.org/.
[43] https://www.trismegistos.org/dataservices/

- use URIs as names for things;
- use HTTP URIs to see URIs;
- use RDF and SPARQL standards to access/query URIs;
- include links to other URIs to interconnect URIs.[44]

While the RDF model has become an option for storing, connecting, and publishing data, most DH applications do not assign URIs to all their constituents, and thus they cannot be fully interrogated via SPARQL endpoints. Moreover, they do not allow data enrichment from complementary databases. Problems such as commonly accepted ontologies and conflicting data coming from different sources raise difficult challenges in this respect.[45]

Despite those difficulties, there are an increasing number of ontologies that recommend standards in different domains: CIDOC-CRM (Comité International pour la documentation - Conceptual Reference Model)[46] and FRBR (Functional Requirements for Bibliographic Records)[47] in cultural heritage, as well as ongoing projects, such as Europeana (resources are addressable and dereferenceable by their URIs)[48] and Lila - Linking Latin (CIRCSE—Centro Interdisciplinare di Ricerche per la Computerizzazione dei Segni dell'Espressione—Milan),[49] integrated in Linguistic Linked Open Data[50] movement.

1.4. Relational database, graph database, and linked open data models

Data and corpora obtained using the previously described methods are stored in databases. The interest to organise and use them optimally is crucial. While the (traditional) relational database model and SQL query language have fully proven their efficacy, the volume and complexity of the data accessed today sometimes require very different structures than relational databases. Experts anticipate dramatic growth in using those new types of systems, called NoSQL, i.e., non-SQL or not only-SQL. They include graph, key-value-based, column-based, document-based, and object-oriented databases. Compared to relational databases, NoSQL ones do not have to follow a fixed schema, store data more intuitively, can add new entities to an existing database (horizontal scalability), and handle large amounts of data. Even if there still needs to be more standardisation and maturity in this domain, it is a very promising one.

---

[44] Berners-Lee, 2006.
[45] Hyvönen et al., 2019.
[46] https://www.cidoc-crm.org/
[47] https://www.ifla.org/resources/
[48] https://pro.europeana.eu/page/linked-open-data
[49] https://lila-erc.eu/; Fantoli et al., 2022.
[50] https://linguistic-lod.org/

Among NoSQL models, graph-based databases play a central role.[51] They can be divided into two categories:

a. native (storage-based) graph databases, usually known as LPG—Labelled Property Graph—databases, developed, for example, by Neo4j,[52] InfiniteGraph,[53] and Ontotext GraphDB;[54]
b. non-native graph or model-graph databases; the RDF triple-store model is the best known among them.

Both LPG and RDF models utilise graph theory concepts. However, they were created with different aims. The first focuses on more efficient storage and fast querying of local (big) data; it provides a solution to explore complex, well-connected data.[55] The second one is less interested in efficiency and more in connectivity and open access; it allows everyone to publish and use data freely. The name commonly used for this model is Linked Open Data.

There are already heated discussions among specialists about the relationship between relational, (native) graph, and linked open data models. Sometimes, the opinions are biased, and the judgements are too severe. I tried to summarise the main common points and differences between the three models as follows:

	SQL database	**NoSQL database**	
	Relational database (RD)	**Labelled property graph (LPG)**	**Linked open data (LOD)**
Structure/ Schema/ Ontology	Collection of tables containing collections of the same type of data (rows) with multiple properties (columns). Tables are connected by (primary and foreign) keys, which are table columns.	Collection of entities/nodes (analogue of RD rows) with multiple properties (analogue of RD columns) grouped in categories (analogue of RD tables). Entities are linked by relationships/edges, which can have properties. Entity and relationship properties guarantee the great flexibility of the model. Together entities and	Collection of URIs (analogue of RD rows and LPG entities)—with literal values but without properties—grouped in classes (analogue of RD tables). Two URIs (subject-object) are linked by a predicate (analogue of LPG relationship), which is also a URI and cannot have properties. Together URIs and

[51] DB-Engines, Knowledge Base of Relational and NoSQL Database Management Systems, https://db-engines.com/en/ranking_categories.
[52] https://neo4j.com/
[53] https://objectivity.com/infinitegraph/
[54] https://www.ontotext.com/products/graphdb/
[55] Barrasa, 2017; Joyce, 2021.

		relationships create a (native property) graph.	predicates create a model-graph structure.
Querying	SQL language queries data using keys (indexed pointers) and joins on tables. The process could have poor performance in complex systems.	Declarative languages allow complex relationship-based queries (not possible with RD keys or LOD properties) optimised for navigating highly connected data. They typically use index-free adjacency (through graph relationships, which can connect every two nodes in the database).	SPARQL language allows for querying across multiple linked datasets using graph patterns. It cannot evaluate graph properties (they have no attributes) and could perform poorly when it includes complex patterns processed over complex systems.
Focus	• databases with many tables and simple relationships	• complex systems; • data storage; • faster querying	• structured data available on the web in open access; • standardisation; • interoperability
Standards	Standard query language (SQL)	There are no standard representation and query language. Every organisation creates its own semantics and querying language (such as Neo4j's Cypher language). There is no interoperability between local graph databases.	World Wide Web Consortium (W3C)[56] supports standard semantics (RDF) and querying language (SPARQL), which allows interoperability between any graphs based on RDF.
Usage	• business • administration • transaction-focused use cases	• social networking; • real-time recommendation engines; • fraud detection; • network management; • knowledge graphs	• "semantic web" (machine-readable and interconnected open web data); • knowledge graphs; • scenarios requiring reasoning and inference with other data stores

[56] https://www.w3.org/

2. Implementation

This part includes two steps. First, I selected texts, manually annotated them, provided entities, and assigned stable identifiers to each entity. Second, I explored Neo4j environment, imported data, and wrote some basic procedures. The picture of the project follows:

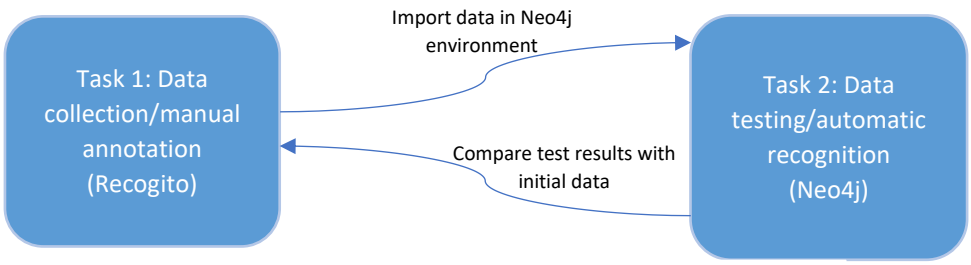


Fig. 4: Project's steps

2.1. Task 1: Data collection and analysis

During the first phase, I manually collected and examined the data needed for implementation. The main steps of this task (described below) were the following:

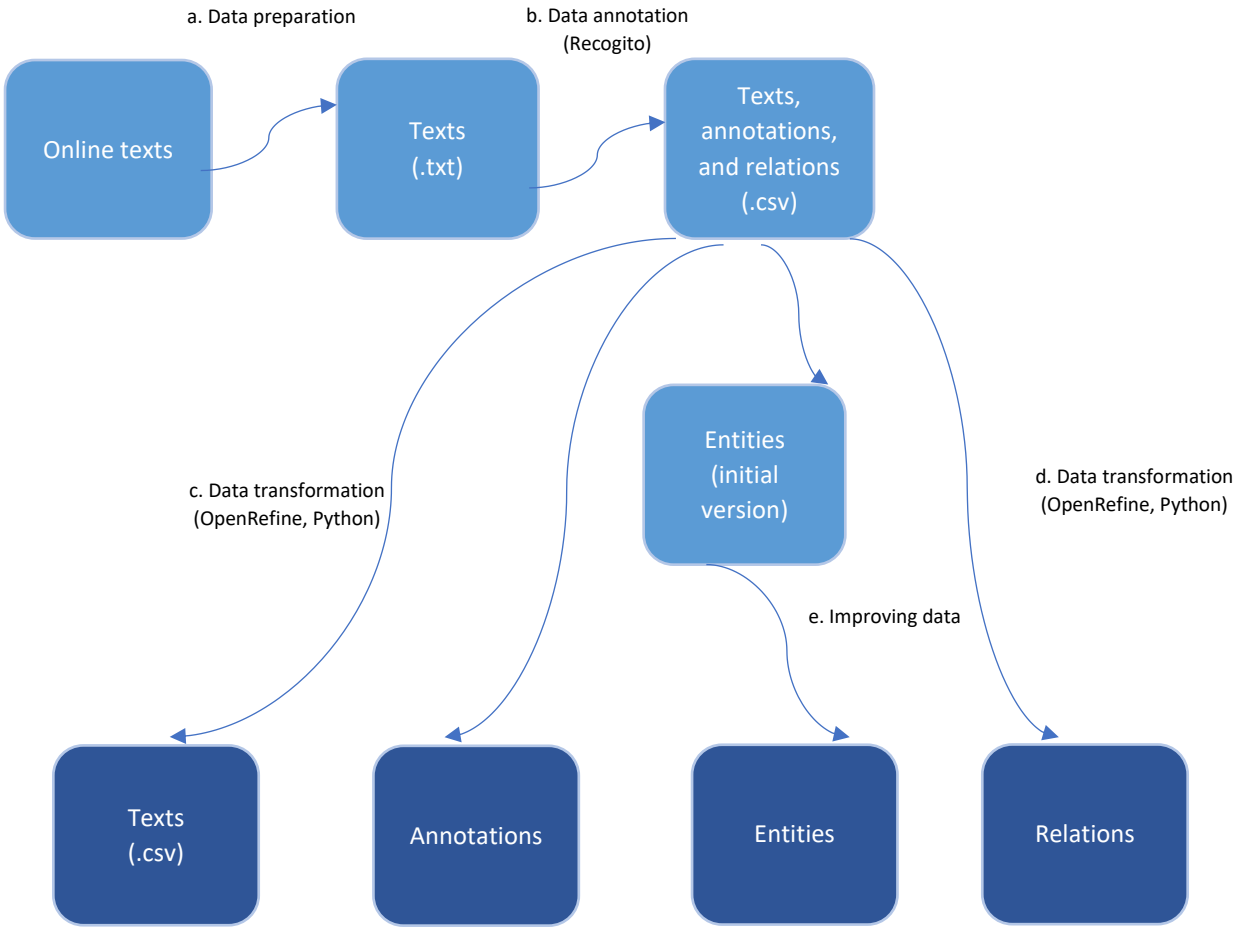


Fig. 5: Annotation task pipeline

2.1.1. Objectives

The main objective of this task was to provide a dataset. For further use during the test phase, the dataset had to contain three files with the following types of information:

a. texts, grouped in a table (future Texts.csv) and identified by an ID_Text (e.g., ID_Text = 1, Text = Prometheus gave fire to the humans and was punished for it by Zeus.);
b. entities of interest found in texts, also grouped in a table (future Entities.csv) and identified by an ID_Entity, a label, and a type; the label refers to the dictionary form of the entity (in the case of the proper Greek nouns, this label is the word in the nominative case); the type refers to the general category from which the word belongs (e.g., ID_Entity = 1, Label = Prometheus, Type = titan; ID_Entity = 2, Label = Zeus, Type = god);
c. annotations, which link entities to texts, grouped in a table (future Annotations.csv) and identified by an ID_Annotation, the text and entity IDs, as well as the start and stop positions in the text (e.g., ID_Annotation = 1, ID_Text = 1, ID_Entity = 1, Start = 0, Stop = 0; ID_Annotation = 2, ID_Text = 1, ID_Entity = 2, Start = 62, Stop = 65).

I added a new one, “Relations,” explained below, to those three files, containing the relationships between entities of type “Travel” and other entities that describe travels.

In relation to the data collection, as a secondary objective, I examined the texts, entities, and annotations and provided the first summary information.

2.1.2. Data source

I chose five medieval Greek writings containing hagiographical narratives as primary texts. The following considerations dictated the choice:

a. the texts are part of a group of Greek narratives that have been the object of my research for several years;
b. for the moment, full ancient Greek texts with diacritics (accents and breathings) are not readily available; however, the selected narratives can be freely accessed, among other similar texts, through the “Hagiography Database” webpage[61] of Dumbarton Oaks Research Centre, Washington, D.C., an institution affiliated to Harvard University; the database contains 120 lives of the 8th-10th-century saints retyped from the original editions; it allows the user searching for citations according to different filters;[62]
c. finally, the five texts contain many references to travels, which is not a regular feature of hagiographical narratives; the second narrative (the Life

[61] https://www.doaks.org/research/byzantine/resources/hagiography/database/dohp.asp
[62] https://www.doaks.org/research/byzantine/resources/hagiography/database/dohp.asp?cmd=CitSearch&catsel=1

of Elias the Younger), for example, describes the travels of its hero in detail, who moved many times from Sicily to northern Africa, Levant and back to Calabria and lived in or visited Rome, Amalfi, Palermo, Taormina, Jerusalem, Alexandria, Mount Sinai, Antioch, Corfu, Sparta, and Thessaloniki; the selected texts allow, therefore, not only extracting places but also connecting them; I intend building a corpus of such connected places and exploit it in its historical context.

The primary texts used in this project are the following:

ID_Text	Label	Word count	Token count	BHG no.[63]	Edition	Online edition
1	Life of Michael Synkellos	11,261	14,378	1296	M. B. Cunningham, The Life of Michael the Synkellos (Belfast Byzantine Texts and Translations 1), Belfast, 1990	https://www.doaks.org/research/byzantine/resources/hagiography/database/texts/22.html
2	Life of Elias the Younger	13,445	16,585	580	G. Rossi Taibbi, Vita di S. Elia il Giovane (Istituto Siciliano di Studi Bizantini e Neollenici, Testi 7), Palermo, 1962	https://www.doaks.org/research/byzantine/resources/hagiography/database/texts/59.html
3	Life of Dounale-Stephen	576	643	2110	H. Delehaye, Synaxarium Ecclesiae Constantinopolitanae, Acta Sanctorum Propylaeum Novembris, Bruxelles, 1902, 317-322	https://www.doaks.org/research/byzantine/resources/hagiography/database/texts/62.html

[63] In the field of Byzantine studies, the hagiographical texts are commonly identified by their BHG, the reference number assigned to the corresponding text in Halkin, 1957-1984.

4	Life of Andrew of Crete	2,510	2,826	113	A. Papadopoulos-Kerameus, Analekta Hierosolymitikes Stachylogias 5, St. Petersburg, 1899 (Brussels, 1963), 169-179	https://www.doaks.org/research/byzantine/resources/hagiography/database/texts/102.html
5	Passion of the Sixty Martyrs of Jerusalem	1,753	2,010	1217	A. Papadopoulos-Kerameus, Pravoslavnyi Palestinkij Sbornik 12.1 (1892), 1-7	https://www.doaks.org/research/byzantine/resources/hagiography/database/texts/116.html
Total		**29,545**	**34,633**			

Table 1: The primary texts

2.1.3. Tools

I used three open software to collect and explore data: Recogito, OpenRefine, and Python.

Recogito[64] is an annotation tool created and maintained by Pelagios Network,[65] a Digital Humanities initiative that aims to link information related to places and periods. Recogito offers a user-friendly workspace for uploading and annotating texts, identifies and maps places, creates relationships between entities, and allow exporting both entities and relationships in different formats.

The annotations are identified by a label (introduced as a comment). The standard categories/types suggested for annotations are “Place,” “Person,” and “Event,” but ignoring those categories allows the user to define any other type, eventually as a tag. This latter functionality also enables the introduction of all the annotations’ properties. A universally unique identifier (UUID) is automatically generated for each annotation. To export annotations, the user can choose between CSV and RDF formats. In the second case, the generated URI includes the text and annotation identifiers. However, the associated vocabulary is currently not available on the web.

The relationships can be created between any two annotations. The relationships are identified by a name and a direction (from source to target) and can be exported as Standard CSV and Gephi CSV.

OpenRefine[66] is a popular and powerful tool for data wrangling. The application was initially developed by Metaweb as Freebase, then acquired by Google and renamed

[64] https://recogito.pelagios.org/
[65] https://pelagios.org/
[66] https://openrefine.org/

first Google Refine, and finally OpenRefine. It allows cleaning and transforming data, using, among other methods, GREL (= General/Google Refine Expression Language) functions comparable to SQL applications. With OpenRefine it is also possible to add data from the web using a "reconciliation" service, which maps database information with identified entities, such as Wikidata[67] records.

Python[68] is a high-level programming language initially designed by Guido van Rossum and now developed by Python Software Foundation. I used Pandas library[69] for more complex operations, which were difficult to realise with OpenRefine.

2.1.4. Methodology

To accomplish this task, I used three main methods:

a. Text analysis and annotation

This method refers to identifying persons, places, and travels in the selected Greek texts. With Recogito, the persons and places can automatically receive the types "Person" and "Place." To better filter those two categories, I added a subtype, such as "emperor," "king," "emir," "pope," "monk," "citizen," or "philosopher" for persons and "city," "region," "mountain," "river," or "island" for places. I partially used the VIAF authority file associated with the entity to define those titles. I listed the complete catalogue of subtypes in the next section.

With Recogito, I classified travels as "Events" and identified each travel by four properties: "TravelTo," "TravelFrom," "TravelPerson," and "TravelTime," defined (also in Recogito) as relationships. In this regard, I also identified entities of type "Time."

b. Data processing

This method concerns all the transformations needed to obtain the final files (in the required form) from the initial data Recogito delivered. It includes building "Annotations" and "Entities" files containing complementary but different data.

c. Data identification

With this final method, the entities previously defined (of type "Person" and "Place") were linked with their stable identifiers. On the one hand, for places, Recogito suggests a first identification of this type based on supported gazetteers, such as Pleiades, GeoNames (see section 1.2.). On the other hand, OpenRefine automatically gives Wikidata identifiers for both persons and places through its reconciliation service. However, I had to check all those links manually.

---

[67] https://www.wikidata.org/wiki/Wikidata:Main_Page
[68] https://www.python.org/
[69] https://pandas.pydata.org/

2.1.5. Workflow

The detailed workflow of the task includes the following steps:

a. Data preparation

The texts downloaded from Dumbarton Oaks' webpage needed small arrangements, such as eliminating the title and pages and identifying a standard layout. The outcome of this step is the file Texts.csv, containing the complete Greek text (column "Text") of the selected hagiographies, identified by their ID_Text (from 1 to 5). I also added supplementary information, such as the edition and translation of each text. The structure of this file is the following:

Texts.csv
ID_Text
Label
BHG
Text
Edition
Translation

Table 2: Texts.csv structure

b. Text annotation

I defined four types of annotations: "Person," "Place," "Event," and "Time," as well as a custom subtype for the entities of types "Person" and "Place." All the annotations of type "Event" have the subtype "travel" and those of type "Time" the subtype "time" (only to keep the symmetry of records). The list of subtypes is the following:

TYPE	**Subtype**
PERSON	apostle, asekretis, biblical man, biblical woman, bishop, brother-in-law, caliph, citizen, delegate, emir, emperor, empress, father, general, god, governor, king, martyr, martyrs, martyrs/pilgrims, monk, mother, mythological winged horse, nun, patriarch, patrikios, philosopher, pope, presbyter, prisoner, prophet, saint, servant, sick person, sister, spatharios, strategos, stratelates, topoteretes
PLACE	bay, church, city, country, diakonia, island, lake, monastery, mountain, ocean, region, river, see, strait

EVENT	travel
TIME	time

Table 3: TYPE and Subtype fields

For people, Recogito's field "COMMENTS" allows the introduction of the English name, title, and dates associated with the annotated person. I chose only those three elements, which are mandatory according to the Functional Requirements for Bibliographic Records (FRBR)[70] and Functional Requirements for Authority Data (FRAD).[71] For "PLACE," "EVENT," and "TIME," this field includes only the name of the entity, like in these examples:

TYPE	**Subtype**	**COMMENTS/Label_ENG**
PERSON	king	Adalbert, King of Italy, 950-961
PERSON	emperor	Constantine VI, Byzantine Emperor, 780-797
PLACE	city	Antioch
PLACE	mountain	Mount Sinai
EVENT	travel	Michael Synkellos and Job from Broussa to Constantinople
EVENT	travel	Chrisionos from Taormina to Erikousa
TIME	time	843
TIME	time	730 (ca.)

Even if Recogito saves the start position of each annotation, I included two extra pieces of information among the tags: the paragraph and pagination. This metadata is helpful for the identification of entities/annotations in the initial texts by a reader not necessarily familiar with digital data:

[70] International Federation of Library Associations and Institutions, Functional Requirements for Bibliographic Records, 2009, 49-50, see https://repository.ifla.org/bitstream/123456789/811/2/ifla-functional-requirements-for-bibliographic-records-frbr.pdf.

[71] International Federation of Library Associations and Institutions, Functional Requirements for Authority Data, 2013, 17-18, see https://www.ifla.org/wp-content/uploads/2019/05/assets/cataloguing/frad/frad_2013.pdf.

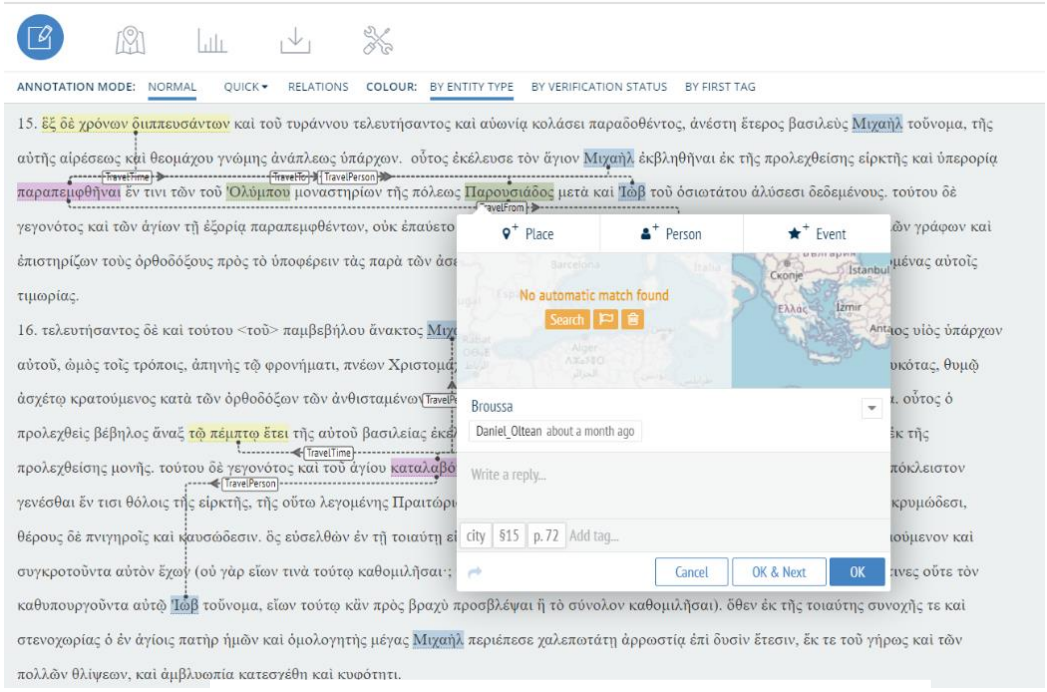


Fig. 6: Recogito annotation facilities

This process's final products are five files (in my notation, 1.csv, 2.csv, etc.), one for each of the five selected texts delivered by Recogito. They contain a copy of the annotation ("QUOTE_TRANSCRIPTION"), the information I introduced ("TYPE," "Subtype," etc.), the position in the text ("ANCHOR"), as well as unique identifiers for each annotation (UUID) and text.

In each Recogito workspace, four types of relationships, already mentioned, link the annotations referring to travels:

Fig. 7: Recogito relationships facilities

I permanently established the relationships' direction from the entity of type "EVENT" (= travel) to persons, places, and times:

Relation	**Source (EVENT)**	**Target (PERSON/PLACE/TIME)**
TravelPerson	John from Caesarea to Jerusalem	John, citizen of Caesarea
TravelFrom	John from Caesarea to Jerusalem	Caesarea, Palestine
TravelTo	John from Caesarea to Jerusalem	Jerusalem
TravelTime	John from Caesarea to Jerusalem	724

Ultimately, I downloaded the relationships as five files (in my notation, 1_Relations.csv, 2_Relations.csv, etc.), one for each selected text.

c. Data transformation: "Annotations" and "Entities" files

To obtain the files in the required format, I performed some transformations using OpenRefine. First, I worked on different columns of the five tables containing annotations (1.csv, 2.csv, etc.) and performed the following basic transformations:

- split column "TAGS" (defined by Recogito) into "Subtype," "Paragraph," and "Page/Column;"

- change the name of column "COMMENTS" with "Label_ENG;"

- change the name of column "FILE" (defined by Recogito) with "ID_Text" and put the number of the file in every cell;

- change the name of column "ANCHOR" (defined by Recogito) with "START" and remove the string "char-offset:" from values using GREL:replace(value, "char-offset:", "");

- add column "Length_quote" based on column "QUOTE_TRANSCRIPTION" using GREL:length(value);

- add column "STOP" based on column "START" using GREL:value + cells['Length_quote'].value -1";

- add value "TIME" in column "TYPE" for the entities of this type;

- add column “ID_Annotation;” as an identifier already exists (“UUID”), this column helped me only to manipulate data easier in the second phase of the project.

This phase’s first important problem was assigning an ID_Entity to each annotation; I added a column with that name. For the first annotated file (3.csv, chosen because it is the shortest one), ID_Annotation received the value of the index; for ID_Entity, I had to group the annotations by entities, i.e., “Label_ENG” column, to avoid assigning a new ID_Entity to an annotation referring to an entity that had already received an ID_Entity. Then, I gave ID_Entity the value of the index. Finally, I split the file into “Annotations.csv” and “Entities.csv” files with the following structures (I will discuss the latter in detail in the following section):

Annotations.csv
ID_Annotation
UUID
ID_Text
ID_Entity
QUOTE_TRANSCRIPTION
Length_quote
START
STOP
Paragraph
Page/Column

Table 4: Annotations.csv structure

Entities.csv
ID_Entity
Label_GR
Label_ENG
TYPE
Subtype
Wikidata

Table 5: Entities.csv structure

Wiki_URI
VIAF
VIAF_URI
TM
TM_URI
PmbZ
Pmbz_URI
Pleaides
Pleaiades_URI
GeoNames
GeoNames_URI
LAT
LNG

The following step consisted in adding the information from the following files (1.csv, 2.csv, 4.csv, 5.csv) to “Annotations.csv” and “Entities.csv” files. It was more complicated, especially because the annotations of the new files could refer (via “Label_ENG” column) to the entities already existing in “Entities.csv” file.

To identify those correlations, I added “ID_Entity” column in every new file using a beautiful function GREL:

cell.cross(“Entities csv”, “Label_ENG”).cells[“ID_Entity”].value[0].

In the case of, for example, 5.csv file, this function compares already existing “Entities.csv” and the new “5.csv” according to column “Label_ENG”, and when it finds the same value, it adds the value of “ID_Entity” from the first file to the second one.

For example, the annotations related to “Jerusalem” and “Justinian II” are treated differently in the function of the existence of such entity in "Entities" file:

Annotations from a new file (“5.csv")			**Already existing entity in “Entities.csv”**	**ID_ENTITY added in the new file (“5.csv")**
Label_ENG	**TYPE**	**Subtype**		

Jerusalem	PLACE	city	yes	12
Justinian II, Byzantine Emperor, 685-695; 705-711	PERSON	emperor	no	-

The next step was to add the new entities belonging, for example, to “5.csv,” to “Entities.csv” file. I ascribed the following notations:

- df_old_Entities = table of entities belonging to the previous steps (= “Entities.csv”);

- df_new = new table (= “5.csv”);

- df_ent = table of new entities extracted from df_new (with “ent” = number of the new entities);

- New_Entities.csv = new (test) “Entities” file including the new entities;

- New_5.csv = new (test) “5.csv” file, including the ID_Entity of each annotation.

The code associated with this transformation was the following:

```
df_old_Entities = pd.read_csv('C:/Users/danie/Desktop/DH 2022-2023/MasterThesis/Texts Thesis/Entities.csv')

df_new = pd.read_csv('C:/Users/danie/Desktop/DH 2022-2023/MasterThesis/Texts Thesis/5.csv')

df_ent = df_new.loc[df_new["ID_Entity"].isna()].groupby("Label_ENG").count()
df_ent.reset_index(inplace=True)
ent = len(df_ent)

index = 0
while index < ent:
    entity = df_ent.loc[index, "Label_ENG"]
    position = len(df_old_Entities)
    quote = df_new.loc[df_new["Label_ENG"] == entity]["QUOTE_TRANSCRIPTION"].iloc[0]
    TYPE = df_new.loc[df_new["Label_ENG"] == entity]["TYPE"].iloc[0]
    Subtype = df_new.loc[df_new["Label_ENG"] == entity]["Subtype"].iloc[0]
    my_list = [str(position+1), quote, entity, TYPE, Subtype]
    df_old_Entities.loc[position, ["ID_Entity", "Label_GR", "Label_ENG", "TYPE", "Subtype"]] = my_list
    df_new.loc[df_new["Label_ENG"] == entity, "ID_Entity"] = str(position+1)
    index += 1

df_old_Entities.to_csv('C:/Users/danie/Desktop/DH 2022-2023/MasterThesis/Texts Thesis/New_Entities.csv')

df_new.to_csv('C:/Users/danie/Desktop/DH 2022-2023/MasterThesis/Texts Thesis/New_5.csv')
```

For the new entities, the results are of the following type:

Annotations from a new file (“5.csv")			Already existing entity in “Entities.csv”	ID_ENTITY added in the new file (“5.csv")
Label_ENG	**TYPE**	**Subtype**		
Justinian II, Byzantine Emperor, 685-695; 705-711	PERSON	emperor	no	296

I used a similar method to add the new annotations (now containing “ID_Entity” column) to the already existing “Annotations” file. I used the following notations:

- df_old_ Annotations = table of annotations belonging to the previous steps (= Annotations.csv);

- df_new = new table (= 5.csv, with “ann” = number of the new annotations);

- New_Annotations.csv = new (test) “Annotations” file, including the new annotations.

The code associated with this transformation was the following:

```
df_old_Annotations = pd.read_csv('C:/Users/danie/Desktop/DH 2022-2023/MasterThesis/Texts Thesis/Annotations.csv')

ann = len(df_new)

index = 0
while index < ann:
    UUID = df_new.loc[index, "UUID"]
    ID_Text = df_new.loc[index, "ID_Text"]
    ID_Entity = df_new.loc[index, "ID_Entity"]
    QUOTE_TRANSCRIPTION = df_new.loc[index, "QUOTE_TRANSCRIPTION"]
    Length_quote = df_new.loc[index, "Length_quote"]
    START = df_new.loc[index, "START"]
    STOP = df_new.loc[index, "STOP"]
    Paragraph = df_new.loc[index, "Paragraph"]
    Page = df_new.loc[index, "Page/Column"]
    position = len(df_old_Annotations)
    my_list = [str(position+1), UUID, ID_Text, ID_Entity, QUOTE_TRANSCRIPTION, Length_quote, START, STOP, Paragraph, Page]
    df_old_Annotations.loc[position, ["ID_Annotation", "UUID", "ID_Text", "ID_Entity", "QUOTE_TRANSCRIPTION", "Length_quote", "START", "STOP", "Paragraph",
    index += 1

df_old_Annotations.to_csv('C:/Users/danie/Desktop/DH 2022-2023/MasterThesis/Texts Thesis/New_Annotatios.csv')
```

I preferred to repeat those steps and that code for each of the five files to easier check the changes that occurred. I could obtain the same result by only one reading of the five files.

d. Data transformation: “Relations” file

I used a similar piece of code to group the five files containing relationships (1_Relations.csv, 2_Relations.csv, etc.) in one final file called “Relations,” with the following structure:

Relations.csv
relation
source
source_UUID
target
taget_UUID

Table 6: Relations.csv structure

The two UUIDs guarantee the link between the relations and annotations.

The number of entries of “Annotations,” “Entities,” and “Relations” files obtained after steps c. and d. is the following:

File	Annotations	Entities	Relations
1	330	110	59
2	354	163	143
3	34	25	12
4	47	33	20
5	45	25	24
Total	**810**	**318**	**258**

Table 7: Total number of entities

e. Improving data in “Entities” file

In the last step, I checked and improved the information in “Entities” file referring to the entries of type “Person” and “Place,” using OpenRefine. First, I modified “Label_GR” column to contain the word in the nominative case. Second, I introduced the following new columns with some standard identifiers of the entities:

- “Wikidata” and “Wiki_URI” = Wikidata identifier, based on reconciliation service;
- “VIAF” and “VIAF_URI” = VIAF[72] (= Virtual International Authority File) identifier;
- “TM” and “TM_URI” = Trismegistos[73] identifier;
- “PmbZ” and “PmbZ_URI” = PmbZ[74] (= Prosopographie der mittelbyzantinischen Zeit) identifier (only for persons);

[72] https://viaf.org/
[73] https://www.trismegistos.org/
[74] https://www.degruyter.com/database/pmbz/html?lang=en

- “Pleiades” and “Pleiades_URI” = Pleiades[75] identifier (only for places);
- “GeoNames,” “GeoNames_URI,” “LAT,” and “LNG” = GeoNames[76] identifier and geographical coordinates.

For a significant number of entries (of type “Place and “Person”), I assigned the Wikidata identifier.

Entity Type	**Entities**	**Wikidata identifier**
Person	136	94
Place	95	81
Event	57	-
Time	30	-
Total	**318**	**175**

Table 8: Number of entities identified in Wikidata

2.1.6. Results

At the end of this task, the dataset was available. It contains “Texts,” “Annotations,” “Entities,” and “Relations” files, which I used in the following step of the project.

[75] https://pleiades.stoa.org/
[76] https://www.geonames.org/

2.2. Task 2: Data testing and querying (Neo4j)

During the second phase, I transferred the data obtained in the previous step in Neo4j format and wrote two procedures that simulate named entity recognition and Wikidata identifiers recognition. The main steps of this task were the following:

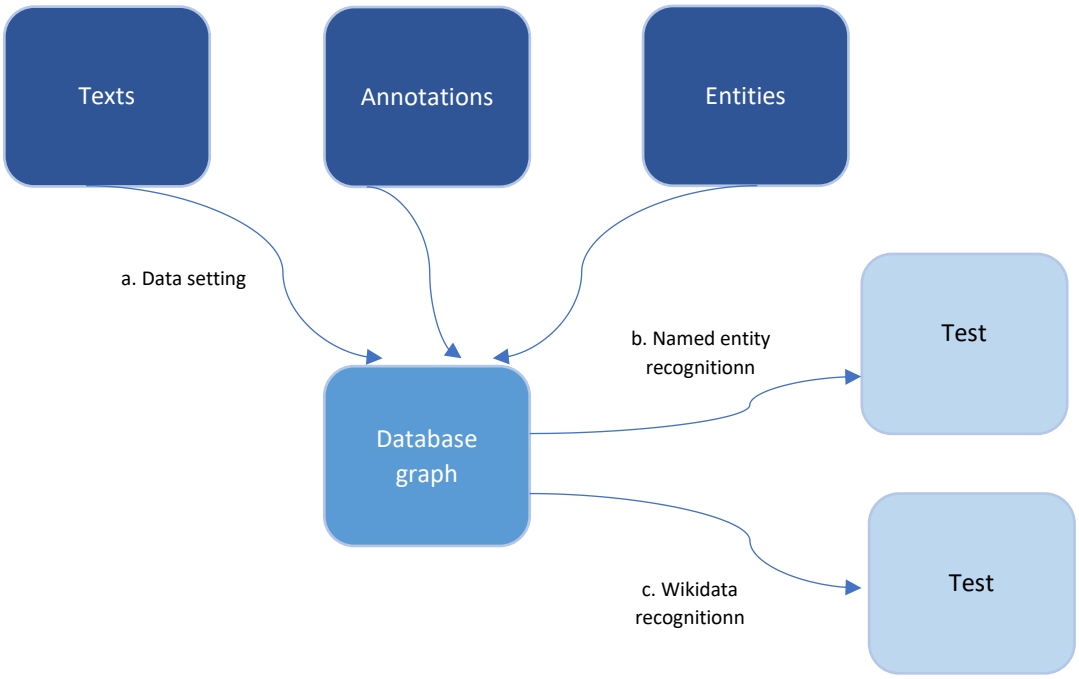


Fig. 8: Querying task pipeline

2.2.1. Objectives

This task aimed to explore the dataset in the Neo4j environment and provide shorts Cypher codes that simulate useful functionalities. There were three main components of this process:

a. importing tabular data in Neo4j environment;
b. recognition of named entities when a new text is added to the graph;
c. recognition of the Wikidata identifier of a new entity added to the graph.

2.2.2. Tools

In the project's second part, I used the Neo4j application (both Desktop and AuraDB versions). This tool allows importing tabular data, creating graph databases, and querying data using Cypher, a powerful declarative language. The application includes APOC (i.e., Awesome Procedures on Cypher),[77] a remarkable library containing

[77] https://neo4j.com/labs/apoc/

hundreds of useful functions. The plugin neosemantics (n10s) enables importing and using data of RDF type in Neo4j projects.[78]

2.2.3. Methodology

To accomplish this task, I used three main methods:

a. Data import

This method refers to the transfer of tabular data (.csv format) in Neo4j environment. I imported the files "Texts" (5 entries), "Annotations" (810 entries), and "Entities" (318 entries) by creating three categories of nodes of types:

- "Text;"
- "Annotation;"
- "Entity."

Those 1133 nodes were linked by creating 1620 properties (= 2x810, two for each annotation) of types:

- "contains," which connects a node of type "Text" with another of type "Annotation;"
- "references," which connects a node of type "Annotation" with another of type "Entity."

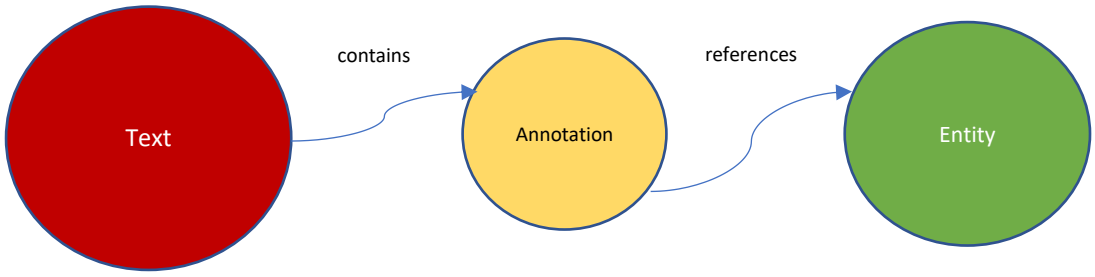


Fig. 9: Graph-database nodes and edges

All the columns of those files were imported as node properties in the graph.

b. Data querying

To simulate results, I used basic Cypher commands, such as CREATE, MATCH, WHERE, WITH, and RETURN, which could be correlated with similar SQL ones, as well as some useful APOC functions, such as apoc.text.indexesOf(),

[78] https://neo4j.com/labs/neosemantics/

apoc.text.replace(), and apoc.text.split(), and apoc.text.compareCleaned, which are similar to Java and Python methods with comparable names.

c. LPG-LOD connections

To obtain Wikidata identifiers, I wrote SPARQL commands integrated into Cypher. The results were imported in JSON format using apoc.load.jsonParam() function.

2.2.4. Workflow

a. Data setting

I imported data in Neo4j as nodes of three categories (“Text,” “Annotation,” and “Entity”), related by two properties (“contains” and “references”). Files’ columns became node properties. After copying .csv files into “Import” folder, I used the following code:

```
//load texts
LOAD CSV WITH HEADERS FROM 'file:///Texts.csv' AS line_txt
CREATE (t:Text)
SET t += line_txt
//load annotations
WITH *
LOAD CSV WITH HEADERS FROM 'file:///Annotations.csv' AS line_ann
CREATE (a:Annotation)
SET a += line_ann
//load entities
WITH *
LOAD CSV WITH HEADERS FROM 'file:///Entities.csv' AS line_ent
CREATE (e:Entity)
SET e += line_ent
//create relationships
WITH *
MATCH (t:Text), (a:Annotation), (e:Entity)
WHERE a.ID_Text = t.ID_Text AND a.ID_Entity = e.ID_Entity
CREATE (t)-[rel1:contains]->(a)-[rel2:references]->(e)
RETURN t
```

With this code, I created 1133 (=5+810+318) nodes and 1620 (2x810) edges. All the files’ columns became nodes’ properties. To display entities, I chose their ID as identifiers (ID_Text; ID_Annotation; ID_Entity) and a distinct colour for each of us (texts = red; annotations = yellow; entities = green), like in the following example:

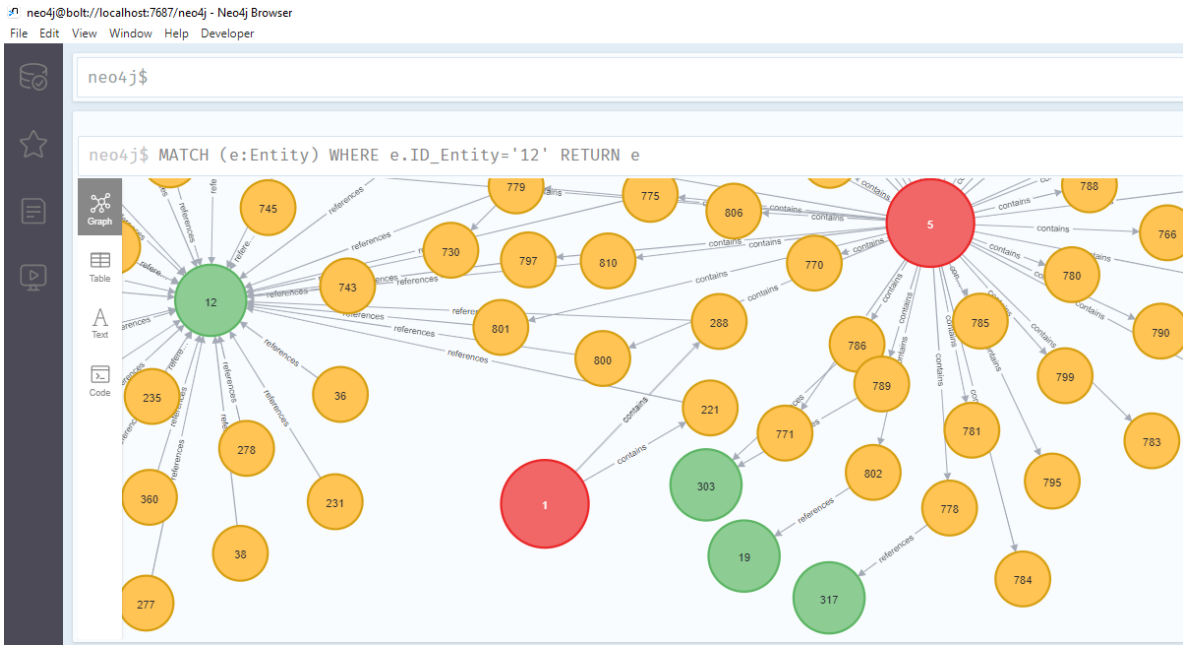


Fig. 10: Graph-database partial visualisation

b. Named entity recognition

How could one use this graph to test named entity recognition? When a new text is added to the graph, the pipeline will first contain the tokenisation of the text and selection of the words beginning with capitals, i.e., the proper names. Then, each proper name will be compared with existing entities. In our case, the comparison must be done between the Greek word found in texts and the Greek label of each entity, i.e., “Label_GR.”

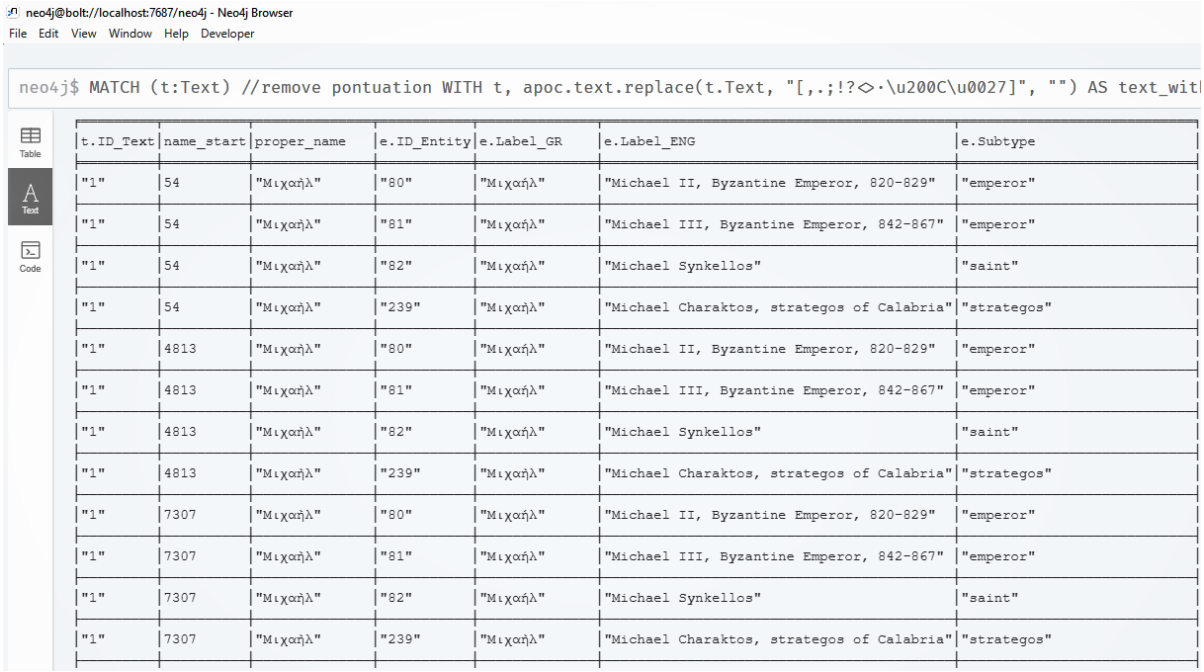
To check the validity of my simulation, I did not introduce new texts in this graph but matched the existing ones with the existing entities. A way to accomplish this task is the following code:

```
MATCH (t:Text)
//remove punctuation
WITH t, apoc.text.replace(t.Text, "[,.;!?<>·\u0027\u200C]", "") AS text_without_punctuation //split the text in distinct words
WITH t, apoc.coll.toSet(apoc.text.split(text_without_punctuation, ' ')) AS list_words
//check if the word is a proper name
// it can begin with capitals containing accents and breathings
UNWIND list_words AS word
WITH t, word AS proper_name
WHERE proper_name =~ "^[A-Ω\u1F08-\u1F0F\u1F38-\u1F3F\u1F68-\u1F6F\u1FEC].*$"
```

```
//match entities containing the same proper name
MATCH (e:Entity)
WHERE apoc.text.compareCleaned(e.Label_GR, proper_name) = TRUE
//for each name, find all the apparitions in texts
WITH t, e, proper_name, apoc.text.indexesOf(t.Text, proper_name) AS List_Starts
UNWIND List_Starts AS name_start
RETURN t.ID_Text, name_start, proper_name, e.ID_Entity, e.Label_GR, e.Subtype
```

With the function apoc.text.replace(), I removed punctuation from text, including apostrophes and zero width non-joiner character; after tokenisation, I selected only the distinct words with apoc.coll.toSet() to improve the procedure's time of execution; I limited the list to the proper names by checking them against a regex pattern that includes the Unicode characters used in the Greek alphabet (not only A-Ω but also polytonic orthography such as Ἀ, Ἁ, Ἂ, Ἃ, Ἄ, Ἅ, Ἆ, Ἇ for the first vowel; I limited the list of character to the necessary pattern); finally, I identified match entities with apoc.text.compareCleaned(), which ignore accents and breathings in the compared texts.

This query returns 648 results; the first ones are the following:



t.ID_Text	name_start	proper_name	e.ID_Entity	e.Label_GR	e.Label_ENG	e.Subtype
"1"	54	"Μιχαήλ"	"80"	"Μιχαήλ"	"Michael II, Byzantine Emperor, 820-829"	"emperor"
"1"	54	"Μιχαήλ"	"81"	"Μιχαήλ"	"Michael III, Byzantine Emperor, 842-867"	"emperor"
"1"	54	"Μιχαήλ"	"82"	"Μιχαήλ"	"Michael Synkellos"	"saint"
"1"	54	"Μιχαήλ"	"239"	"Μιχαήλ"	"Michael Charaktos, strategos of Calabria"	"strategos"
"1"	4813	"Μιχαήλ"	"80"	"Μιχαήλ"	"Michael II, Byzantine Emperor, 820-829"	"emperor"
"1"	4813	"Μιχαήλ"	"81"	"Μιχαήλ"	"Michael III, Byzantine Emperor, 842-867"	"emperor"
"1"	4813	"Μιχαήλ"	"82"	"Μιχαήλ"	"Michael Synkellos"	"saint"
"1"	4813	"Μιχαήλ"	"239"	"Μιχαήλ"	"Michael Charaktos, strategos of Calabria"	"strategos"
"1"	7307	"Μιχαήλ"	"80"	"Μιχαήλ"	"Michael II, Byzantine Emperor, 820-829"	"emperor"
"1"	7307	"Μιχαήλ"	"81"	"Μιχαήλ"	"Michael III, Byzantine Emperor, 842-867"	"emperor"
"1"	7307	"Μιχαήλ"	"82"	"Μιχαήλ"	"Michael Synkellos"	"saint"
"1"	7307	"Μιχαήλ"	"239"	"Μιχαήλ"	"Michael Charaktos, strategos of Calabria"	"strategos"

The procedure finds a Μιχαὴλ/Michael in the first text at the position "name_start" = 54 and matches it with the four Μιχαήλ/Michael that define the entities nos. 80 (Michael II, Byzantine Emperor, 820-829), 81 (Michael III, Byzantine Emperor, 842-867), 82 (Michael Synkellos), and 239 (Michael Charaktos, strategos of Calabria). The user

could choose the right option among those four. The code provides this type of result for all proper names found in texts.

However, this method has limits. First, in ancient Greek texts, the proper names do not always begin with capitals. Second, the words are often inflected. I ignored both aspects in the next considerations.

In this general framework, I limited my simulation to only one existing text, identified by:

ID_Text = 3 (the shortest one)

In the existing graph, this text relates to 8 entities of type “PERSON” or “PLACE” through 11 annotations, identified by this search:

```
MATCH (t:Text)-[rel1:contains]->(a:Annotation)-[rel2:references]->(e:Entity)
WHERE t.ID_Text = "3" AND apoc.text.compareCleaned(e.Label_GR,
a.QUOTE_TRANSCRIPTION) = TRUE AND e.TYPE IN ["PERSON", "PLACE"]
RETURN e, a, t, rel1, rel2
```

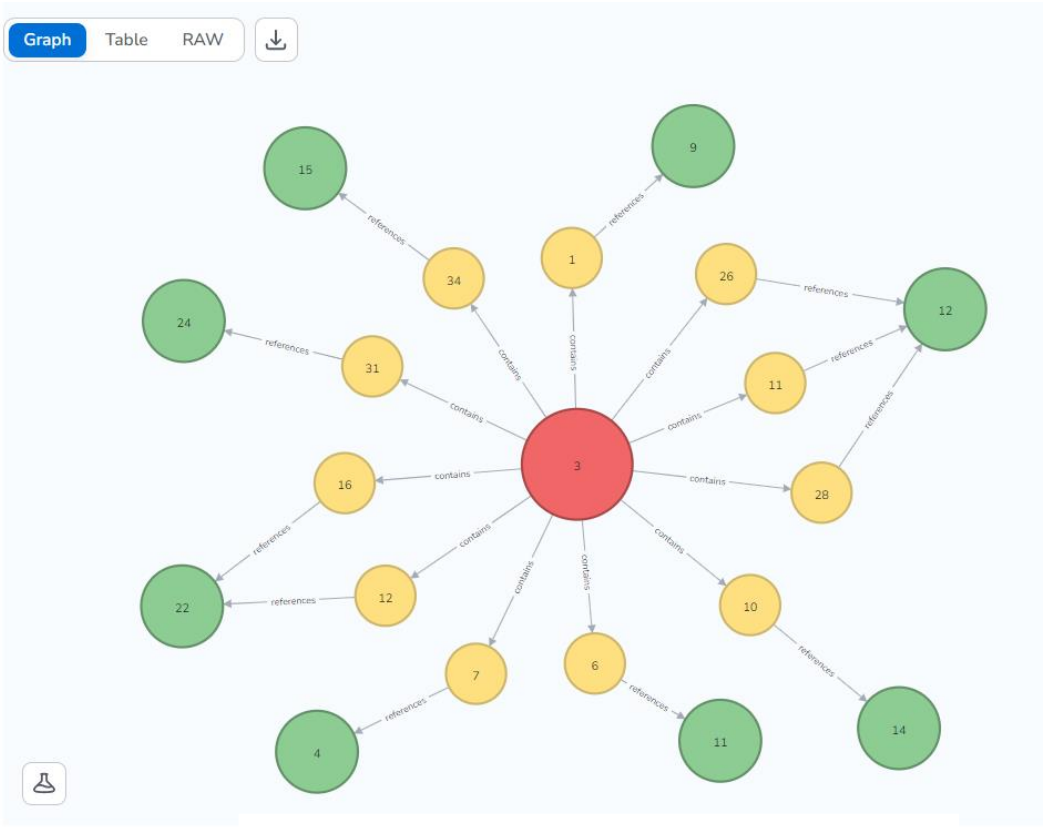
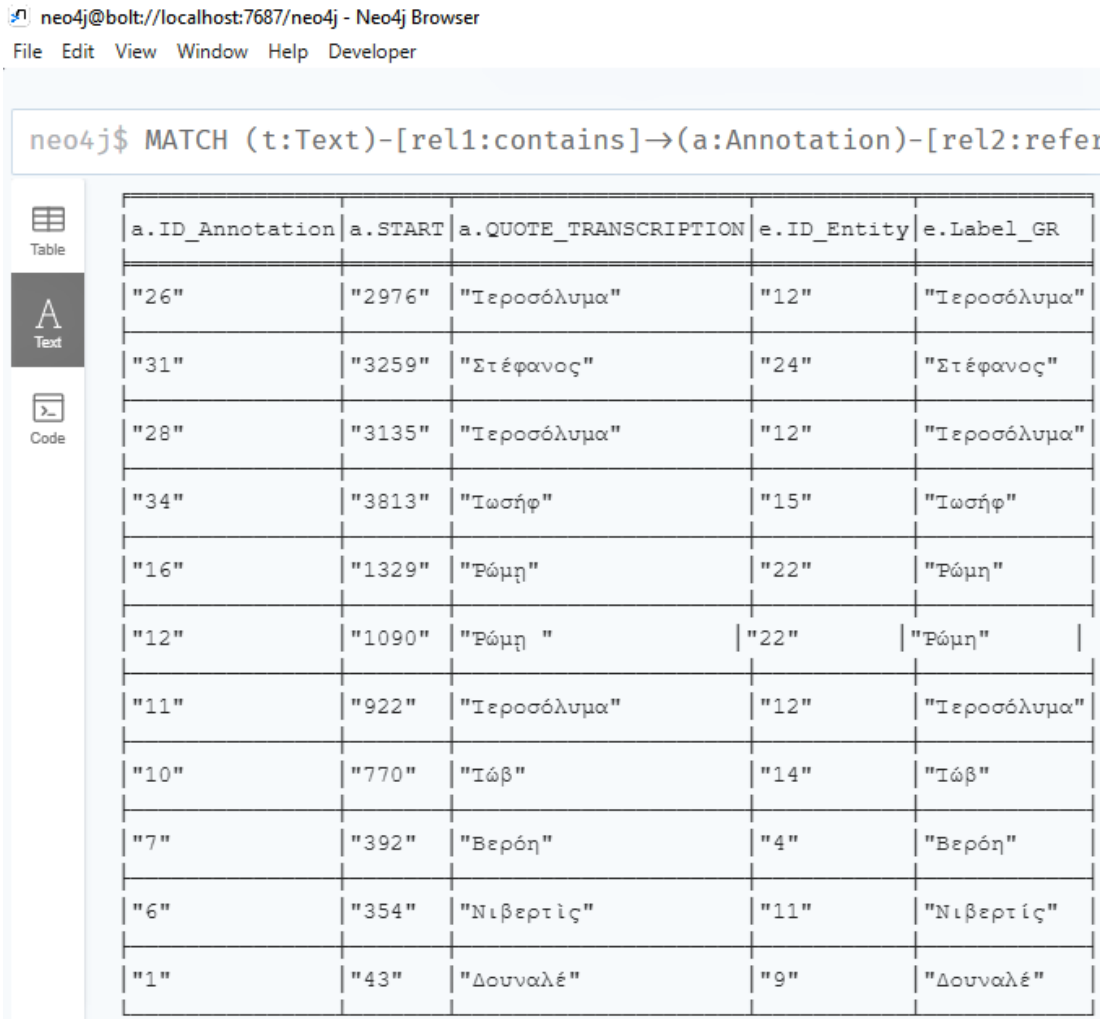


Fig. 11: Graph visualisation of the expected results

The already existing entities, annotations, and start positions are the following:

```
MATCH (t:Text)-[rel1:contains]->(a:Annotation)-[rel2:references]->(e:Entity)
WHERE t.ID_Text = "3" AND apoc.text.compareCleaned(e.Label_GR,
a.QUOTE_TRANSCRIPTION) = TRUE AND e.TYPE IN ["PERSON", "PLACE"]
RETURN a.ID_Annotation, a.START, a.QUOTE_TRANSCRIPTION, e.ID_Entity,
e.Label_GR
```



a.ID_Annotation	a.START	a.QUOTE_TRANSCRIPTION	e.ID_Entity	e.Label_GR
"26"	"2976"	"Ιεροσόλυμα"	"12"	"Ιεροσόλυμα"
"31"	"3259"	"Στέφανος"	"24"	"Στέφανος"
"28"	"3135"	"Ιεροσόλυμα"	"12"	"Ιεροσόλυμα"
"34"	"3813"	"Ιωσήφ"	"15"	"Ιωσήφ"
"16"	"1329"	"Ῥώμῃ"	"22"	"Ῥώμη"
"12"	"1090"	"Ῥώμῃ "	"22"	"Ῥώμη"
"11"	"922"	"Ιεροσόλυμα"	"12"	"Ιεροσόλυμα"
"10"	"770"	"Ἰώβ"	"14"	"Ἰώβ"
"7"	"392"	"Βερόη"	"4"	"Βερόη"
"6"	"354"	"Νιβερτὶς"	"11"	"Νιβερτίς"
"1"	"43"	"Δουναλέ"	"9"	"Δουναλέ"

Table 9: Table of the expected results

Those are the results the automatic recognition of named entities must find when Text no. 3 is considered a new one added to the graph.

In this respect, I used the previous Cypher code. However, to simplify this simulation, I skipped the interaction with the user to check which entities refer to the text and considered only the matches corresponding to text no. 3 (by imposing the condition toInteger(e.ID_Entity) <= 25).

The following code creates new annotations for each proper name found in the text that refers to an already existing entity:

```
MATCH (t:Text)
WHERE t.ID_Text ="3"
//remove puntuation
WITH t, apoc.text.replace(t.Text, "[,.;!?<>·\u200C\u0027]", "") AS
text_without_punctuation
//split the text in distinct words
WITH t, apoc.coll.toSet(apoc.text.split(text_without_punctuation, ' ')) AS list_words
//check if the word is a proper name
UNWIND list_words AS word
WITH t, word AS proper_name
WHERE proper_name =~ "^[A-Ω\u1F08-\u1F0F\u1F38-\u1F3F\u1F68-
\u1F6F\u1FEC].*$"
//match entities with the same name
MATCH (e:Entity)
WHERE apoc.text.compareCleaned(e.Label_GR, proper_name) = TRUE AND
toInteger(e.ID_Entity) <= 25
//for each name, find all the apparitions in texts
WITH t, e, proper_name, apoc.text.indexesOf(t.Text, proper_name) AS List_Starts
UNWIND List_Starts AS name_start
CALL {
      WITH t, proper_name, e, name_start
      MATCH (a:Annotation)
      WITH t, proper_name, e, name_start, COUNT(a)+1 AS new_a
      CREATE (new_A:Annotation {
         ID_Annotation:toString(new_a),
         ID_Text:t.ID_Text,
         ID_Entity:e.ID_Entity,
         QUOTE_TRANSCRIPTION:proper_name,
         START:name_start
      })
      CREATE rel1 = (t)-[:contains]->(new_A)
      CREATE rel2 = (new_A)-[:references]->(e)
      RETURN new_A, rel1, rel2
   }
RETURN t, e, new_A, rel1, rel2
```

I displayed the results first as a subgraph and compared it with the expected one (see above):

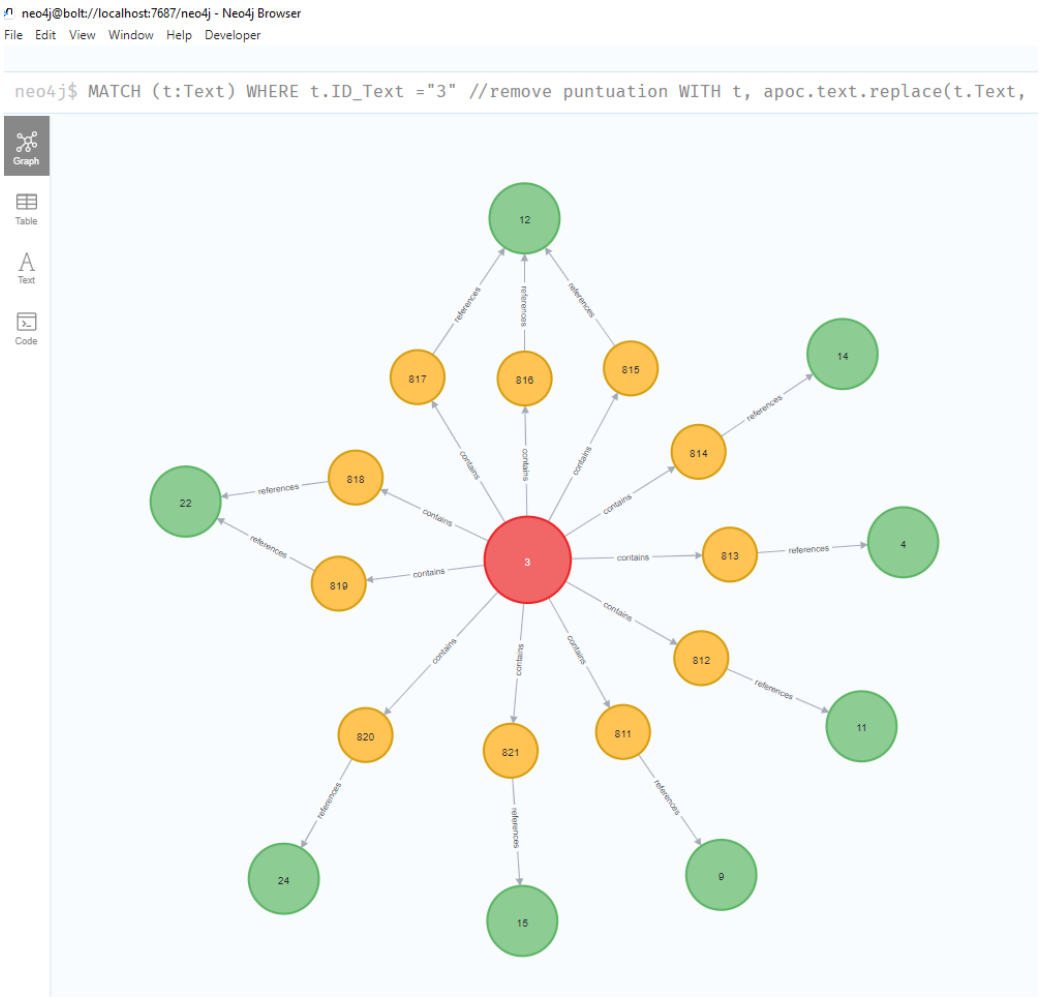
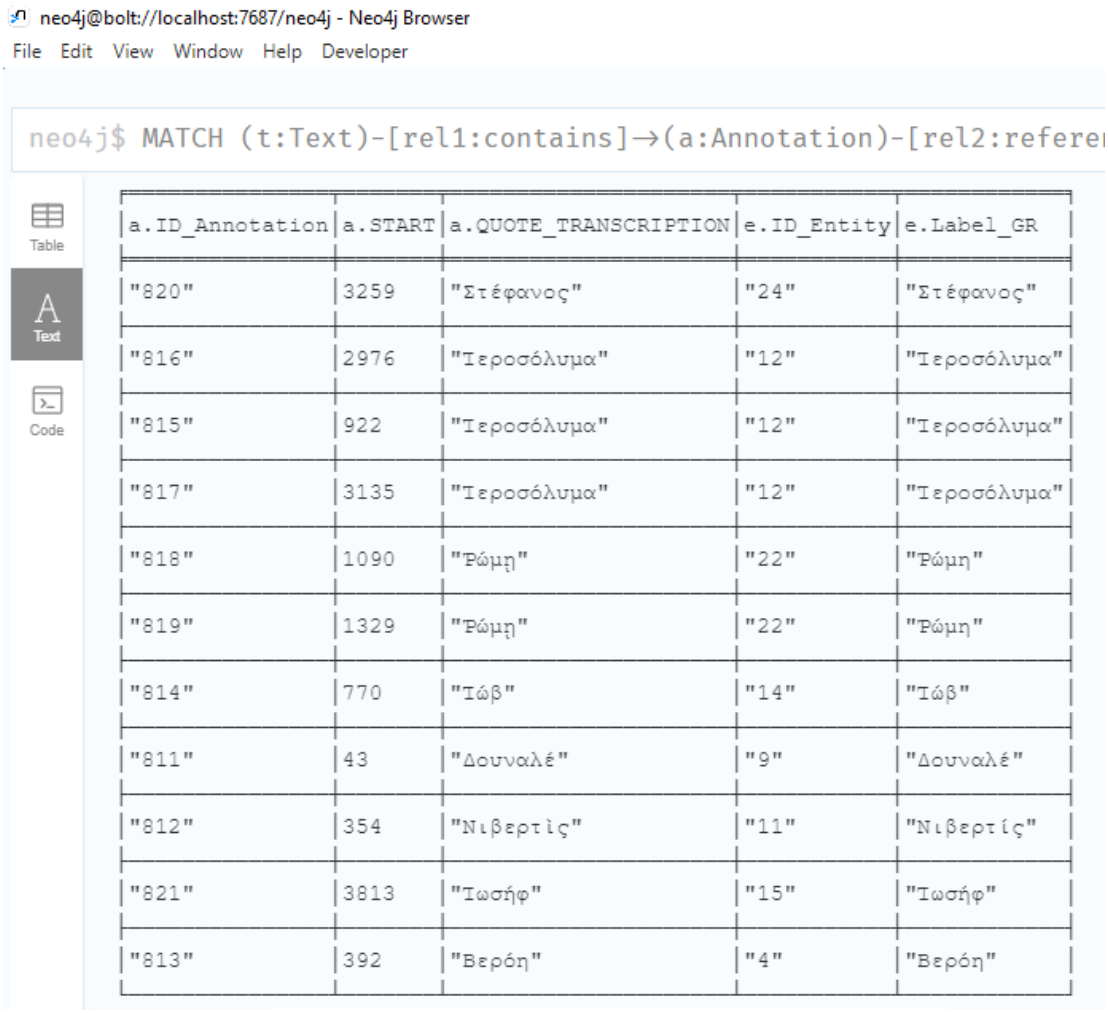


Fig. 12: Graph visualisation of the results

As expected, the subgraph is identical to that previously displayed. The code created 11 new annotations (ID_Annotation = 811-821) and 22 new relationships. The start positions are also identical to the previous ones:

```
MATCH (t:Text)-[rel1:contains]->(a:Annotation)-[rel2:references]->(e:Entity)
WHERE a.ID_Annotation >= 811
RETURN a.ID_Annotation, a.START, a.QUOTE_TRANSCRIPTION, e.ID_Entity,
e.Label_GR
```



a.ID_Annotation	a.START	a.QUOTE_TRANSCRIPTION	e.ID_Entity	e.Label_GR
"820"	3259	"Στέφανος"	"24"	"Στέφανος"
"816"	2976	"Ιεροσόλυμα"	"12"	"Ιεροσόλυμα"
"815"	922	"Ιεροσόλυμα"	"12"	"Ιεροσόλυμα"
"817"	3135	"Ιεροσόλυμα"	"12"	"Ιεροσόλυμα"
"818"	1090	"Ῥώμῃ"	"22"	"Ῥώμη"
"819"	1329	"Ῥώμῃ"	"22"	"Ῥώμη"
"814"	770	"Ἰώβ"	"14"	"Ἰώβ"
"811"	43	"Δουναλέ"	"9"	"Δουναλέ"
"812"	354	"Νιβερτὶς"	"11"	"Νιβερτίς"
"821"	3813	"Ἰωσήφ"	"15"	"Ἰωσήφ"
"813"	392	"Βερόη"	"4"	"Βερόη"

Table 10: Table of the results

Therefore, this procedure checks the graph and correctly finds all the entities that refer to the proper names of a new text (except the inflected names).

c. Wikidata recognition

When a new text is added to the graph, the proper names that do not match existing entities are defined as new entities. The user must manually add information concerning the standard Greek name (“Label_GR”), English name (“Label_ENG”), type, and subtype of the entity, etc. However, the automatic recognition of the Wikidata identifier is possible (if it exists). The user could choose from a list of suggestions.

To simulate this facility, I selected one entity that does not raise supplementary complication (same word in ancient and modern Greek), defined by its “Label_GR,” “Δαμασκός /Damascus” (obviously, the search by “Label_ENG” is also possible). This city is already present in the graph database in the following format:

```
$ MATCH (e:Entity) WHERE e.Label_GR = "Δαμασκός" RETURN e
```

Graph | Table | RAW

e


```
{
  identity: 1103,
  labels: ["Entity"],
  properties: {
    GeoNames_URI: "https://www.geonames.org/170654/damascus.html",
    ID_Entity: 289,
    Wiki_URI: "https://www.wikidata.org/wiki/Q3766",
    LNG: "36.29128",
    Label_GR: "Δαμασκός",
    TM_URI: "https://www.trismegistos.org/place/533",
    Label_ENG: "Damascus",
    VIAF: "146573830",
    VIAF_URI: "https://viaf.org/viaf/146573830/",
    TM: "TM Geo 533",
    Pleiades_URI: "https://pleiades.stoa.org/places/678106",
    Wikidata: "Q3766",
    GeoNames: "170654",
    Pleiades: "678106",
    TYPE: "PLACE",
    Subtype: "city",
    LAT: "33.5102"
  },
  elementId: "4:bc279d25-0a98-4fa1-88d2-c204dca0ac9c:1103"
}
```


Previously, in the first step of the project, I had added the Wikidata identifier (Q3766).

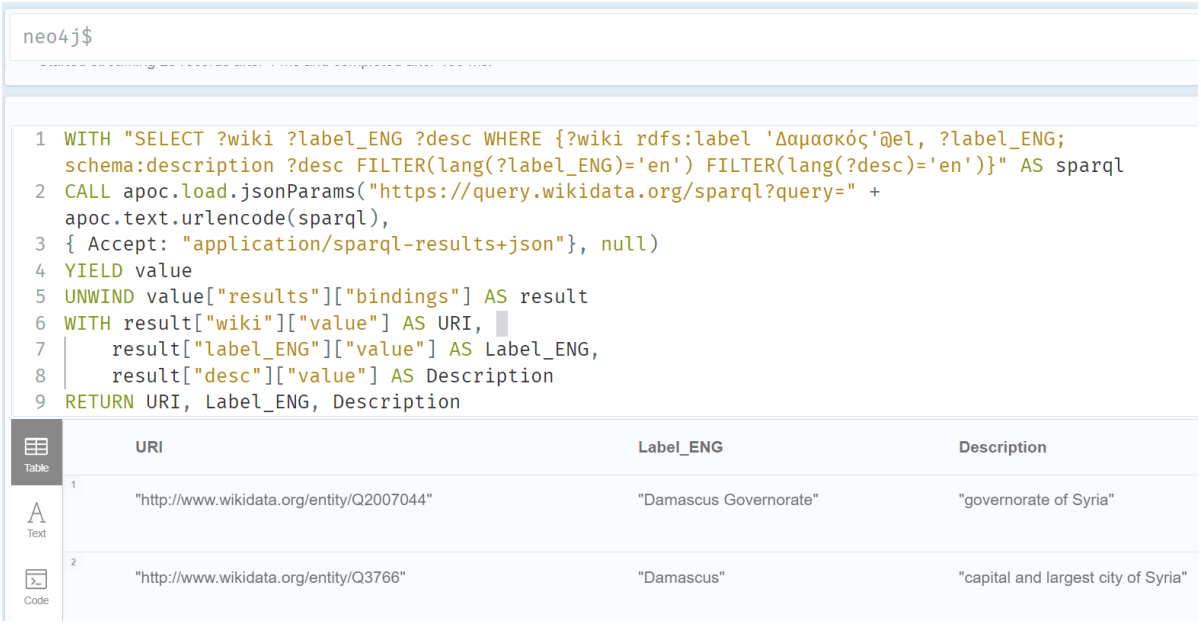
I considered that the word “Δαμασκός” would have been found in a (new) text (in our case, ID_Text = 4). The following procedure uses SPARQL code to identify Wikidata entities with this label. The results are imported in Neo4j in JSON format and displayed as a list:

```
// write SPARQL query
WITH "SELECT ?wiki ?label_ENG ?desc WHERE {?wiki rdfs:label 'Δαμασκός'@el,
?label_ENG; schema:description ?desc FILTER(lang(?label_ENG)='en')
FILTER(lang(?desc)='en')}" AS sparql
// execute SPARQL query in Neo4j environment
CALL apoc.load.jsonParams("https://query.wikidata.org/sparql?query=" +
apoc.text.urlencode(sparql),
{ Accept: "application/sparql-results+json"}, null)
YIELD value
```

```
// select results
UNWIND value["results"]["bindings"] AS result
WITH result["wiki"]["value"] AS URI,
    result["label_ENG"]["value"] AS Label_ENG,
    result["desc"]["value"] AS Description
RETURN URI, Label_ENG, Description
```

In this code, I first formulated a SPARQL query as a parameter, which uses “Label_GR” to identify Wikidata identifier, as well as the English name and short description of the identified items. The function apoc.load.jsonParams() executes the query in Neo4j. The results are in the following JSON format:

The procedure finds two entities labelled Δαμασκός in Greek. To choose between them, the user can check the English name and description:



The second option, Damascus as the “capital and largest city of Syria,” is the correct; the first refers to the “governorate of Syria.” The right (second) “URI” can now be introduced as a new property of the entity “Δαμασκός /Damascus.”

This procedure can be applied to all new entities added to the graph.

2.2.5. Results

This second task provided a general framework and some procedures that could be useful for future research. The dataset was tested for particular and sometimes isolated cases. However, the proposed codes give the same outcomes in the case of more significant texts. In my opinion, the primary aspect that must be considered remains the words’ inflection, a peculiarity of the ancient Greek language that creates difficulties in finding data and querying the graph database. I attached the Neo4j database in .dump format.

Conclusion

The current research focused on different methods and data models used in textual annotation and recognition. It revealed some motivating aspects, especially related to the particularities of the Greek texts.

First, the traditional (manual) text annotation method allowed to me explore this beneficial domain more deeply. With Recogito, I selected entities of interest (persons, places, events) from some Greek texts and explored them in the traditional tabular format with OpenRefine.

Second, I also investigated data in the graph-based format using Neo4j. My selected and already annotated Greek texts will act as test cases. I suggested some Cypher procedures that simulate named entity recognition when a new text is added to the graph database.

Finally, I studied the domain of open-linked data in connection with the existing graph dataset. I suggested a method of recognising Wikidata identifiers using SPARQL queries integrated into the Neo4j environment.

Abbreviations

BHG = Bibliotheca Hagiographica Graeca

DH = Digital Humanities

GREL = General/Google Refine Expression Language

HTTP = Hypertext Transfer Protocol

JSON = JavaScript Object Notation

LOD = Linked Open Data

LPG = Labelled Property Graph

NEL = Named Entity Linking

Neo4j = Network Exploration and Optimisation for Java

NER = Named Entity Recognition

NLP = Natural Language Processing

NoSQL = non-/not only-SQL

OCR = Optical Character Recognition

PmbZ = Prosopographie der mittelbyzantinischen Zeit

RDF = Resource Description Framework

SPARQL = SPARQL Protocol and RDF Query Language

SQL = Structured Query Language

URI = Uniform Resource Identifier

UUID = Universally Unique Identifier

VIAF = Virtual International Authority File

W3C = World Wide Web Consortium

Selected bibliography

Abadie et al., 2022 = N. Abadie et al., A Benchmark of Named Entity Recognition Approaches in Historical Documents Application to 19th Century French Directories, in S. Uchida et al., Document Analysis Systems, DAS 2022, La Rochelle, France, Springer, 2022, 445-460, https://doi.org/10.1007/978-3-031-06555-2_30

Barrasa, 2017 = Jesús Barrasa, RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?, 2017, https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/?ref=blog

Berners-Lee, 2006 = Tim Berners-Lee, Linked Data, 2006, https://www.w3.org/DesignIssues/LinkedData.html

Berti, 2019 = M. Berti, Named Entity Annotation for Ancient Greek with INCEpTION, in CLARIN Annual Conference Proceedings, 2019, 1-4, https://office.clarin.eu/v/CE-2019-1512_CLARIN2019_ConferenceProceedings.pdf

Boutsis et al., 2000 = Sotiris Boutsis et al., A System for Recognition of Named Entities in Greek, in D.N. Christodoulakis (ed.) Natural Language Processing — NLP 2000, Springer, 2000, https://doi.org/10.1007/3-540-45154-4_39

Brandsen et al., 2020 = Alex Brandsen et al., Creating a Dataset for Named Entity Recognition in the Archaeology Domain, in Proceedings of the 12th Conference on Language Resources and Evaluation, LREC 2020, 4573-4577, https://aclanthology.org/2020.lrec-1.562/

Ehrmann et al., 2016 = Maud Ehrmann et al., Diachronic evaluation of NER systems on old newspapers, in Proceedings of the 13th Conference on Natural Language Processing, Bochum, 2016, 97-107, https://infoscience.epfl.ch/record/221391/files/13_konvensproc.pdf

Ehrmann et al., 2021 = Maud Ehrmann et al., Named Entity Recognition and Classification on Historical Documents: A Survey, arXiv, 2021, https://arxiv.org/pdf/2109.11406.pdf

Elmasri and Navathe, 2016 = Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, Boston, 2016, https://amirsmvt.github.io/Database/Static_files/Fundamental_of_Database_Systems.pdf

Fickers and Tatarinov, 2022 = Andreas Fickers and Juliane Tatarinov (eds.), Digital History and Hermeneutics: Between Theory and Practice, Berlin, 2022, https://doi.org/10.1515/9783110723991

Garz et al., 2016 = A. Garz et al., Creating Ground Truth for Historical Manuscripts with Document Graphs and Scribbling Interaction, in Proceedings of 12th IAPR Workshop on Document Analysis Systems, DAS 2016, 126-131,

https://diuf.unifr.ch/main/hisdoc/sites/diuf.unifr.ch.main.hisdoc/files/uploads/hisdoc2.0-publications/DAS2016_Garz_GT.pdf

Fantoli et al., 2022 = Margherita Fantoli et al., Linking the LASLA Corpus in the LiLa Knowledge Base of Interoperable Linguistic Resources for Latin, in Proceedings of the Linked Data in Linguistics Workshop, LREC 2022, 26-34, https://aclanthology.org/2022.ldl-1.4.pdf

Johnson et al., 2021 = Kyle P. Johnson et al., The Classical Language Toolkit: An NLP Framework for Pre-Modern Languages, in Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, 2021, https://doi.org/10.18653/v1/2021.acl-demo.3

Halkin, 1957-1984 = F. Halkin, Bibliotheca Hagiographica Graeca, 5 vols., Brussels, 1957-1984

Hamdi et al., 2021 = Ahmed Hamdi et al., A Multilingual Dataset for Named Entity Recognition, Entity Linking and Stance Detection in Historical Newspapers, in SIGIR'21: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, 2328-2334, https://doi.org/10.1145/3404835.3463255

Humbel et al., 2021 = Marco Humbel et al., Named-entity recognition for early modern textual documents: a review of capabilities and challenges with strategies for the future, Journal of Documentation 77.6 (2021), 1223-1247, https://doi.org/10.1108/JD-02-2021-0032

Hyvönen et al., 2019 = Eero Hyvönen et al., Linked Data – A Paradigm Shift for Publishing and Using Biography Collections on the Semantic Web, in Proceedings of the Third Conference on Biographical Data in a Digital World, BD 2019, 16-23, http://ceur-ws.org/Vol-3152/BD2019_paper_3.pdf

Joyce, 2021 = Kara E. Joyce, Graph database vs. relational database: Key differences, 2021, https://www.techtarget.com/searchdatamanagement/feature/Graph-database-vs-relational-database-Key-differences

Jurafsky and Martin, 2023 = Dan Jurafsky and James H. Martin, Speech and Language Processing, 3rd ed. draft, 2023, https://web.stanford.edu/~jurafsky/slp3/

Kahle et al., 2017 = Philip Kahle et al., Transkribus - A Service Platform for Transcription, Recognition and Retrieval of Historical Documents, in 14th IAPR International Conference on Document Analysis and Recognition, ICDAR, 2017, 19-24, https://doi.org/10.1109/ICDAR.2017.307

Kindt at al., 2022 = Bastien Kindt at al., Analyse automatique du grec ancien par réseau de neurones. Évaluation sur le corpus De Thessalonica Capta, Bulletin de l'Académie Belge pour l'étude des langues anciennes et orientales 10-11 (2022), 537-562, https://doi.org/10.14428/babelao.vol1011.2022.65073

Koutsikakis et al., 2020 = John Koutsikakis et al., GREEK-BERT: The Greeks visiting Sesame Street, in SETN 2020: 11th Hellenic Conference on Artificial Intelligence, 2020, 110-117, https://doi.org/10.1145/3411408.3411440

De Marneffe, Manning, Nivre, and Zeman, 2021 = Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman, Universal Dependencies, Computational Linguistics 47.2 (2021), 255-308, doi: https://doi.org/10.1162/coli_a_00402

Nadeau and Sekine, 2007 = David Nadeau and Satoshi Sekine, A survey of named entity recognition and classification, Lingvisticæ Investigationes 30.1 (2007), 3-26, https://doi.org/10.1075/li.30.1.03nad

Palmer, Gildea, and Kingsbury, 2005 = Martha Palmer, Dan Gildea, and Paul Kingsbury, The Proposition Bank: A Corpus Annotated with Semantic Roles, Computational Linguistics Journal, 31.1 (2005), 71-105, https://aclanthology.org/J05-1004.pdf

Pan, Vetere, Gomez-Perez, and Wu, 2017 = Jeff Z. Pan, Guido Vetere, Jose Manuel Gomez-Perez, Honghan Wu (eds.), Exploiting Linked Data and Knowledge Graphs in Large Organisations, 2017, https://doi.org/10.1007/978-3-319-45654-6

Pontes et al., 2020 = Elvys Pontes et al., Entity Linking for Historical Documents: Challenges and Solutions, in Emi Ishita et al. (eds.), Digital Libraries at Times of Massive Societal Transition: Proceedings of the 22nd International Conference on Asia-Pacific Digital Libraries, ICADL 2020, Kyoto, Springer, 2020, 215-231, https://doi.org/10.1007/978-3-030-64452-9_19

Ramshaw and Marcus, 1995 = Lance Ramshaw and Mitch Marcus, Text Chunking using Transformation-Based Learning, in Proceedings of the 3rd Annual Workshop on Very Large Corpora, 1995, https://aclanthology.org/W95-0107.pdf

Roux and Depauw, 2015 = Yanne Broux and Mark Depauw, Developing Onomastic Gazetteers and Prosopographies for the Ancient World Through Named Entity Recognition and Graph Visualization: Some Examples from Trismegistos People, in L. Aiello and D. McFarland (eds.), Social Informatics, 2014, Springer, 304-313, https://doi.org/10.1007/978-3-319-15168-7_38

Schreibman, Siemens, and Unsworth, 2004 = Susan Schreibman, Ray Siemens, and John Unsworth (eds.), A Companion to Digital Humanities, Oxford, 2004, https://companions.digitalhumanities.org/DH/

Simon et al., 2017 = R. Simon et al., Linked Data Annotation Without the Pointy Brackets: Introducing Recogito 2, Journal of Map & Geography Libraries 13.1 (2017), 111-132, https://doi.org/10.1080/15420353.2017.1307303

Sturgeon, 2021 = Donald Sturgeon, Chinese Text Project: A dynamic digital library of premodern Chinese, in Digital Scholarship in the Humanities 36 (2021), Supplement_1, 101-112, https://doi.org/10.1093/llc/fqz046

Vidal-Gorène et al., 2020 = Chahan Vidal-Gorène et al., Lemmatization and POS-tagging process by using joint learning approach. Experimental results on Classical Armenian, Old Georgian, and Syriac, in Proceedings of the 1st Workshop on Language Technologies for Historical and Ancient Languages, LT4HALA 2020, 22–27, https://aclanthology.org/2020.lt4hala-1.4.pdf

Yousef et al., 2023 = Tariq Yousef et al., Named Entity Annotation Projection Applied to Classical Languages, in Proceedings of the 7th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature, LaTeCH-CLfL, 2023, 175-182, https://aclanthology.org/2023.latechclfl-1.19

Zeldes and Schroeder, 2016 = Amir Zeldes and Caroline T. Schroeder, An NLP Pipeline for Coptic, in Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH), Berlin, 2016, 146-155, https://doi.org/10.18653/v1/W16-2119