References and citations in ancient greek documents

Samuel Gesche, Elod Egyed-Zsigmond, Sylvie Calabretto

Journal of Data Mining and Digital Humanities

References and citations in ancient greek documents

Samuel Gesche, Elöd egyed-Zsigmond, Sylvie Calabretto*

Université de Lyon
LIRIS UMR5205 CNRS / INSA Lyon
69621 Villeurbanne Cedex, France

***Corresponding author: elod.egyed-zsigmond@insa-lyon.fr**

ABSTRACT

This work focuses on refining and finding quotations of Greek Church Fathers within the Septuaginta and the Greek New Testament. Our corpus features over 700 works of various Church Fathers, which amounts to around ten million words.
In order to reach our aim, we had to explore the notion of quotation as relevant to the first centuries of our era, and to discuss the efficiency and usability of modern digital approaches, including their evaluation metrics. We mainly studied unsupervised approaches, either statistical, statistico-structural and statistico-semantic. We ended up building a generic and flexible solution based on a robust document model and an algorithm merging several other approaches. We are currently working on implementing this solution within the GATE framework to facilitate reusability.
In this article, we will talk about aspects of the challenges, the model we built to ensure genericity and flexibility, the algorithm we built from the various approaches we explored and discuss our results applying this algorithm to two different control sets.
keywords
ancient greek texts, citations, text modeling

1. OBJECTIVE

The purpose of our research is to find references of a work within another. Our focus is on ancient (Koiné Greek) documents written by the Greek Church Fathers that reference both the Tanakh and the New Testament (which will eventually be known together as the Bible). We do not delve deeply into the issue of versioning, although it is a really relevant problem in this research field, and one we had to clear beforehand (we still present it among the challenges we had to overcome).

Our research corpus is composed of the Septuagint (the Koiné translation of the Tanakh that was available at the time), the Greek New Testament, and around 700 works of various Church Fathers, which amounts to approximately ten million words.

Documents are structured and therefore can be referenced easily. Biblical texts are structured as books/chapters/verses, and patristic texts as work/chapter/paragraph/line (we discarded the page structure because it depends too much on which edition is used). This allows a single

word to be located unambiguously, because we do not have to take into account the differences between editions (which were, as we wrote, cleared beforehand).

Figure 1 and Figure 2 show an example of patristic text with its logical structure.

Quis dives salvetur, by Clement of Alexandria
...
chapter 42
paragraph 1
line 1: Ἵνα δὲ ἐπιθαρρήσῃς οὕτω μετανοήσας ἀληθῶς ὅτι σοὶ μένει
line 2: σωτηρίας ἐλπὶς ἀξιόχρεως ἄκουσον μῦθον οὐ μῦθον ἀλλὰ ὄντα
line 3: λόγον περὶ Ἰωάννου τοῦ ἀποστόλου παραδεδομένον καὶ μνήμῃ πεφυλαγμένον
paragraph 2
line 1: ἐπειδὴ γὰρ τοῦ τυράννου τελευτήσαντος ἀπὸ τῆς Πάτμου
...

Figure 1 Example of a patristic structured text (beginning of chapter 42 of 'Quis Dives Salvetur' by Clement of Alexandria)

```
<work title="Quis dives salveur" author="Clement of Alexandria">
…
<chapter num="42">
<paragraph num="1">
<line num="1">
<word num="1">Ἵνα</word> <word num="2">δὲ</word>
<word num="3">ἐπιθαρρήσῃς</word> <word num="4">οὕτω</word>
<word num="5">μετανοήσας</word> <word num="6">ἀληθῶς</word>
<word num="7">ὅτι</word> <word num="8">σοὶ</word> <word num="9">μένει</word>
</line>
<line num="2">
<word num="1">σωτηρίας</word> <word num="2">ἐλπὶς</word>
<word num="3">ἀξιόχρεως</word> <word num="4">ἄκουσον</word>
<word num="5">μῦθον</word> <word num="6">οὐ</word> <word num="7">μῦθον</word>
<word num="8">ἀλλὰ</word> <word num="9">ὄντα</word>
</line>
<line num="3">
<word num="1">λόγον</word> <word num="2">περὶ</word>
<word num="3">Ἰωάννου</word> <word num="4">τοῦ</word>
<word num="5">ἀποστόλου</word> <word num="6">παραδεδομένον</word>
<word num="7">καὶ</word> <word num="8">μνήμῃ</word>
<word num="9">πεφυλαγμένον</word>
</line>
</paragraph>
<paragraph num="2">
<line num="1">
<word num="1">ἐπειδὴ</word> <word num="2">γὰρ</word> <word num="3">τοῦ</word>
<word num="4">τυράννου</word> <word num="5">τελευτήσαντος</word>
<word num="6">ἀπὸ</word> <word num="7">τῆς</word> <word num="8">Πάτμου</word>
</line>
…
</work>
```

Figure 2 Alternative xml representation of the same text

2. CHALLENGES

Finding references is not a new problem, but like many problems that involve documents and language, the approach depends on the material. In this section, we will present the challenges presented by our corpus, and how we approach them. we will also present some works that already used Computer Science to perform reference retrieval.

2.1. Koiné Greek

Koiné Greek is a language that gained relevance following the conquests of Alexander the Great, went global during the hellenistic period and the Roman empire, was supplanted by Latin around 300 A.D., and developed further within the Byzantine empire, becoming the much different Medieval Greek in the process.

Koiné Greek was therefore the global language during the first centuries in the Roman Empire, which means that it was constantly evolving to meet new needs, new markets and new concepts, much like English nowadays. This means that *koiné* Greek is really different from both the former Classical Greek and, of course, the Modern Greek. This is true on the level of named concepts, and this is also true on the level of some basic language features (for instance, the dative case has disappeared in Modern Greek, as has the dual number).

The consequence is that semantic resources tailored to either classical or modern Greek cannot be used easily, if at all. Words have changed meaning, some have disappeared, while others appeared. This also impacts the use of language processing resources, such as lemmatizers: when tailored to classical Greek, they do not yield satisfactory results when used with the four centuries older *koiné* (for example, our tests showed less than ¼ overlap between the lemmatised forms available in Perseus, which is mainly classical, and the forms found in our *koiné* corpus).

There are, however, some things that remained constant:

- The alphabet did not evolve much: as a matter of fact, the Unicode standard for polytonic Greek works for both classical and *koiné* Greek (we will point out however that there are still other encoding standards, which still causes conversion issues);
- The language itself remains based on inflection, which means that the word order within a sentence bears no importance, at least concerning the meaning of the sentence (this means in particular that a quotation can reorder words); unfortunately, we have to add here that in most antique manuscripts, there is no visual indication of sentence beginning or end (and more generally, very little punctuation; as a matter of fact, even the spirits and accents were added later when trying to assert how to pronounce the sentences - we could say that the original text is not far from being normalised already). Depending on the digital source, we thus may or may not be able to split the text into sentences.

2.2. Quotation in the Antiquity

Quotation finding, and more generally finding any kind of text similarities between documents (including, of course, plagiary), is a research field with many applications, and many approaches (some of which we will present in section 2.4). However, most of those

apply to quotation, and text similarity, as it is defined today. We quickly realised that, in order to work on ancient documents, we had to define what a quotation was at the time.

Quoting other people -be it written text, vocal speech, gestures, or other means of communication- is intrinsically a part of the communication process. While the core concept stayed mostly the same (which is, recalling or telling what someone has expressed), the actual process of quotation, and the final result, has evolved because of several tremendous revolutions. Ancient texts come after the development of written language (with oral tradition still being very strong and as likely to be quoted as written text), but since then, the world has seen the development of the printing press, which made copy of written texts available for every scholar and every library, and the digital revolution, which made accessing texts, copying, quoting, transforming and broadcasting them trivial (to the point where the concept of document itself may not be clear anymore (Pédauque, 2007)).

In the Antiquity, texts (physically tablets, parchments, papyri or codexes) were not as readily available as nowadays. Copying was a long -and therefore expensive- process. However, during the first centuries of our Era, roads were mostly secure and travel was quick along the commercial axes, so travelling to read a specific book -and to copy some parts of it- was possible, as was transporting a specific text (apostles' epistles, for instance, initially were transported from community to community to be read, while the Great Library of Alexandria, as well as numerous others, were busy with scholar visitors).

Nonetheless, while today a literal quotation is a portion of text put between brackets that must be lexically *strictly identical* to the original text, at the time a literal quotation was merely *exactly corresponding* to the original text, as a result of the writer either having the original text before his eyes, or having a faithful memory of the text. In consequence, an ancient literal quotation says exactly what the original says, but not necessarily with the same words.

In this context, non-literal quotations are quotations that do not strictly correspond to the original text; either because of a partial memory, or by a deliberate change. And then, there are the references that are not quotation *per se*, or that are *implicit quotations* -or cross-references. While lexically they do not contain any form of similarity to the original text, they do reference this text. Often, an analysis at semantic level is not sufficient to detect such a reference: the proper level is the pragmatic level. This is for example where the use of that very characteristic formula leads to considering that this author has been influenced by some work of this other author -which can be a material for thesis rather than automated digital process. If we were to provide a single relevant number, it would be that 20% of the references we worked on do not have even a single lemma in common (yet lemmatisation drops this number from an initial 43%).

What remains clear in that context is that searching for lexical similarities, while helpful, will not be nearly as successful as in a modern context. And since semantic resources that may be available for Modern Greek are irrelevant for *Koiné*, to use any approach by this angle we had build them first. We cover this aspect in section 4.2.

2.3. Versioning

Taking two documents and a control set of references and tuning our reference search engine to find the best compromise between precision and recall, besides the difficulties mentioned previously, also brings into light another challenge, and it is that of versioning.

The first aspect of versioning is what we could call present versioning. We work on documents that have been written so long ago that they are essentially lost, destroyed by time, arsons and carelessness (among other factors). This means that we only have copies (of copies) of the original documents, and many copies can co-exist, with different proximities to the original text. When we take our list of references, the following question is raised: to which copy do they refer? For instance, we work on the Septuagint; however, our references were initially using the Jerusalem Bible, a modern version. As we were quickly informed, there are much more than a (mostly harmless in our case) reordering of the books between the two. Some sections have been moved around, which alters the numbering and thus the references. Some sections have been intertwined, two texts being reordered and mixed together. This means that we had to basically transform the references as between two vector spaces (or at least, ensure that the transformation had been done when receiving a reference list).

The second aspect has more to do with the ability to find the references in the texts; this past versioning deals with the issue of finding the actual texts that were used by the authors to write their discourse. We use the Septuagint because it is contemporary to our authors, but some references are made to other versions that were available, and may be lost to us. If the text differs too much, we may be unable to find the references (even though they may have been documented by scholars over the centuries). Of course, we cannot do much with this aspect of versioning; it is merely a hard limit to our ability to find some references.

2.4. Related Work

2.4.1. Related fields

Quotation detection is a research field that grew increasingly important during the last years, benefiting from the more large-scale interest for text reuse and plagiary detection that is crucial to fields such as the patent business or copyright enforcing.

One great asset of this field is that it can be considered as a specific subfield of information retrieval, using any part of a whole document as a query. Of course, using such large-scale querying means that specific measures must be taken to ensure that the answers can be found in a reasonable amount of time; however, both the approaches and the metrics (notably precision, recall and f-measure) can be reused to a certain extent, as long as the core of the problem remains similarity detection (a process for which (Lukashenko *et Al.*, 2007) or (Bao *et Al.*, 2006) recall most approaches and paradigms).

However, this means that plagiary and quotation detection also suffers from the same difficulty as information retrieval, which is the translation of the query -here some part of a text- into a form that will allow finding relevant documents -those that quoted that part of text- in a large corpus. Therefore, if the detection of literal quotations is as easy as matching words, in most cases more advanced strategies are required. There is a gradation between

literal, erroneous literal, literary or paraphrased quotations and allusions (references that sometimes require a large context awareness) that is not unlike the gradation between lexical, syntactic, semantic and pragmatic levels of analysis. Plagiary detection has shown advances, for example in paraphrase detection ((Faisal *et Al.*, 2012), (Jo *et Al.*, 2007)), but even then relies more on knowledge of the actor's mindset -the psychological mechanisms of plagiarism and paraphrasing- than on crude text analysis heuristics. After all, plagiarism is not taught at school like quotation for everyone to abide by the rules, but rather is discovered by individuals and remains competitive -the plagiarist versus the plagiarism detection. There is, however, the case where text reformulation merely serves the purpose of merging the quotation within a new discourse, and the work is far easier then. (Ernst-Gerlach *et Al.*, 2008) work on this kind of reformulation, including errors, word addition, deletion or change, and language evolution. Lastly, semantic approaches emerge as semantic resources are made available. For instance, (Nawab *et Al.*, 2012) uses synonymy to enhance n-grams overlap detection.

Automatically detecting allusions is still very difficult unless specific semantic or pragmatic resources are tailored to the task. Meme tracking on the Internet may be the beginning of an answer, but it focuses on quick mapping of the spread of a widely-used, quickly mutating formula and not on detecting less widely used formulae over a large time frame (Leskovec *et Al.*, 2009). Still, the similarity between both cases does exist; though the 140 characters long document standard on Twitter makes quotation detection far easier, since the quotation, if any, is likely to be the whole document -a document that is even timestamped to ensure that the order of publication is known.

In addition to being related to the field of information retrieval and plagiary detection, the field of text reuse detection is also a well established field of research in the humanities. Using a computer to enhance the process of critical edition, in particular, has given birth to several projects using the hypertext paradigm to annotate intertextuality. In this case, automatically detecting text reuse is not the focus; it is the work of experts of whatever work is studied. Exhaustivity (recall) is therefore often abandoned in favor of correctness (precision), and these systems reflect that, giving the paternity of the link to the expert that found it, and allowing discussions to take place to confirm, refine or refute it.

2.4.2. Quotation detection in antique texts

(Ernst-Gerlach *et Al.*, 2008) proposes an approach for discovering references in a Latin text corpus. This work provides a typology for the text differences inherent to the mechanics of referencing (regular and irregular differences, deletions, insertions and substitutions). The proposed method assumes the availability of simple resources such as proper nouns (characters, authors, ethnic groups, places *etc.*) and number format variations (digits, letters, abbreviations). Their algorithm is based on the concept of sliding overlapping windows and is very tolerant, two identical words within a window being sufficient to assert a quotation. The method is tested on quotations taken from a Latin dictionary (basically using the examples of an entry as the quotations to find), and these quotations are looked for in the Perseus corpus. The results have a good recall but a varying precision. Among the reasons for the lack in precision is the impact of short words or that of not taking word order into account. They suggest taking into account the amount and frequency of terms, their orders and stop-words.

The expected results are also far worse on short quotations (especially one word quotations, which are more than half their corpus).

(Lee J., 2007) takes another approach to antique text (although that focus is accidental) and provides many elements on the specificities of these texts and their languages. Applying cosine similarity to verses of the Greek New Testament, and consolidating highly similar verse groups, this work is able to find many quotations within the three synoptic Gospels. One of the main hypotheses is that quotations follow the same text order in both the quoted text and the quoter. The approach is evaluated against nine quotation tables obtained from nine different experts, and the block approach is thus validated empirically.

(Büchler M. *et Al.*, 2010) studies text reuse for the purpose of discussing the available versions of the texts. This work explains the difference in the practice of quotation between Antiquity and nowadays (and in particular the absence of the explicit reference). The work also adds a visual analysis over the text processing aspect, which is enhanced itself using the Perseus lemmatizer. The described algorithm is based on n-grams as well as prefix filtering and aims at finding similar areas in the texts, then performs a more semantic analysis using significant words co-occurrence in sentences. The approach is able to take into account language evolution and dialect change, as well as word omission and substitution. However, it cannot discriminate between quotations and large idiomatic expressions, and is not good at finding the exact quotation boundaries.

2.5. Available resources

Testing the content of two documents against each other to find similar fragments is the core of quotation finding, but it is often not enough, especially if the used method is to compare the raw text of the documents. To refine the material and ensure better results, this text, and words within, are usually refined along two dimensions: a, so to say, vertical one, and a rather horizontal one. Of course, every approach includes both to various degrees.

Vertical refining aims at trying to reach the ideal case where what is compared is not mere text, but the actual sense of the discourse. Several levels have been characterised, mainly:

- the level of signal (using the theory of information);
- the level of medium, language and format (which translates into import routines, from simple file access to much more complex scan/recognition/translation chains);
- the morphosyntactic level (using the words and sentence structure);
- the semantic level (using the meaning of the words and the relations between them within the language);
- the pragmatic level (computing the sense of the global discourse, taking intertextuality into account)

Horizontal refining, comparatively, aims at making the most out of each level. For instance, the morphosyntactic level can benefit from lemmatisation or stemming; similarly, the semantic level can use tables for all kind of relations from synonymy to metonymy, up to a full-fledged ontology.

While vertical refining is theoretically the most powerful tool -going from a level to the next grants a great amount of relevance to the results- it is horizontal refining that is, pragmatically, what distinguishes an approach from another. Which means that the building, and sharing, of resources that achieve better horizontal refining have *de facto* more importance than simply advertizing a method as being at some vertical level. The problem is, however, that such resources are intrinsically dependant of the features of the discourses, and the higher on the vertical ladder, the worse it gets. This makes simply building an academical library of resources a very complex task. For instance, Wordnet is a great tool; unless your text is not in English, or is really old, or is full of highly specialised jargon that happens to hold most of the sense.

2.5.1. Lemma and morpho-syntactic analysis

The available resources were mainly lemma and morphological analysis for a number of word forms in ancient Greek. We used three main sources: Perseus, initially through the Archimedes Project Morphology Service, then through their database dump available for download; BibleWorks, a Bible-focused software who provided an export for morphologically analysed texts; finally, the lemma tables of Sources Chrétiennes, built over the last years. In total, we got over 380 000 lemmatised ancient Greek terms, discounting homonymy and polysemy.

Without a morphological analyzer for our own texts, and without a morphological analysis accompanying the lemma tables of Sources Chrétiennes, we decided to limit our language processing to matching a term with a lemma from these lists. Of course, we were then vulnerable to the issues of homonymy and polysemy (Coulie, 1996). We thus took a priority rule as follow: first, if the term is already a lemmatised form (that is, if one of the lists has the term identical to one of its possible lemmas), the term itself is kept. If it is not the case, we take the first lemma present in these tables for the term, prioritizing the tables according to their focus to patristics:

1. Sources Chrétiennes first, with 21 505 lemmatised terms in koiné Greek;
2. then BibleWorks, with 115 498 lemmatised terms in koiné Greek;
3. finally Perseus and its 318 584 lemmatised terms spread over multiple forms of ancient Greek language, including koiné;

All in all, we were able to find a lemma for 40% of our entire text corpus (however, several texts were entirely lemmatised, and we experimented on them), and 24% of the available lemmatised terms were present in the text corpus (78% if we discount Perseus due to its language spread). Figure 3 presents the distribution of the lemmatised terms within the different sources, and the contribution of these sources to the lemmatisation of the terms present in the corpus.

As a side effect, we got the list of around 5 000 proper nouns used in the Septuagint.

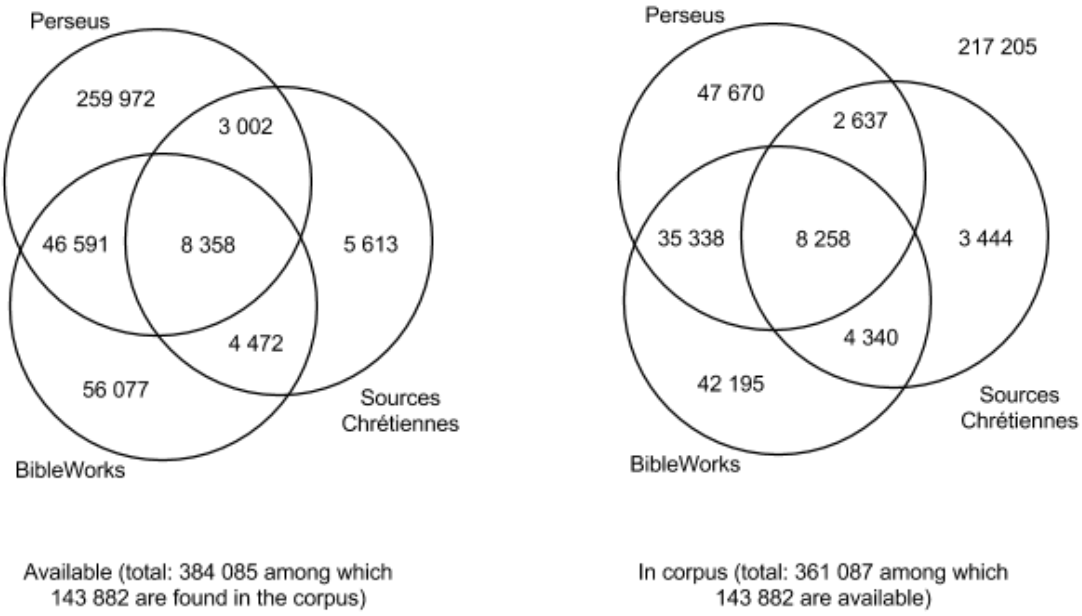


Figure 3 Distribution of the available and necessary lemmatised terms among the three sources Perseus, BibleWorks and Sources Chrétiennes (left: terms available for lemmatisation; right: terms both available and actually present in our corpus -the outer number represents the amount of terms for which none of the sources provide a single lemma)

Automated lemmatisation like (Dimitrios *et Al.*, 2008) may greatly enhance these results but we have not delved into it currently, since it is not the focus of our research. The same is true for stemming, which we could not achieve with simple approaches (Greek is a not an easy language to stem), and for which we could not find lists of already stemmed terms.

2.5.2. Stopwords and other recurring terms

We have a very narrow list of stopwords, the experts having determined that a larger list did not reflect the list of basically disposable terms. We have 18 lemma in this list (οὖν, τε, ὁ, καί, δέ, γάρ, τίς, τις, ἤ, ἠ, μέν, μήν, δή, γε, ἄρα, ἄν, αὐτός, πότε).

Using a table of the most frequent terms, we built two more lists, a list of recurring terms in the language (111 lemma such as ὅσιος - hallowed), and a list of recurring terms in the corpus (870 lemma such as κύριος - lord). These lists allow us to better adjust the sensibility of our algorithm, and better, to use each list at the relevant steps (for instance, a stopword can be eliminated from the start, but recurring words will typically be used to discard results that would only contain them).

On top of that, we listed recurring expressions as well (24 in the language and 202 in the corpus, still growing).

We called those recurring formulae, terms or expressions nonquotes. While they do contribute to the meaning of the text, and as such cannot be simply discarded, their presence cannot be used alone as an argument to support a quotation.

2.5.3. Semantic resources

Semantic approaches are greatly limited as long as they rely on resources that are highly language-dependent. However, statistico-semantical approaches remain usable as long as their underlying hypotheses on language are valid for koiné Greek. (Büchler *et Al.*, 2010) actually uses the statistico-semantical tool of significant terms co-occurrence, and (Ernst-Gerlach *et Al.*, 2008) uses names as hints for a greater chance of quotation. Other semantic analysis methods such as LSA may yield results too.

3. MODEL

With that many challenges, having in particular so many differences between the definitions in Computer Science and Humanities -reference retrieval is best led by a transdisciplinary approach- we could not rush through these issues of definitions to begin implementation immediately. Before building algorithms, heuristics and optimizing a process, we took the time to define the concepts relating to our research field. Four concepts came almost immediately on the table: we had *documents*, these documents were linked by *quotations*, these quotations were carried by *references*, and finally all of this would require working on the level of *terms*. We had to sort out many ambiguities in the language we were using, and it is finally a whole class model that we had to define. As a result, development was made far easier, and we ended up with several project branches.

To devise such a model, we used part of the Praxeme UML-based paradigm. In particular, insofar as this paradigm separates semantic concerns -what we work on- from pragmatic ones -how we work on them. Some UML design patterns were also borrowed from them.

In the Praxeme paradigm, a semantic model is seen as a description of the knowledge of the domain, much like an ontology. It does not include the know-how, but merely the specification of the objects that interact within the modeled system. Depiction of these objects mainly use two of the UML diagrams available, class and state diagrams, and must make the most use of the primitives of modeling to get as generic as possible; at least, making a clear distinction between essences (classes) and roles, the textbook case being the distinction between a person (essence) and a client (role).

Unsurprisingly, we obtained a class system centered around the four notions that we had found: document (or rather passage), quotation, reference (as a coordinate), and term.

3.1. Language ambiguities and definitions

Our field of work -detecting quotations among ancient documents- is intrinsically multidisciplinary, and therefore there are some concepts that are defined differently from different viewpoints. We had to find those, and sort them out, or we were likely to confuse them down the road. Among these concepts (without claiming that it is the best terminology):

- A *term* is a string of characters, and is defined only by the characters it is composed of. For example, ‘bird’, as composed of the four letters *b i r d*, is a term. A term cannot contain spaces or hyphens (unless spanning over multiple lines for the latter). *Words* are, on the contrary, a located unit of discourse composed of a single term. While all

occurrences of the term ‘bird’ are treated as the same entity, all occurrences of the word ‘bird’ are separate entities, because they occur at different places in the texts.

- A *reference* is a piece of code giving the position of a fragment of text within the whole text. We can approximate it as textual coordinates. But *reference* is also a concept that encompass quotations and allusions, as we defined them in section 2.2. We could not decide on two different terms, mainly because both concepts actually *refer to* something (and the same term is actually used for both concepts in the domain). Fortunately, this means that context is enough to know which one the discourse is about.
- A *passage* is simply a fragment of text. It is not necessarily connected, and as such it can span over multiple intervals. A *document*, in contrast, is the result of a work of construction by a single or several authors. *Text* is the term which refers to the loose concept of *anything written*.
- A *n-gram* refers to a succession of n words (not terms) from a text. They follow the text order.

3.2. Text Structure as a Graph

The core of our work lies in working with documents. Since we often manipulate fragments of them, and have to be able to locate the parts within the whole, the obvious choice was to model a document as a text graph. We quickly discarded the document itself as the essence of our main object though, and fell back on the concept of passage. The passage is therefore the node of our text graph, documents potentially being roots and words being the leaves.

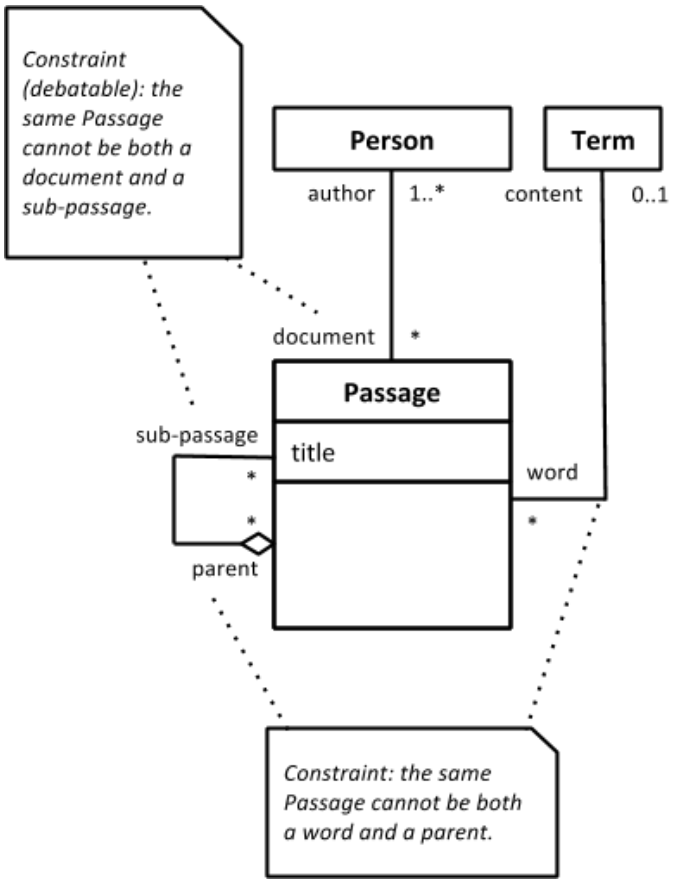


Figure 4. Class representation of the concept of Passage

As implied by Figure 4, text graph composed of passages is oriented and acyclic, and thus describes hierarchies of passages that combine from words. The document is a root of the graph (because we chose to define the document as the the top-level of composition, although other definitions are possible), but quoted passages also are potential roots. However, what remains is that a *document* is a passage that is linked to a person (the author) while a *word* is a passage that is linked to a term (the content). A document cannot be a sub-passage (by our definition) while a word cannot be composed of other passages.

To depict the inherent order of passages within a parent passage, we used the concept of coordinates; coordinates depend on strict orders which contain the semantics of the sequences (many sequences are numerical, but we also had to work with non-numerical sequences such as the book orders in different versions of the Bible).

The class diagram of the text graph is presented in

Figure 5, and an object diagram using part of the example presented in Figure 1 can be found in Figure 6.

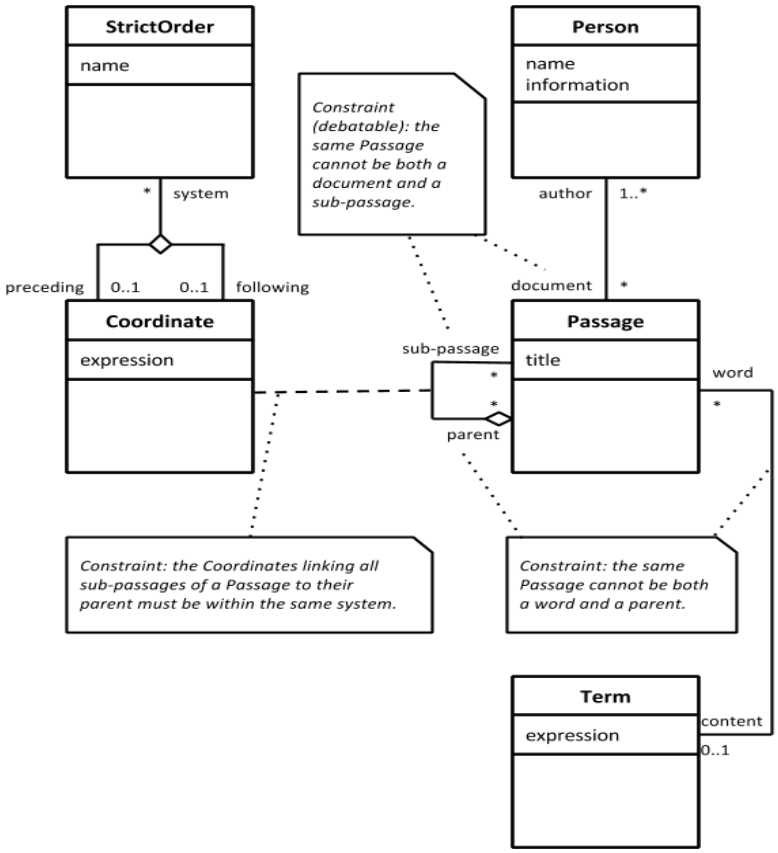


Figure 5. Class diagram of the graph structure for documents

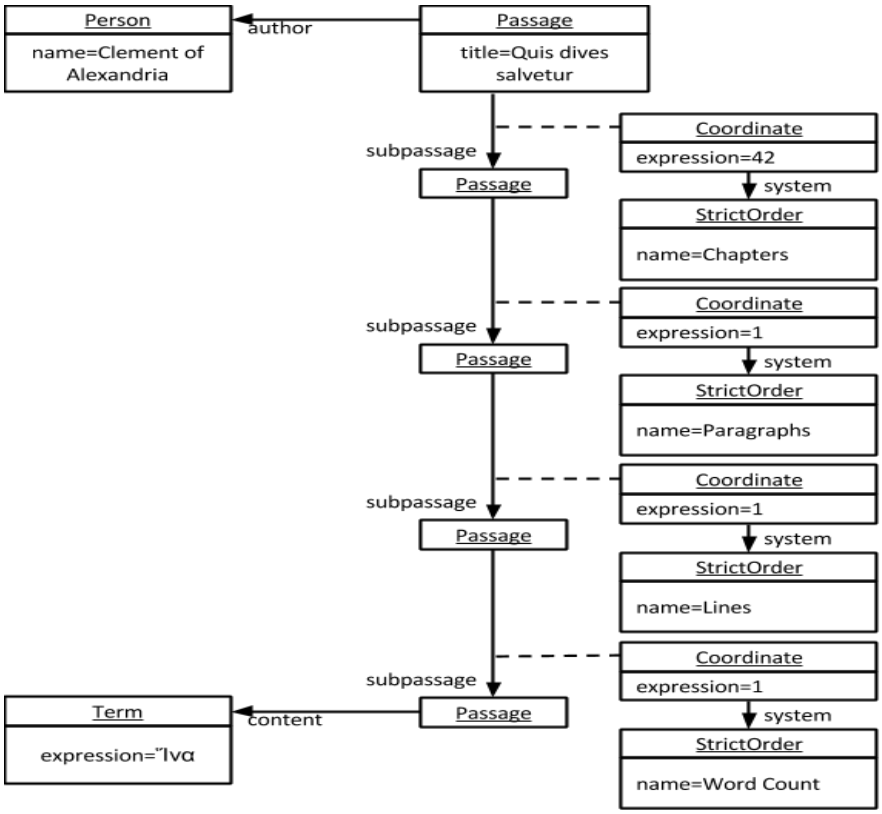


Figure 6. Object diagram of chapter 42, paragraph 1, line 1, first word of "Quis dives salvetur"

Passages contain all structural elements (from upper-level elements like work or chapter to leaf-level elements like words), Terms contain the terms that are contained within those structural elements, and Coordinates and StrictOrders carry the logical description of the structure (for example numerical coordinates for ordered lists such as chapters and lines).

3.3. Reference as a Query

Thinking of the texts as graphs has the immediate additional advantage to allow us to consider references -pieces of code purposely built to unambiguously describe a fragment of text- as queries on these graphs. We therefore see the reference object as a subgraph describer, using it to isolate a subgraph within a whole graph. In UML, this translates as a ternary relation between a Reference, a Passage that is a fragment, and a Passage that is a whole. For a query to be relevant, the *whole* must of course contain the *fragment*. Figure 7 shows the relationship between the concepts of Passage and Reference.

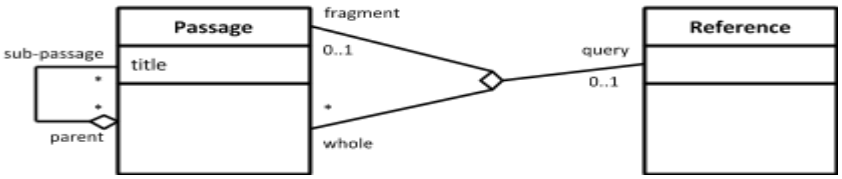


Figure 7. Class representation of the relationship between a Reference and the referenced Passages

Designing the concept of reference is more difficult than simply wrapping a beginning and an end into an object. Most of the references were fortunately simple intervals, such as ‘chapter 1 paragraph 3 lines 4 to 7’. However, we also have numerous cases where the reference spanned over multiple intervals (*e.g.* ‘lines 4 to 7 and 12 to 18’). Moreover, we found several instances of a case where a reference was described by exclusion (e.g. ‘line 4 words 1 to 10 but 4 and 7’) when a single text referenced two different mixed texts.

Building a model able to integrate all these schemas into one structure was facilitated, however, by the similarity between this problem and the case of a period of time, for which Praxeme had already provided a robust model. We simply adapted it.

Basically, a reference is either a point or an interval, and can be related to other references by three relations: inclusion (the related reference is added to this one, *e.g.* line 3 and line 6), exclusion (it is taken out, *e.g.* lines 4 to 8 but 6) or precision (the related reference describes what is kept of this one at a lower level of granularity, *e.g.* line 4 but only words 2 to 4).

The corresponding class diagram can be found in Figure 8.

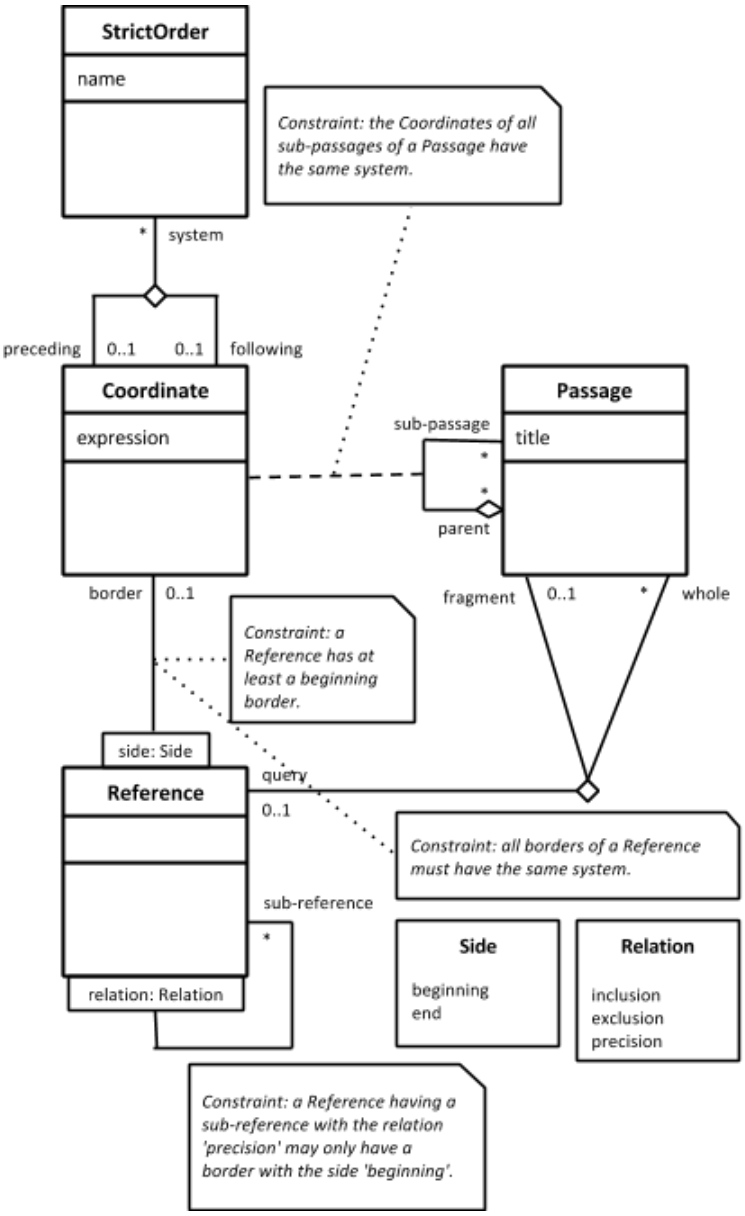


Figure 8. Class diagram of the query structure for references, reusing of the Coordinate and StrictOrder model from section 3.1.

So what will happen when we parse, for example, the reference ‘Quis Dives.17.1.4.8-5.2’? The Reference object will be built as shown in Figure 9.

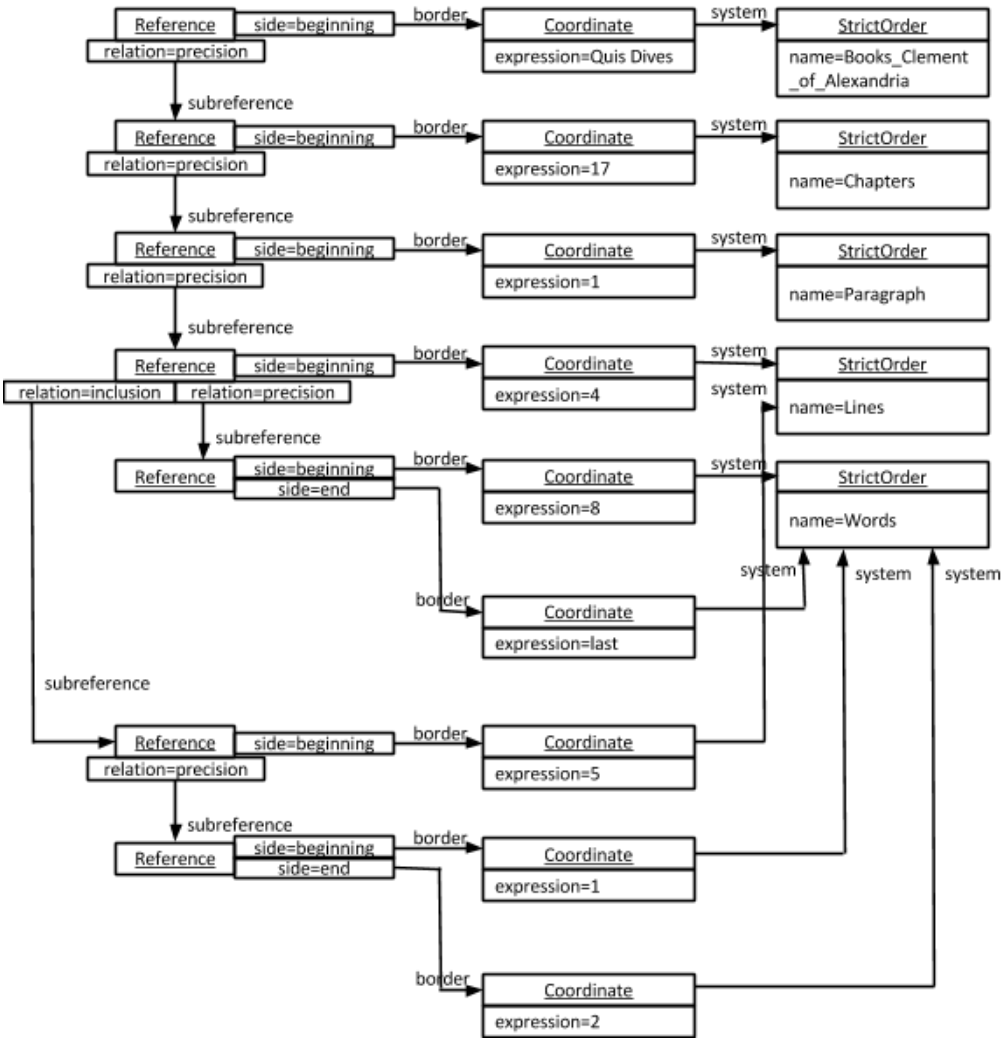


Figure 9. Object diagram of the reference 'Quis Dives.17.1.4.8-5.2'

We had to include the concept of 'last' as a possible value for every strict order. The reason is that for most of the references, we cannot know what the last value is before matching it to a passage. It depends on both the specific passage and the whole reference chain (which is, as we designed it, simply linked from top granularity level to bottom: a given sub-reference does not know its super-references).

The matching process will then proceed to successively find the nodes in the passage that match each of the sub-references. If such nodes exist, the sub-graph will be returned as a passage.

3.4. Text Content as Processable Language

While we used the terminology of 'word' to represent the leaf of the document graph structure, we used 'term' to refer to the actual linguistic item composed of characters. This means that, in our model, the term is the content of the word.

A term can be subjected to language processing, like normalisation, lemmatisation, stemming, or named entity tagging. We did not use language processing at the level of sentences (although we have morpho-syntactical analysis for a significant part of the terms), mostly because in several cases delimiting the sentence had to be done first (antique greek did not use sentence delimiters and this means that not all our texts have them).

Processing finds its way into our model as shown in Figure 10, with an example object diagram presented in Figure 11.

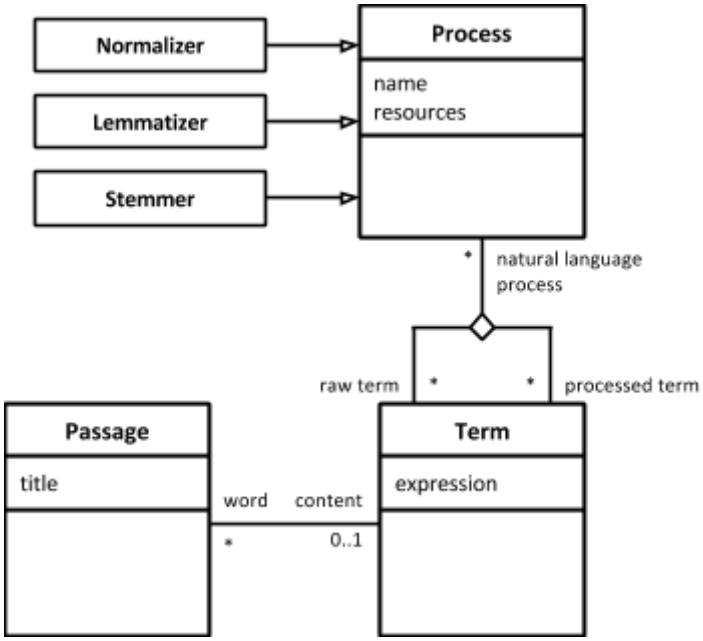


Figure 10. Class diagram showing the interaction between Terms and language processing.

The main objective of processing terms is merging terms that would refer to the same thing in context (for example child(ren)/progeny/heir(s)/descendant(s)). There are mainly two sets of tools, working either at the lexico-syntactic level (normalisation, lemmatisation and stemming) or at the semantic level (using synonymy, antonymy, metonymy, meronymy, or even lexical field). Using the resources we could obtain (see section 2.4.4) and an algorithm for statistically inferring semantic relations between terms in the corpus, we could choose a level of processing for our analysis.

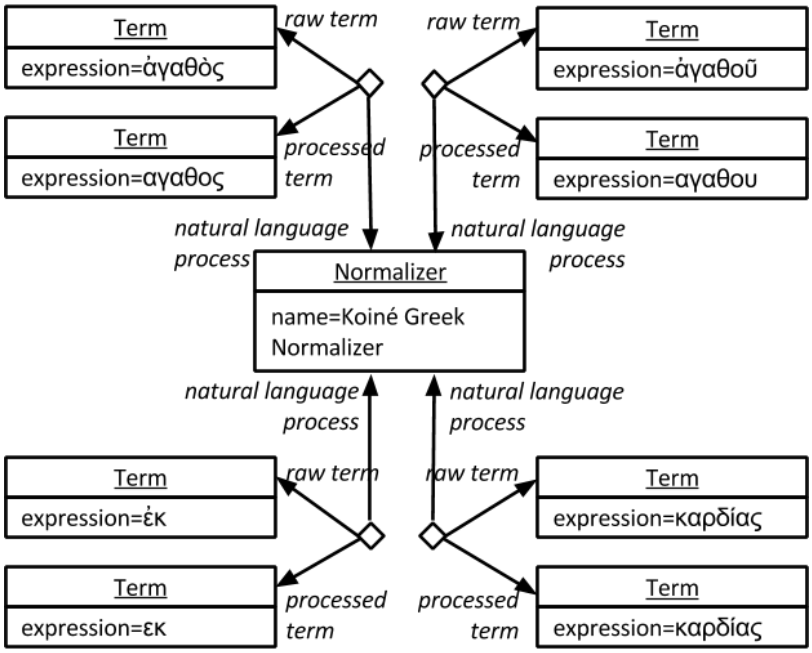


Figure 11. Example object diagram showing the processing of some terms.

3.5. Quotation as a Graph Match

Having used the structure of the documents to build document graphs, and using a coordinate system to be able to label a part of these graphs was aimed towards using graph matching to find quotations between the documents. A quotation is thus seen as the association between two Passages, these Passages being subgraphs of the document graphs. Figure 12 presents the relevant class diagram.

These subgraphs can be described under the same coordinate system as their respective documents, which allowed us to easily link quotations that we have found and references that were already expressed as coordinates in a textual manner. This in turn allowed us to compare both sets, which gave us a control set against which evaluate our quotation finding processes.

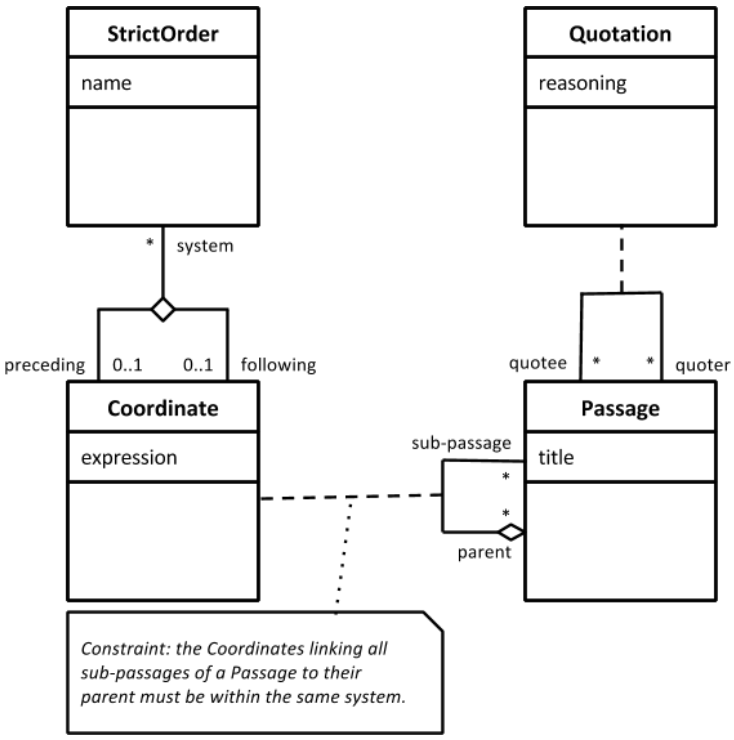


Figure 12. Class diagram presenting the Quotation as an association between two Passages.

The quotation is a dynamic object, in that there its status relatively to the state-of-the-art set of known quotations can vary over time. We defined four main states that a quotation can have:

- Quotations from the fund that Sources Chrétiennes provided us are *refinable*. This means that they are, as a list that has been maintained for a long time, not accurate enough any more to satisfy the current standards.
- Quotations that are detected by our automated quotation detection are *suggested*. Precision is not 100% in our search, and some quotations may very well not be accurate -or even not quotations to begin with. It is the responsibility of an expert to validate these quotations.
- Once past the validation process, both refinable and suggested quotations become either *valid* or *invalid*. This sanction is given by an expert that did verify whether the bounds of the quotation are accurate enough -and whether it is a quotation at all.

Accuracy depends on the granularity that is demanded by the experts. In the case of our study, the line or verse was a sufficiently deep level. Other cases may want an accuracy to the level of the word. Other cases may only want to know which document quotes which other.

Figure 13 presents the states of a Quotation object, as well as the allowed state changes.

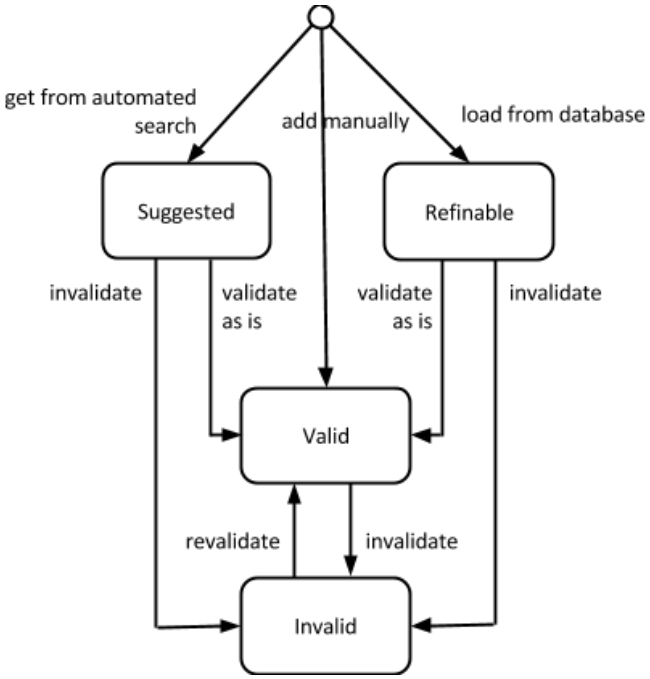


Figure 13. Statechart diagram of the Quotation object

4. ALGORITHMS AND HEURISTICS

Based on the model we just presented, we built a processing strata to perform the actual work of retrieving references (actually the references that we can find are best described as quotations). In this section, we will present our algorithm and the optimisations we used to keep it running in an acceptable time frame. While the whole algorithm uses the documents and references formatted in a manner compliant with the model presented in Section 3., we will not use the UML formalism to describe the algorithm. UML is far too verbose in that context. However, the text presented in Figure 4.1. is obviously internally represented as shown in, for instance, Figure 3.3.

However, terminology as detailed in section 3.1. will still be used here, in particular in regards to the difference between words and terms.

4.1. General quotation retrieval algorithm

The general algorithm can be broken down as usual into pre-processing (reading the data into instances of our model), main processing (computing the quotations) and post-processing (presenting the results).

To illustrate the different steps, we will use a fairly easy to find quotation between Luke 6.45 (in the Greek New Testament) and chapter 17 paragraph 2 line 2 of the patristic text ‘Quis Dives Salvetur’ by Clement of Alexandria. The texts and the referenced quotation are presented in Figure 14.

book of Luke
chapter 6
verse 44: ἕκαστον γὰρ δένδρον ἐκ τοῦ ἰδίου καρποῦ γινώσκεται οὐ γὰρ ἐξ ἀκανθῶν συλλέγουσιν σῦκα οὐδὲ ἐκ βάτου σταφυλὴν τρυγῶσιν
verse 45: ὁ ἀγαθὸς ἄνθρωπος ἐκ τοῦ ἀγαθοῦ θησαυροῦ τῆς καρδίας προφέρει τὸ ἀγαθόν καὶ ὁ πονηρὸς ἐκ τοῦ πονηροῦ προφέρει τὸ πονηρόν ἐκ γὰρ περισσεύματος καρδίας λαλεῖ τὸ στόμα αὐτοῦ
verse 46: Τί δέ με καλεῖτε Κύριε κύριε καὶ οὐ ποιεῖτε ἃ λέγω
(Greek New Testament)

chapter 17
paragraph 2
line 1: Θησαυροὺς δέ γε ὁ κύριος οἶδε διττούς τὸν μὲν ἀγαθόν ὁ
line 2: γὰρ ἀγαθὸς ἄνθρωπος ἐκ τοῦ ἀγαθοῦ θησαυροῦ τῆς καρδίας προφέρει
line 3: τὸ ἀγαθόν τὸν δὲ πονηρόν ὁ γὰρ κακὸς ἐκ τοῦ κακοῦ
(Quis Dives Salvetur)

Figure 14 - Example of a quotation (underlined) with the immediate textual context.

4.1.1. Pre-processing

Pre-processing takes care of getting the resources into a form that can be directly processed. It involves six different steps:

- Discarding the stop-words if such a choice has been made (the result on the example text of Figure 14 is shown in Figure 15).

book of Luke
chapter 6
verse 44: ἕκαστον δένδρον ἐκ ἰδίου καρποῦ γινώσκεται οὐ ἐξ ἀκανθῶν συλλέγουσιν σῦκα οὐδὲ ἐκ βάτου σταφυλὴν τρυγῶσιν
verse 45: ἀγαθὸς ἄνθρωπος ἐκ ἀγαθοῦ θησαυροῦ καρδίας προφέρει ἀγαθόν πονηρὸς ἐκ πονηροῦ προφέρει πονηρόν ἐκ περισσεύματος καρδίας λαλεῖ στόμα
verse 46: με καλεῖτε Κύριε κύριε οὐ ποιεῖτε λέγω

chapter 17
paragraph 2
line 1: Θησαυροὺς γε κύριος οἶδε διττούς ἀγαθόν
line 2: ἀγαθὸς ἄνθρωπος ἐκ ἀγαθοῦ θησαυροῦ καρδίας προφέρει
line 3: ἀγαθόν πονηρόν κακὸς ἐκ κακοῦ

Figure 15. The previous text and quotation without the stop-words.

- Reading the texts into graphs - detecting the document formats (among those that can be parsed by the available software), reading the texts and their structures and creating the graphs according to the structures (see Figure 16 for an exemple).

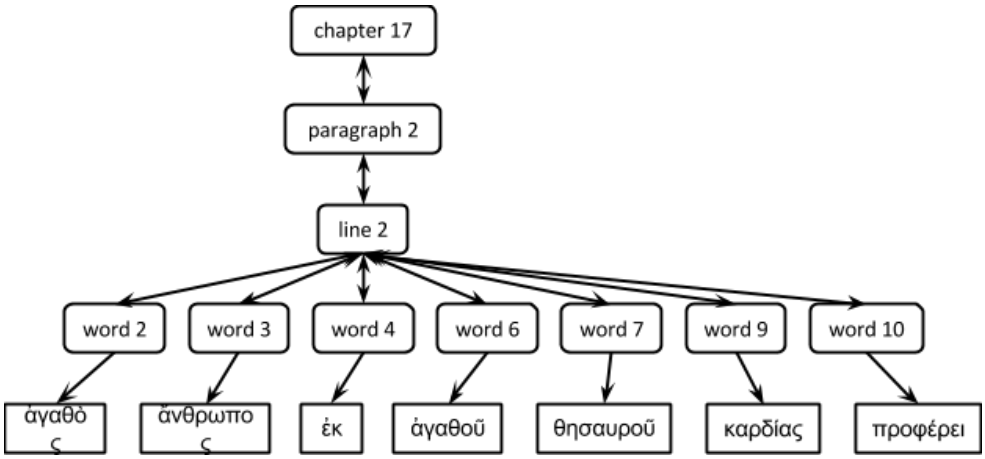


Figure 16. Part of a document structure graph according to the model presented in section 3.2.1 (arrows represent references, rounded rectangles are passages and straight rectangles are terms; stopwords are still counted but they are not part of the structure).

- Language processing - if a specific level of processing has been defined -raw text, normalisation, lemmatisation- the read text is processed and the text filling the leaves of the graph is replaced by the processed text (see Figure 17).

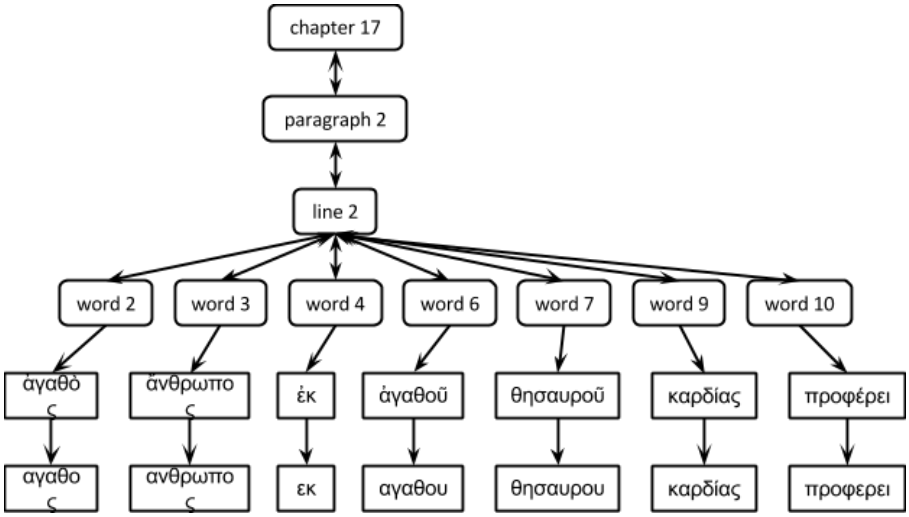


Figure 17. Document structure graph with processed text (here the processing is normalisation).

- Tokenizing - the algorithm is based on n-grams, with a configurable *n*; the sequence of n-grams is produced for both texts (Figure 18 continues the example).

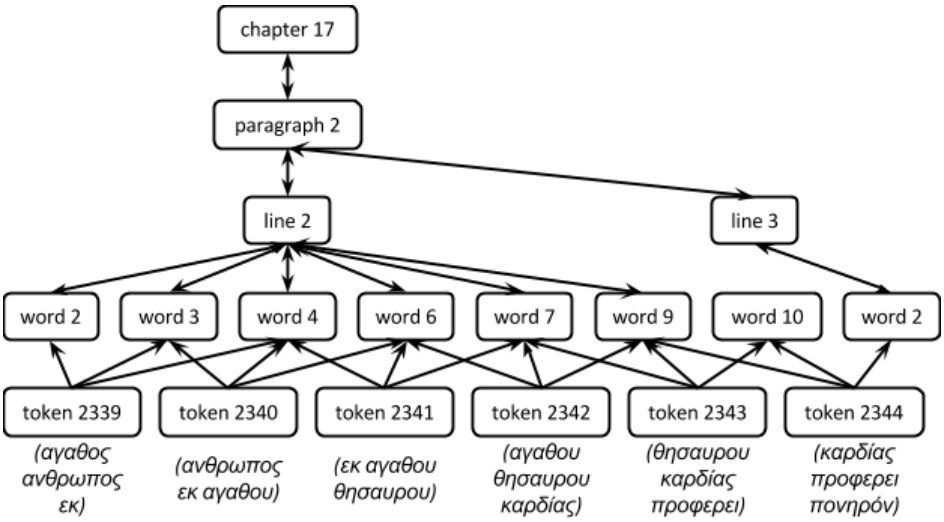


Figure 18. Result of tokenisation, with both the structure graph and token using the same words as leaves (in our example tokens are 3-grams).

- Indexing - an index is created linking processed terms with all tokens of the quoted document containing at least a word that contain the corresponding processed term (see Figure 19).

Figure 19. Representation of part of the generated index: each term is linked to all tokens (here, 3-grams) that contain it.

- Preparing the similarity measure - the similarity between terms is computed parsing the text corpus, according to the chosen parameters.

4.1.2. Main processing

The main processing is a three-steps process, the first step being to dredge all possible candidates from the pairs of tokens from both documents, while the second involves filtering out those of the candidates that are too different. Finally, the third step is the computation of the bounds of the quotations.

Finding all the possible candidates is done by matching all the token of the first document (the quotee) with the tokens of the second one (the quoter). To avoid a quadratic complexity, we use the indexes that have been computed in the pre-processing phase. What this means is that we browse the quotee tokens. For each of them, we look in the index for all the tokens of the quoter that contain at least one of its terms. This gives us a first set of candidates (which are pairs of tokens, one from the quotee, one from the quoter). Figure 20 shows the different steps of the calculation.

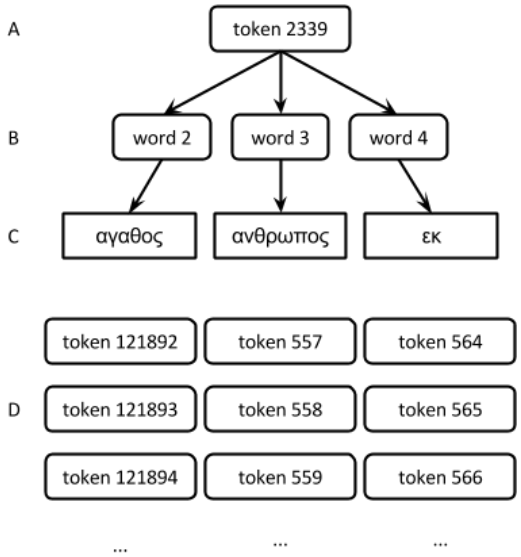


Figure 20. Schema of computation of the first set of quotation candidates, featuring A) the tokens from the quoter document, B) the words, as part of the document structure, of the quoter document, C) the processed terms contained within these words, and D) the set of tokens from the quoted document that contain each processed term.

Then, for each of these candidates, we compute the distance between the tokens. This distance is the amount of different terms between them (the actual definition of difference is dictated by several parameters, see section 4.4). If the difference is greater than an allowed threshold, the candidate is discarded (see Figure 21 for an illustration). The non-quotes can then be pruned out of the remaining candidates.

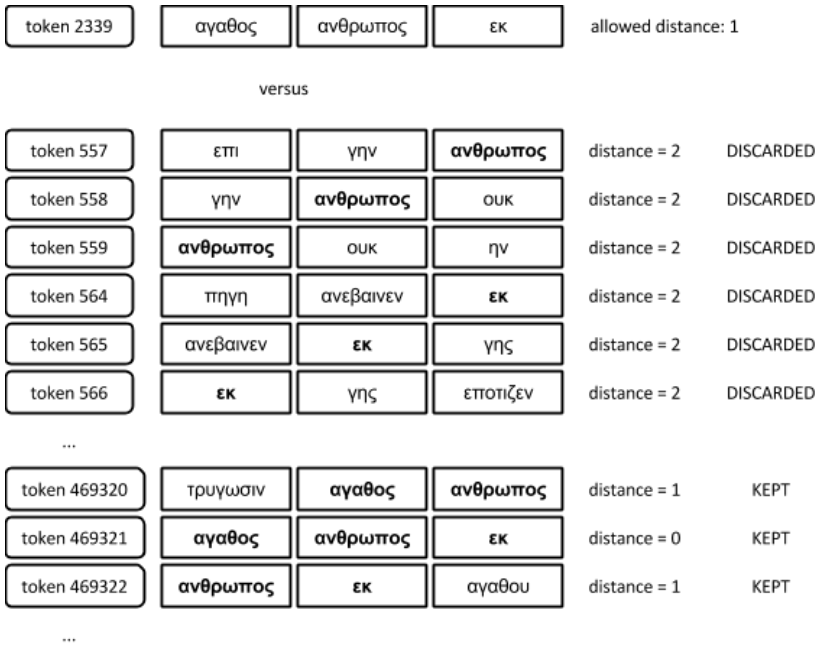


Figure 21. Distance computation (discounting word order, comparing normalised terms, allowing 1 different word between the quoter text and the quoted text; in bold are the common normalised terms).

Once the candidates set has been defined, each one of them undergoes the process of computing its actual bounds, as shown in Figure 22. Basically, we add words one after another to the base tokens (alternating both extremities) until the distance between the quotee and the quoter is greater than the threshold. The parameter defining the threshold can also define whether it is static or depends on the current length of the word strings.

Once the boundaries are found, the different terms at the extremities are truncated (which means that the words that use them are removed from the quotation structure).

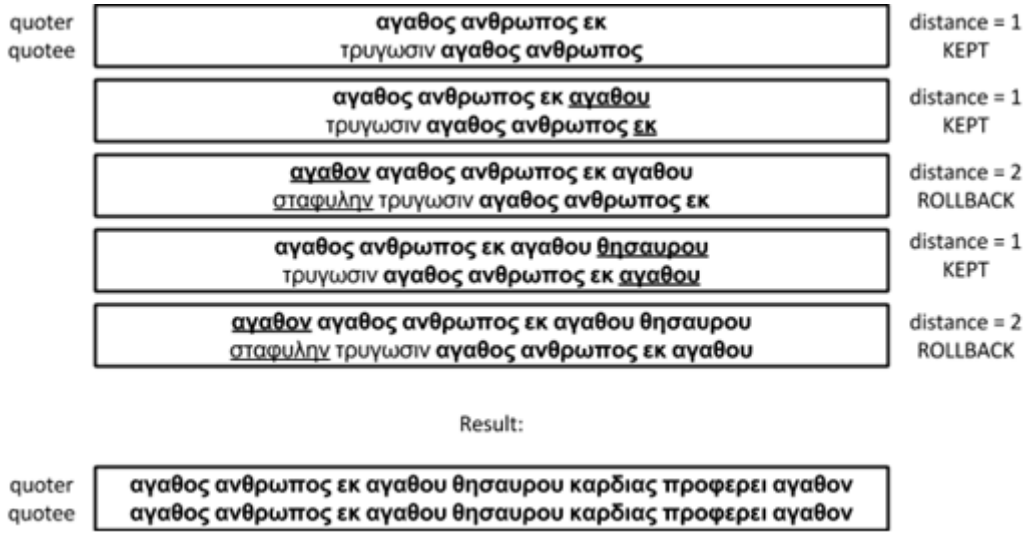


Figure 22. Illustration of the process used to search for the boundaries: words are added to both tokens (they are underlined here for clarity), and if the difference grows greater than the threshold, the addition is rolled back. The process stops when additions are rolled back for both of the extremities. Note that this illustration does not

show the underlying document structure, and that although the comparison is done on the processed terms, what is compared through them are two word strings.

Once the bounds of the quotations have been computed, the resulting quotations are returned to undergo post-processing.

4.1.3. Post-processing

Once main processing has provided a set of quotation candidates, post-processing step ensures that they are displayed in the form that has been requested, and that there are no doubles. Post-processing also includes user feedback that is used both to enrich the resources (for example the lemma or the non-quotes tables).

The first step of post-processing is merging the quotations that are too near one from another. The actual meaning of being too near is defined by a parameter, ranging from overlap only (two quotations are merged only if in both the quotee and quoter part they have at least a word in common) to proximity (they are merged if the amount of text between them -again in both the quotee and the quoter text- is less than a threshold).

The amount of text that has been chosen with the experts is 5 words: this means that we merge any quotations that have at most 5 words between them in both texts.

Figure 23 illustrates the different cases that can be met.

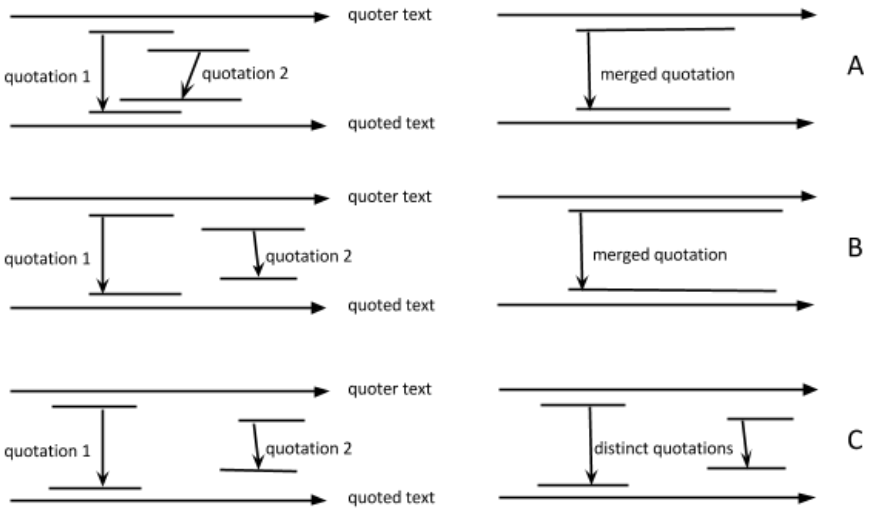


Figure 23. Illustration of the merging step. Quotations in case A) overlap in both texts, so they are merged. Quotations in case B) do not overlap, but they have few text between them, less than the specified threshold: they are merged too. Quotations in case C) are far enough one from the other to be considered distinct, and are thus not merged.

Once this merging is done, granularity is taken into account. Up to this step, quotations have been defined at the level of words (the boundaries are beginning and end words). Often the wanted granularity is higher level than this (for example, it can be sufficient to know what verses are quoted and not specifically what part of verses constitute the exact quoted text).

The second step of post-processing uses the structure of the documents to infer the correct-level reference to return. Figure 24 provides an example.

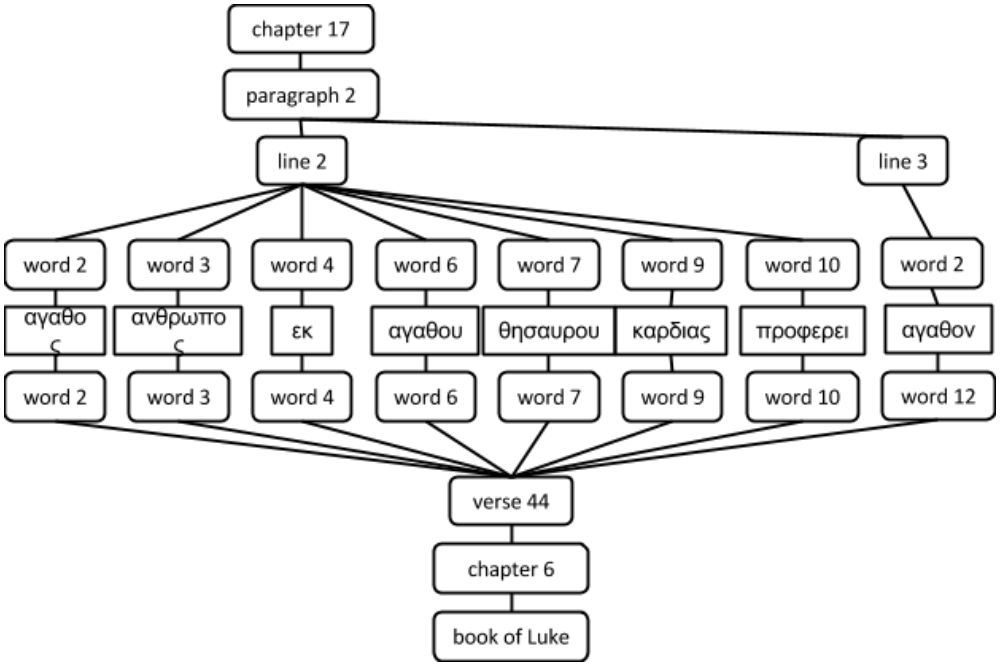


Figure 24. Underlying structure of the quotation. If the desired granularity is lines for the quoter text and verse for the quoted text, the answer will be that chapter 17, paragraph 2, lines 2 and 3 of the first text quote the book of Luke, chapter 6, verse 44. Even if only one word of line 3 is included in the quotation.

Then, the results are produced in the necessary format (html report, spreadsheet, database entries…) and can be validated or rejected. This last step can in turn provide valuable knowledge to improve the process, either by adjusting the parameters or by enriching the resources.

4.2. Statistical-semantic Resources

Since we lacked any semantic resource that we could have used, we tried building one from the 700 texts we had.

Our resource asserts a semantic proximity between two terms by pondering the probability of some semantic link between them. This probability is based on the concept of potential interchangeability, which means that where we have some term, we could have another term instead in some context.

This particular algorithm takes an amount N of the most frequent lemmas and splits the texts into units (verses for the Bible, sentences for punctuated texts, lines or n-grams for other texts). Then, for every lemma, a ‘spectrum’ is computed counting in how many units the lemma is found in co-occurrence with each of the N most frequent lemmas. The spectrum is normalised and is used as a feature vector for the lemma.

Then, to assert the semantic distance between two lemmas, we compute the Euclidian distance between their spectrums (which is then divided by $\sqrt{2}$ to stay in the [0,1] range).

4.3. Optimizing the algorithm

Term indexing and sharing

Because our Term class is basically, from a coding point of view, the java String class with new methods, we shared the instances between the whole execution context. This means that every time a given term appears in the documents, it is represented with the same object. This provides massive memory save, as well as a much quicker term comparison.

Besides, within the algorithm, we had to build several indexes to decrease time complexity (at the cost, this time, of memory consumption). Such indexes include for instance in which n-grams every term can be found.

Graph manipulation and indexing

As much as possible, we ensure that the structural graph of a document (*i.e.* a Passage) can be processed as if it were a tree, whereas in our model it is a hierarchical graph. This is achieved using the following two heuristics:

- The actual graph is in fact a bunch of trees that share the same leaves (words) but have each their own root depending on the problem at hand (using the approach of *facets* or *viewpoints*); for instance, the main document tree shares the same words as every tree representing a quoted passage. They are each passages of their own, but they share the same words.
- Whenever the graph is browsed from the leaves up, the parents that are returned while browsing are those that are in the main document graph. This means that we act as if branches of the main document tree were doubly linked while branches of the other trees were merely simply linked. This is actually featured in Figure 18.

Most calculations take advantage of the graph structure of the documents. However, we have several processes that have to work at the level of words, which are the leaves of the graphs. Therefore, to each relevant graph, we added an ordered list of its words as an index. The structure can still be accessed by browsing the graph, which allows for example to build a n-gram without browsing the graph, and then to compute just as easily where the words composing the n-gram are within the text structure.

Progressive pruning

During the process, we find many results that will eventually be rejected. Rejecting them as soon as possible in the process is a key to overall performance. Since we learn much on these results the longer they are processed, we set several gates along the process to filter out the results that at each step we already know will not be kept at the end.

Overall:

- The exact range of what encompasses nonquotes (from stopwords only to all recurring formulae) is provided as a parameter to the algorithm. At the very beginning, every n-gram that is composed exclusively of nonquotes (according to this parameter) is marked as a nonquote and as such will not be processed at all.
- During the first step of the main processing, it is possible that the amount of candidates grow so fast that memory is in risk of being saturated. To avoid this issue as much as possible, we set a threshold of 100 000 candidates before triggering degraded mode. Degraded mode only keeps one of several overlapping matches. It is most probable that they would result in the exact same boundaries anyway, and we only need to keep them once. Because this filter comes with a quadratic complexity, and because when invoked it is likely to be so several times, we set two measures to prevent it to be too time-consuming. Firstly we increase the threshold by 10 000 every time degraded mode is invoked on the same set of candidates, building a buffer before resorting to it again. Secondly, we analyze the candidates in a last in, first out basis, and we stop the analysis as soon as 10 consecutive candidates are not filtered (to ensure that we will not process the first ones each time). We called it degraded mode because we could technically lose matches when using long n-grams with a high tolerance (for instance, we could reject a n-gram whose only matching terms are at the end because its beginning overlaps with the end of a n-gram whose only matching terms are at the beginning).

Other

- Looking for the boundaries of a quotation is a time-consuming process, in that we have to chain several tests to assert the exact position. Our first choice was a progressive extension (extending the analysed text a word alternatively at each end), but as tolerance increases, long matches become more common and complexity quickly approaches the worst-case o(m.n), n being the text size in words and m the amount of quotations to study. We thus adopted a quicker exponential/dichotomic approach :
    - we set a parameter p to 1 at the beginning
    - we alternatively add p words at each end
    - while the result remains within the tolerance parameters for at least one end, we double p and continue (if it does not remain within the tolerance for any end, we rollback its particular addition)
    - while it does not remain within the tolerance for either end, we rollback both additions, halve p and continue unless p is less than 1
- Even the exponential/dichotomic approach is challenged by very long texts and high tolerance, when the quotation finally returns the whole documents (which is as worthless as it is time-consuming). Therefore, we had to set a parameter to describe at which size a quotation would be considered too long to be credible. This heuristics basically transfers the work of finding the boundaries of these particular quotations to the expert (who will, more often than not, merely invalidate the quotation). The value for this parameter that has been validated by the experts is 500 words.

4.4. List of Parameters

Level of language processing

This first parameter sets the level at which the terms will be processed before the whole process. There can be steps where some level of processing is used on top of this (for instance, when searching for stopwords, lemmatisation is necessary as we have a list of stop-lemmas). But mainly, the process will use terms at the specified level of processing.

- None: the terms will be used as is;
- Normalisation: accents and punctuation (except for sentence enders) will be taken out, and uppercase letters will be turned to lowercase;
- Lemmatisation: all terms will be replaced by their lemmas.

It is worth noting that depending on the edition, texts can be originally in a normalised form by the simple fact that the edition does not add lowercase, accents and punctuation to an antique text that did not contain them.

We are working on stemming, but it is not available currently.

Importance of word order

- Word order is important
- Word order is unimportant

Since word order within a sentence is unimportant in Greek, which is based on inflection as presented in section 2.1. To see which impact it has in quotation finding (assuming that a quotation can reorder the words without changing the meaning of the sentence), we allowed to specify its importance.

It is worth noting that in a text without punctuation, it is impossible for the algorithm to assert when a sentence begins and ends: syntactic analysis is not included. In these cases, word order must be set as important.

Similarity measure

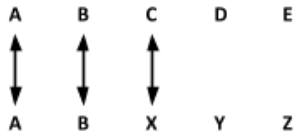
The algorithm can use a similarity measure to infer relations between terms (see section 4.2.3.). Therefore, the policy on the use of this measure can be set to:

- Idle: the measure will not be used, so terms are either identical (textually) or different

Figure 25. Similarity computing without a similarity measure (terms are either identical or infinitely different).

- Clustering: terms that are sufficiently similar (as defined by a threshold that must be provided) are grouped and considered as identical. The challenge is to set the

threshold. Too high of one barely makes any difference, while a threshold too low (even as high as 0.95) rapidly results in several massive clusters that do not make sense.

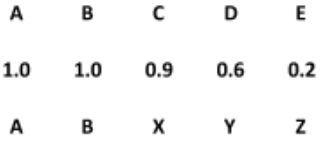


Similarity: 3/5

	A	B	C	D	E	X	Y	Z
A	1.0	0.1	0.2	0.7	0.5	0.3	0.4	0.1
B		1.0	0.5	0.3	0.2	0.4	0.1	0.6
C			1.0	0.1	0.2	0.9	0.1	0.3
D				1.0	0.3	0.4	0.6	0.9
E					1.0	0.1	0.4	0.2
X						1.0	0.2	0.3
Y							1.0	0.7
Z								1.0
Clusters with threshold 0.9:		{A}	{B}		{C,X}		{D,Z}	{Y}

Figure 26 Similarity computing with a clustering approach. Note that if word order is not deemed important, the similarity between D and Z will bring the overall similarity to 4/5.

- Passage similarity: similarity will be used at the level of passage matching (and a score of similarity between the passages will replace the simple amount of differences).



Similarity: 3.7/5

	A	B	C	D	E	X	Y	Z
A	1.0	0.1	0.2	0.7	0.5	0.3	0.4	0.1
B		1.0	0.5	0.3	0.2	0.4	0.1	0.6
C			1.0	0.1	0.2	0.9	0.1	0.3
D				1.0	0.3	0.4	0.6	0.9
E					1.0	0.1	0.4	0.2
X						1.0	0.2	0.3
Y							1.0	0.7
Z								1.0

Figure 27 Similarity computing with a clustering approach. Similarly to Figure 26, if word order is not kept, the overall passage similarity is brought to 4.2.

Other parameters for the similarity measure include the size of the spectrum and the similarity threshold (if clustering).

Threshold for matching passages

Quotations can reformulate part of the quoted text, changing, adding or deleting words. Strict identity between the fragments in the source and the quoting texts can find the most obvious quotations, but then a compromise must be found between too much leniency (and many false

positives in the results, so poor precision) and too much strictness (and few quotations found, so poor recall).

The algorithm starts by searching for similar n-grams and then looks for the boundaries of the similar part. Therefore, allowing difference can be done in three ways:

- Strict identity is needed: after language processing and similarity computing, all words must be identical from a passage to the other (if using passage similarity, the similarity between the passages must be 100%). Words can still be reordered if the according parameter is set.
- Difference must be lower than a threshold that is proportional to word count: a number of differences (different terms, different clusters of terms or a lower than 100% similarity score using passage similarity) are allowed without discarding the match, and the amount of differences (or the corresponding allowed decrease in passage similarity) increases proportionally to word count (so 3 allowed differences in the initial 10-gram will translate to 6 allowed differences once the passages reach 20 words).
- Difference must be lower than a static threshold: a number of differences (or a lower than 100% similarity score) are allowed without discarding the match, but the amount is the same in the initial n-gram and in the final, potentially much longer, quotation. This is useful for instance to define a really permissive initial search (such as 3 common terms among 10 words) without propagating it when searching for the actual quotation boundaries.

Non-linear relation between threshold and word count has been envisaged, but we did not develop it further.

Initial policy on stopwords

This parameter defines whether stopwords are initially deleted from the text (they still count as words in the graph, but will be jumped over when searching for consecutive words), or kept for the analysis. Which list is used can be set either to the handmade 16 lemmas list, or to the list of around 600 lemmas that appear in every of the 700 texts of our corpus.

It is worth noting that we do not delete nonquotes here: they can still be useful for the analysis, as they can still be part of a quotation.

Policy on multiple quotations

(Ernst-Gerlach *et Al.*, 2008) specifies that a given passage can only quote a single passage. Speaking of the Bible and patristics, this is not true, and many quotations quote, in fact, several passages. However, it can still be harmful to allow for it, so we kept it as a parameter, allowing us to toggle it on and off.

- A given passage can quote a single passage only: when several candidates are found, the one with the most common terms will be kept
- A given passage can quote multiple passages

Post-processing will still process multiple quotations in the cases explained in section 5.2.

Policy on nonquotes

While keeping or discarding stopwords before starting the analysis is already covered, there is still a case that must be taken into account. A non-negligible part of the found quotations are, in fact, mere recurrences among the texts of expressions or words that are common either in the language, or in the field of the corpus, and were listed as nonquotes by the experts. These findings that rely exclusively on nonquotes are typically not quotations (and even if they were, they would not be recognised as such). The algorithm can, if specified, filter out these results. Alternatively, it can use the list of stopwords defined by a statistical analysis of the entire corpus instead.

The filtering occurs after initially finding a match between two n-grams, and before attempting to find the boundaries.

5. RESULTS

In order to evaluate our ability to retrieve references in Koin\'{e} Greek, we tested our algorithm on two experiment sets, both consisting in a source text, a quoting text, and a control set of known references. In this section, we will first present some discussion about the metrics that can be used for this evaluation, then we will present our experiment sets and perform a predictive analysis, and finally we will present how our algorithm performs on these sets. The experiment sets as well as the software implementing the algorithm are available on our website (http://liris.cnrs.fr/dire/wiki/doku.php?id=greek_reuse_toolkit).

5.1. Metrics

Precision is defined as the proportion of found answers that are good answers, and recall as the proportion of good answers that are found. In our case, however, asserting whether a found quotation is a good result is difficult, as is asserting whether a quotation was found.

5.1.1. Matching found results with the control set

Firstly, we have a list of results to find -results that are taken from already published material. However, these results are often approximate, to the point where one of our goal is to find the actual boundaries of the quotations within them. Moreover, we know where these results begin, and often not where they end. So, how do we decide that a quotation is good? If we know that paragraph 2, line 5 quotes a passage, and we find this passage quoted by paragraph 2 lines 4 to 6, is it good? Line 5 but only the three first words? From line 5 last word to line 6 second word?

Fortunately, in our small test sample (329 references total, including allusions that do not have any word, lemma or otherwise in common), a manual work has been done to ensure that we know the boundaries at the granularity of words. But how is this number sufficient to assert the performance of our algorithm?

We eventually decided that if a quotation that we found overlaps with a quotation that must be found, we will take it as a good result. Obviously, overlapping must occur in both texts.

5.1.2. Managing false negatives

Secondly, another of our aim is to find new quotations -quotations that have not been detected, or documented, in the previous centuries. Digital processing allows being exhaustive, if biased by the limits of an algorithm, and we already found several new quotations during our tests. While recall does not suffer from this aspect of the project, we have to first pass the results to a second process -in this case, a manual study- to assert whether a 'wrong' result (as defined by precision) should actually not been found, or whether it is a new result, which should enrich the original list, and thus irremediably corrupt the independence between the expected results and the algorithm that we want to test. Because if the algorithm finds new answers, how many new answers has it not found? We cannot artificially enhance its performance by biasing the test sample.

We can of course tag these results as 'good but not mandatory', thus keeping the independence, and simply not count them. But it still means that we have to manually control the results before asserting a precise precision score.

5.1.3. Precision, recall and self-quoting documents

Lastly, we have a specific bias that comes from the very text we study. While it may not be infrequent for a given text -or even more likely for a given author- to self-quote, the Bible is a web of self-quotations. Of course, self-quotations are chronologically oriented -newer texts quote more ancient texts; but it still means that when we find that some non-biblical text quotes the Bible, there are cases when more than 20 texts use the same formula -it is obviously a quotation of course, but in a recurring theme. In this case, which of the texts is the right one? The one in the list of results is, obviously. But can we hold an algorithm -that basically matches strings- accountable for deconvolving the strata of successive quotations and finding the right author that was quoted, as opposed to those he quoted and those who quoted him? In many cases, a pragmatical analysis is necessary. In some extreme cases, research is necessary.

This bias was more difficult than the others to overcome. We first sought to differentiate real quoted text from mere recurring formulae -either in the language or in the specific corpus- and it brought us to expand the notion of stopwords to nonquotes as defined in section 4.2.2. Then, we looked with the experts for a threshold for the amount of candidates; either there was less candidates, and they were all considered right, for needing a manual validation (we had actually quotations that were considered enriching by the expert, so quotations that they wanted to have, even if not formally the right one); or the amount of candidates exceeded the threshold and they were all considered false, for having (maybe) found the right answer for wrong reasons. Intertwining both processes, searching for nonquotes in the numerous candidates, was very effective at building our lists of recurring expressions. The drawback, however, was the risk for them to become highly sensitive to the specific kind of text we were testing with, and to bias the results once more. However, this was solved by integrating the search for recurring formulae as a learning process in the algorithm itself.

We thus defined both a raw score -precision and recall using the amount of found answers- and a refined score, factoring that last bias and the according threshold. It is worth noting that while in the first case a single number serves as the dividend of both precision and recall, in the refined version the dividend of precision is greater than the dividend of recall (factoring

the numerous cases where more than one answer is deemed correct). Besides, the impact of this 'refining' on the recall depends on the amount of found answers (which normally only impacts precision). This devalues sets of parameters that increase the amount of candidates: in addition to being subject to low precision, they also get low recall.

5.2. Predictive analysis

5.2.1. Experimental sets

We defined two experimental sets to test our algorithm and the impact of its parameters. These experiments differ both on their scale and on the accuracy of the control set of quotations.

- The first experimental set uses a single document, 'Quis dives salvetur' (around 10 000 words), quoting the Bible composed of the Septuagint and the Greek New Testament (around 850 000 words). To control our results, we have a narrow set of 329 references, the exact boundaries of which have been produced. These references include 19 allusions. We trained our algorithm on this document, which means that the sets of nonquotes is relevant (although probably still not exhaustive) in its case.
- The second experimental set uses a whole author's publication (namely Philo of Alexandria, so around 460 000 words; even though he is not a Church Father, his work is relevant), quoting the Septuagint. We only use the 8168 known references to the Torah (the first five books of the Septuagint, circa 150 000 words) as a control set, and we only know where these references begin within the Philo texts. We do not know how many of them are allusions. We did not train our algorithm on these documents, which means that the set of nonquotes is of lesser help here.

5.2.2. Similarity breakdown

Expected results depend greatly on the structure of references. In this part, we analyse the corpus of references that we were given for our experimental sets and try predicting which approaches may prove successful, in particular insofar as word processing and tolerance are concerned.

The key metric here is the amount of terms in common between quoting and quoted passages. If they have consistently many terms in common, we will be able to find them, whereas in the opposite case, we may only find them among an overwhelming amount of false positive (which means little precision, and with our refined measure, it is not even sure than we do have a good recall).

However, while finding common terms is the common way of searching for quotations (allusions elude this kind of search though), the actual similarity between terms may be a bit more subjective than simply checking the character string. And then, if we consider the set T of terms appearing in the documents, we can define a transformation of this set into another set T' that will yield different common terms -maybe more.

In our cases, we defined three parameters for such a transformation:

- Level of language processing: two terms may not be identical but they can still share the same lemma, for instance; or appear in the same contexts, which is a commonly used similarity metric;
- Level of filtering: terms may be common in both the senses that they are in both passage, yes, and then in pretty much all other passages we can choose. Filtering stopwords is a powerful tool to keep results relevant, and we also defined usual and recurring terms lists, which lets us define whether a term is significant or not;
- Level of tolerance: while a quotation may have many terms in common in both passages, they may not be found in succession; we can then accept a non-null edit distance, up to some threshold that again keeps results relevant; even if we did have strictly identical passages it would be useful, all the more given our similar in meaning but not identical ancient references.

Figure 28 and Figure 29 present the base reference structure (that is, with stopwords, infinite tolerance and varying levels of processing) in both our experimental sets. We can achieve up to 65-75% of references having at least 3 common lemmata, which is not a bad expected recall given the circumstances. However, it is obvious that we will not reach this score with the refined metrics.

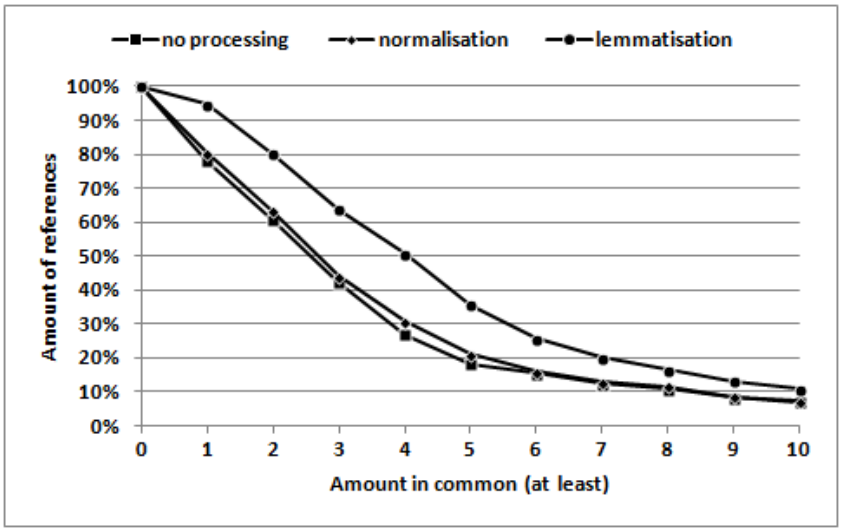


Figure 28. Structure of the Quis Dives reference set - no filtering, infinite tolerance.

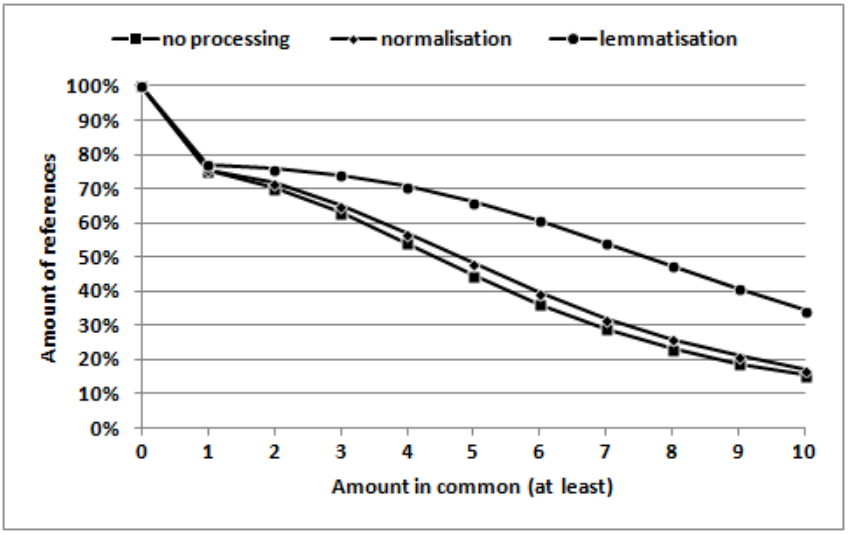


Figure 29. Structure of the Philo reference set - no filtering, infinite tolerance.

Figure 30 and Figure 31 show what happens if we take out the stopwords. We lose about half of the previous matches, and even if we only search for two common terms, we are barely over a 50% expected recall.

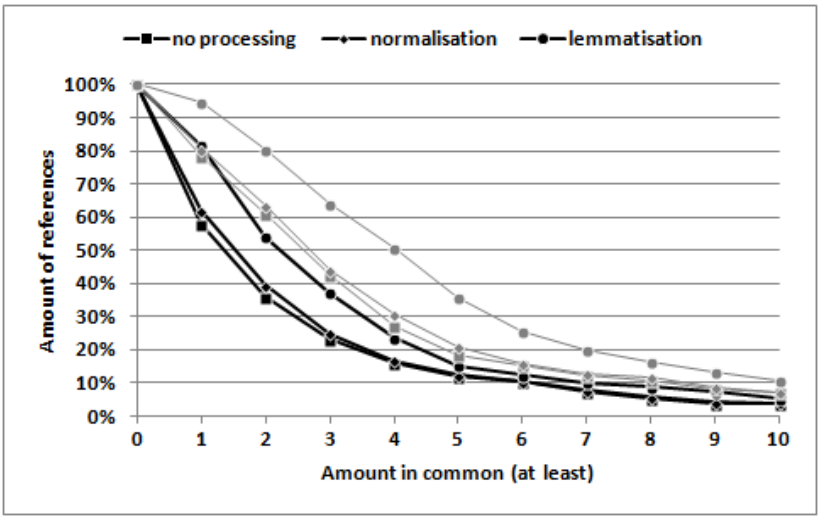


Figure 30. Structure of the Quis Dives reference set - no stopwords, infinite tolerance (the lighter curves are those from Figure 5.1).

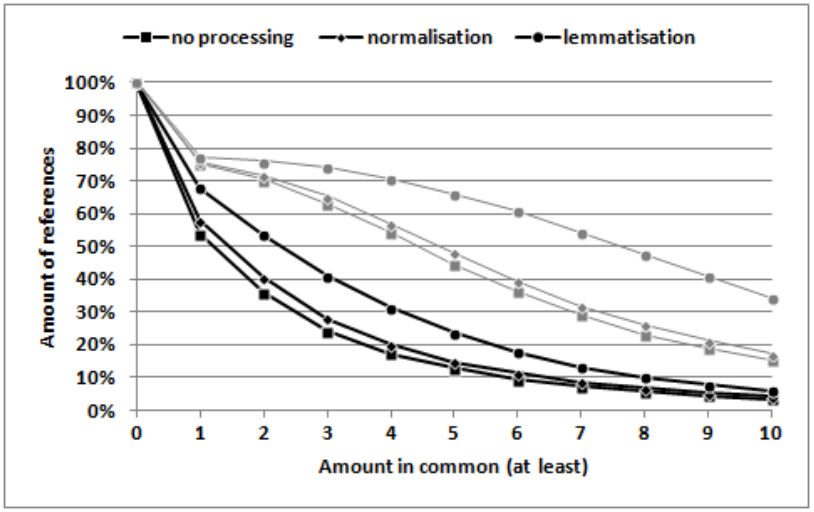


Figure 31. Structure of the Philo reference set - no stopwords, infinite tolerance (the clear lighter are those from Figure 29).

Then, Figure 32 and Figure 33 show the expected results with only significant terms. Noisy though usual and recurring terms (or lemmata) may be, they happen to hold the majority of the matches, only leaving us 10-15% of references having at least two terms in common.

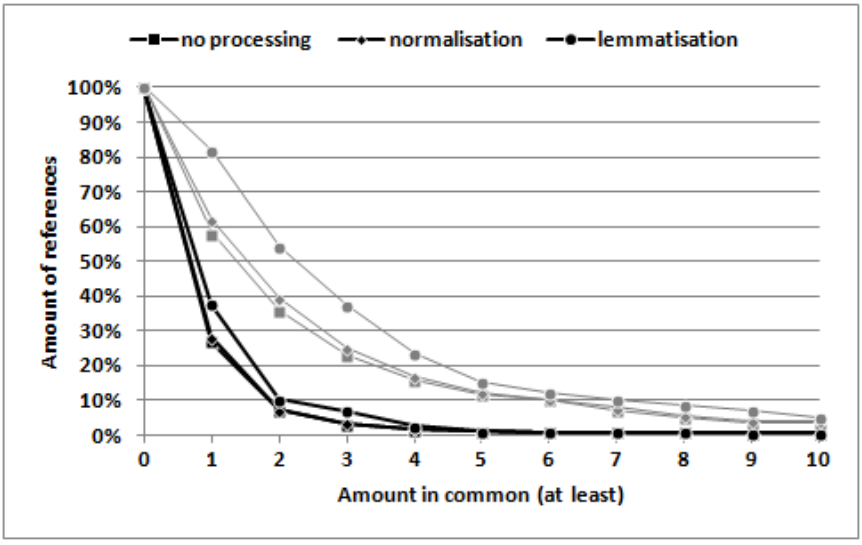


Figure 32. Structure of the Quis Dives reference set - significant words only, infinite tolerance (the lighter curves are those from Figure 30).

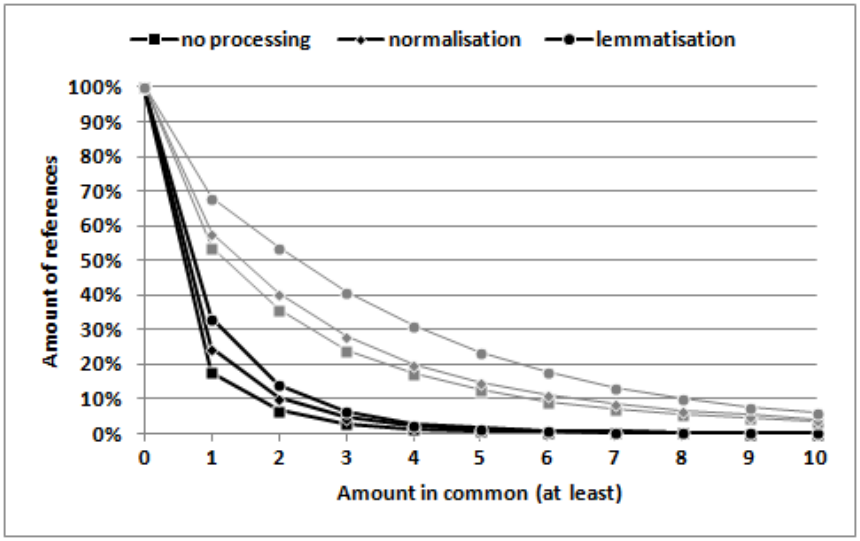


Figure 33. Structure of the Philo reference set - significant words only, infinite tolerance (the lighter curves are those from Figure 31).

Finally, Figure 34 and Figure 35 show the breakdown of consecutive common terms (that is, the size of the longest common chain of terms present in both the quoting and the quoted passage). Around 3% of quotations have a chain of three significant terms (which still may have stopwords and usual / recurring words inbetween), and 6-7% have a chain of two. As previously, the usual and recurring terms are necessary for most of the matches.

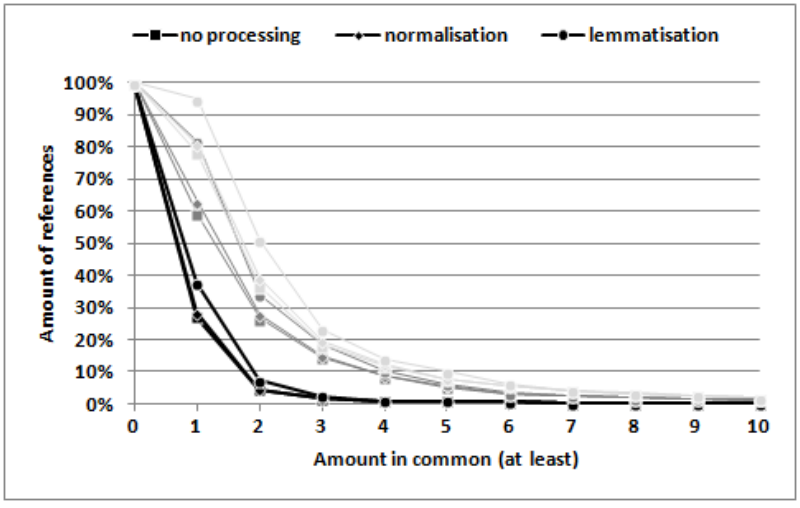


Figure 34 Structure of the Quis Dives reference set - no tolerance (in order from darkest to lightest: significant words only, no stopwords, no filtering).

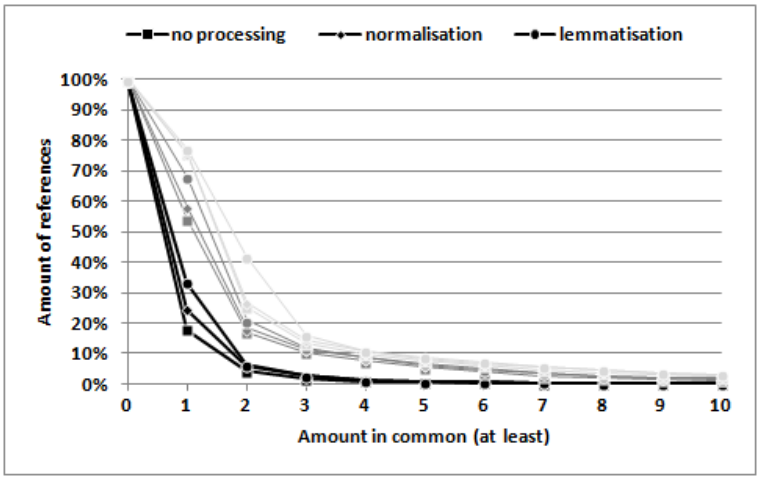


Figure 35 Structure of the Philo reference set - no tolerance (in order from darkest to lightest: significant words only, no stopwords, no filtering).

The conclusions of this analysis are that:

- Lemmatisation is a must-have: he curves using lemmata are always significantly higher than the others;
- Tolerance is key: searching for consecutive common terms is a waste of time; it remains to be seen which tolerance threshold gives the better f-measure;
- Filtering is a trade-off: while filtering stopwords is likely to be necessary, going further and filtering usual or recurring terms may not be an optimal solution;
- We are bound to have low recall if we want any relevance in the results.

5.3. Experimental Analysis

In order to compare the result of our algorithm with what we can expect to find, we mapped the parameters of our algorithm to the three transformation parameters. Then, we chose several values for each of these transformation parameters. Overall, all possible combinations of these three parameters and their values have been tested on both experiment sets.

5.3.1. N-gram size

As the previous section showed, the n-gram size remains the biggest parameter in the process. Each size will have its own equilibrium between precision and recall, but at the end, we will have to decide how many quotations we are able to manage.

Most of the approaches dealing with quotation retrieval start with searching for two common terms and elaborate from there. Our tests, regardless of the other parameters, returned with at least several tens of thousands of matches. we could have sorted them like other approaches do to keep only the most relevant, however most of even those were disqualified by the refining process presented in section 5.1 (because the texts are highly self-quoting). Since in addition the recall was limited anyway (only at most 55% of the known quotations have two non-stopwords in common, as shown in section 5.2), we decided that such a drop in precision was not worth it in the current state of our algorithm.

Searching for more than 3 common terms did give us interesting results, but they were very few (less than a hundred unless we allowed the stopwords in). Even though the f-measure itself was better with 4- or 6-grams than with 3-grams, due to the increased precision, the low amount of matches (and subsequent recall) made it inferior. Even though self-quotations, albeit still present, were not nearly enough to cause the refining step to alter the results.

We therefore chose 3-grams as the base unit for our algorithm. The data provided in the following sections will all originate from experiments with the n-gram size set to 3.

According to the predictive analysis, we should be able to get a maximum recall equal to the proportion of references having three common terms. Figure 35 presents this theoretical maximum recall. The actual recall is altered by the distance between these common terms (remember how few of these references had three consecutive common terms) and the recall refining.

		Quis Dives			Philo	
	all words	no stopword	only significant	all words	no stopword	only significant
no processing	42.25%	23.10%	3.04%	62.90%	24.00%	2.96%
normalisation	44.07%	24.92%	3.34%	65.17%	28.01%	4.84%
lemmatisation	63.83%	37.08%	6.99%	73.96%	40.88%	6.42%

Table 1 Theoretical maximum recall for 3-grams

The first column corresponds to experiments with no filtering, while the second means simple or double filtering. The third column does not correspond to any parameter we tested and is only put here for reference.

5.3.2. Level of tolerance

It is the amount of differences that we allow in a match before discarding it. We chose the following modes, and thresholds :

- Strict match: for two passage to be declared a potential reuse, they had to contain the exact same words in the exact same order.
- Unordered match: contrary to the previous case, while the exact same words were still necessary, the order in which they occurred in the respective passages was not considered important.
- Absolute tolerance: we allowed a fixed amount of differences. We chose the amount of 7 allowed references for this test. Previous experiments showed little difference in the results for numbers in the range 5-15 (Using 3-grams as the base search unit, 7 references meant that we searched for 3 common terms within 10 words).
- Relative tolerance: we allowed an amount of differences proportional to the length of the match. We chose 70% for this test, again due to the little difference in results for values in the range 50%-150%.

The results along precision and recall (our refined version) are presented in Figure 36 (for the Quis Dives experiment) and Figure 37 (for the Philo experiment). The maximal found values are provided in Table 2. While the gain in recall is around 25% in both experimental sets

using tolerant matching, there is a loss in precision to compensate (especially in the Philo set, which has the bigger control set).

If we now compare the maximal f-measure that we were able to obtain with each parameter, there is a slight gain in using a tolerant approach, as opposed to a strict one, but the gain is not that significant (even less with the Philo experiment set, where there is very little gain, if at all).

Tolerance is therefore not as impactful as we expected it to be. The question arises whether this is caused by the use of the refined metrics: if tolerant approaches provide more matches but in doing so more quoted candidates for a given text, we should expect a diminished recall. Table 3 presents the same results as Table 2 with the usual metrics, and show the same patterns. It is therefore not not the case. We should conclude that tolerance, while having a great expected gain, is not so effective in reality.

		Quis Dives			Philo	
	max precision	max recall	max f-measure	max precision	max recall	max f-measure
strict	8.65%	11.85%	0.68%	15.49%	8.16%	0.98%
unordered	9.62%	12.77%	0.91%	15.07%	7.53%	0.98%
tolerant (7)	8.28%	16.41%	1.05%	12.60%	10.20%	1.00%
tolerant (70%)	12.41%	16.11%	1.02%	12.02%	10.14%	0.95%

Table 2 Maximal results obtained for various levels of tolerance (refined metrics).

		Quis Dives			Philo	
	max precision	max recall	max f-measure	max precision	max recall	max f-measure
strict	18.11%	18.84%	1.60%	33.11%	13.05%	2.02%
unordered	15.16%	22.80%	1.71%	32.15%	14.60%	1.99%
tolerant (7)	22.88%	27.96%	2.23%	25.40%	16.36%	2.02%
tolerant (70%)	22.63%	29.18%	2.13%	25.19%	17.76%	2.06%

Table 3 Maximal results obtained for various levels of tolerance (usual metrics).

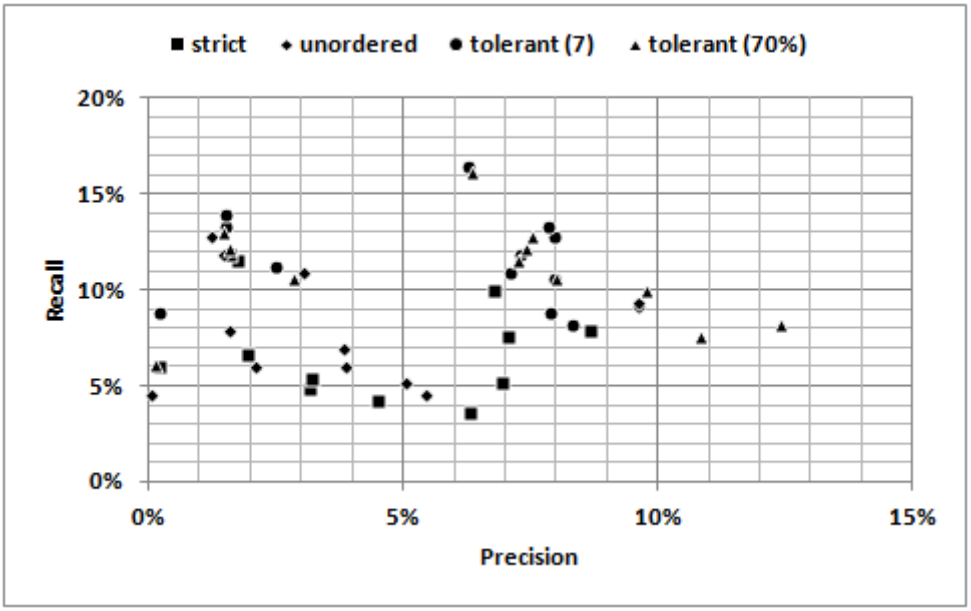


Figure 36 Results of the matching process for various levels of tolerance with the Quis Dives experiment. The different points using the same level of tolerance indicate different choices for the other parameters.

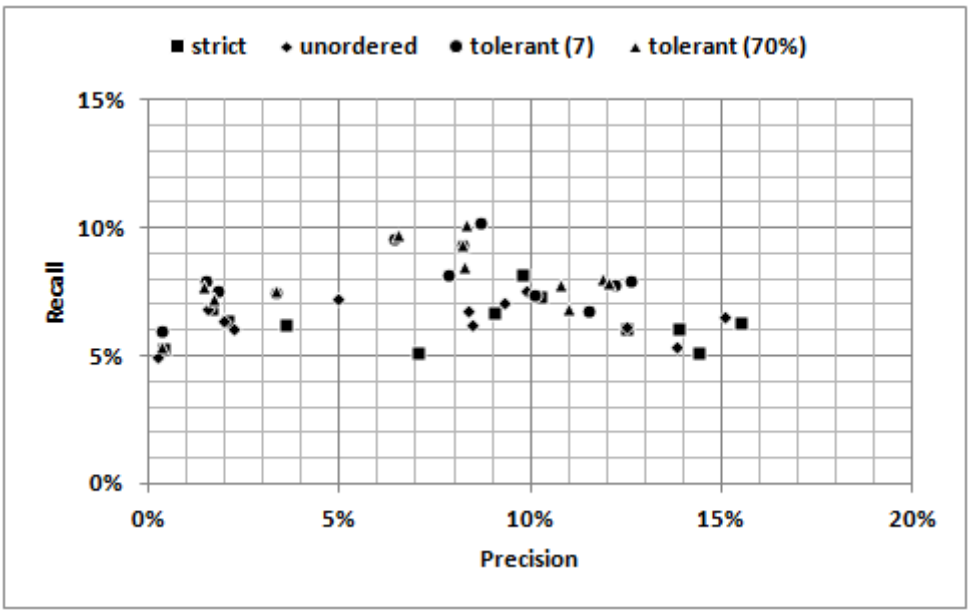


Figure 37 Results of the matching process for various levels of tolerance with the Philo experiment. The different points using the same level of tolerance indicate different choices for the other parameters.

5.3.3. Filtering

Filtering directs the way terms are weighted during the matching process. The choices are:

- No filtering: both filtering steps in the algorithm are disabled, and every term and match is kept. Especially, stopwords are not filtered.

- Simple filtering: before splitting the documents into n-gram tokens, we discard all the stopwords. This means that stopwords will not count as neither similar nor different terms, as they simply will not be there anymore when the processing takes place. This also means that the final results can have more similarities or differences than what the algorithm ends up with, depending on the presence or absence of stopwords in them.
- Double filtering: in addition to the above, after the initial search for matching n-grams, all matches that only contain usual or recurring terms will be discarded.

The results along precision and recall (our refined version) are presented in Figure 38 (for the Quis Dives experiment) and Figure 39 (for the Philo experiment). The maximal found values are provided in Table 4, while Table 5 provides the results without refining for reference. The impact of stopword filtering is obvious, as can be seen in the figures that show clearly distinguishable clusters for each value of the parameter.

Precision is very low without filtering, and if there is a loss in recall that should be expected (in section 5.2.2 there were more references that had common terms when including stopwords than when not), it is alleviated by the refining process: the correct matches that benefit from stopwords are likely to be lost in the noise anyway.

The impact of double filtering is much more on par with a compromise between precision and recall. The f-measure is similar with simple and with double filtering (actually a bit better with double filtering in the Philo experiment set), while precision and recall themselves vary (more filtering meaning more precision and less recall).

		Quis Dives			Philo	
	max precision	max recall	max f-measure	max precision	max recall	max f-measure
no filtering	1.75%	13.98%	0.21%	2.04%	7.94%	0.13%
simple filtering	9.62%	16.41%	1.05%	10.20%	10.20%	0.88%
double filtering	12.41%	10.64%	1.02%	15.49%	8.04%	1.00%

Table 4 Maximal results obtained for various levels of filtering (refined metrics).

		Quis Dives			Philo	
	max precision	max recall	max f-measure	max precision	max recall	max f-measure
no filtering	2.58%	29.18%	0.43%	2.91%	17.76%	0.37%
simple filtering	16.96%	20.36%	2.11%	16.23%	12.37%	1.52%
double filtering	22.88%	11.85%	2.23%	33.11%	8.58%	2.06%

Table 5 Maximal results obtained for various levels of filtering (usual metrics).

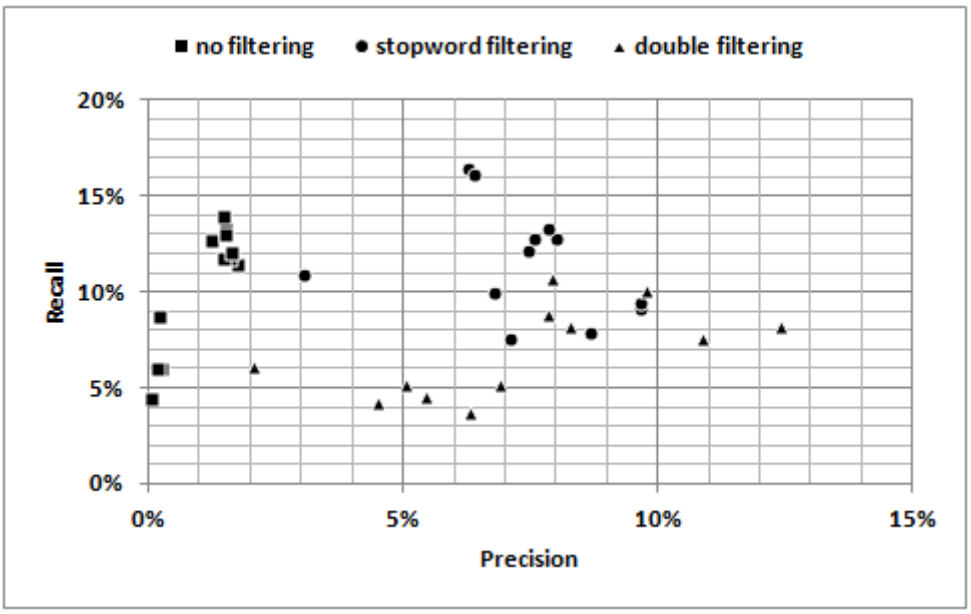


Figure 38 Results of the matching process for various levels of filtering with the Quis Dives experiment. The different points using the same level of filtering indicate different choices for the other parameters.

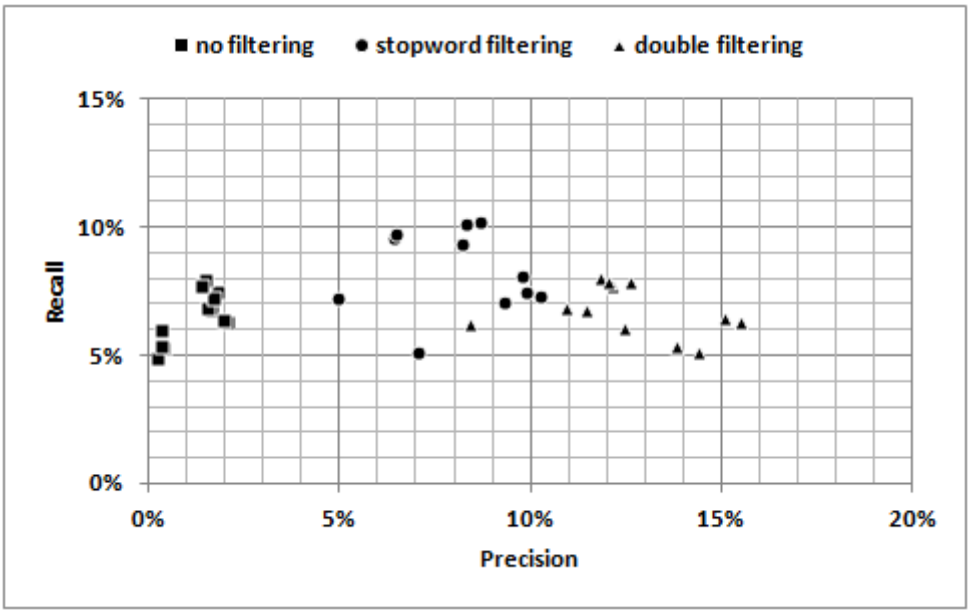


Figure 39 Results of the matching process for various levels of filtering with the Philo experiment. The different points using the same level of filtering indicate different choices for the other parameters.

5.3.4. Processing

Text processing defines the way terms will be matched. Lists such as stopwords, usual words and recurring words will still use the lemma no matter which level of processing is chosen. We tested the following levels:

- No processing: terms are used as they are.

- Normalisation: uppercase and diacritics are removed.
- Lemmatisation: lemmata are used instead of terms.

The results along precision and recall (our refined version) are presented in Figure 40 (for the Quis Dives experiment) and Figure 41 (for the Philo experiment). The maximal found values are provided in Table 6, while Table 7 provides the results without refining for reference.

There is no clear pattern to be seen in the figures. Normalisation performs generally better than no processing at all, but the impact of lemmatisation is mitigated by the amount of noise it produces. While there is a sizeable increase in recall, especially without refining this recall, it does not translate in the f-measure at all.

		Quis Dives			Philo	
	max precision	max recall	max f-measure	max precision	max recall	max f-measure
no processing	10.85%	13.37%	1.01%	14.37%	9.41%	0.84%
normalisation	12.41%	13.98%	1.05%	15.49%	10.20%	1.00%
lemmatisation	12.41%	16.41%	1.02%	15.49%	9.76%	0.95%

Table 6 Maximal results obtained for various levels of processing (refined metrics).

		Quis Dives			Philo	
	max precision	max recall	max f-measure	max precision	max recall	max f-measure
no processing	22.76%	22.19%	2.08%	33.11%	14.41%	1.82%
normalisation	22.88%	23.40%	2.23%	30.86%	15.29%	2.06%
lemmatisation	22.88%	29.18%	1.64%	33.11%	17.76%	1.53%

Table 7 Maximal results obtained for various levels of processing (usual metrics).

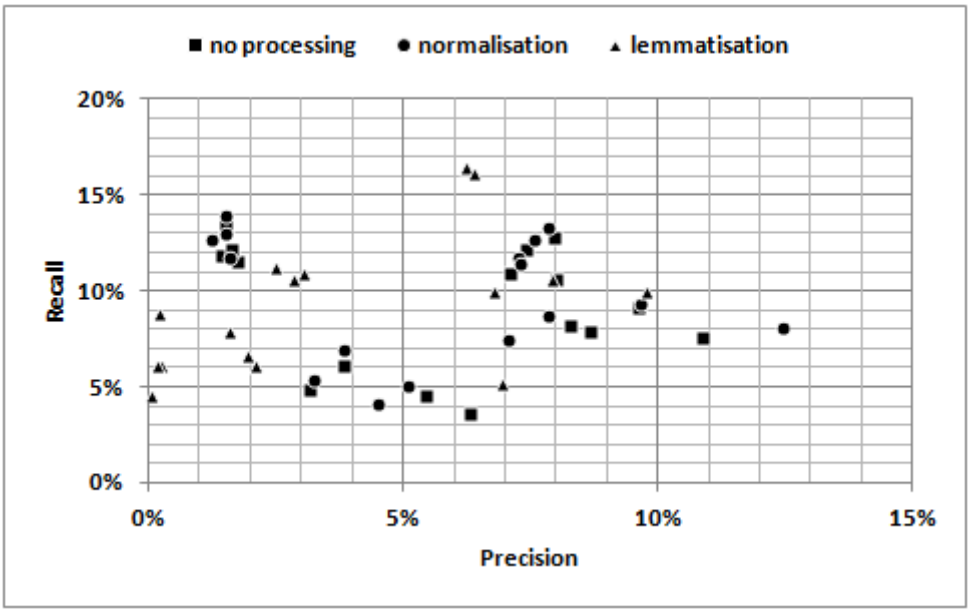


Figure 40 Results of the matching process for various levels of processing with the Quis Dives experiment. The different points using the same level of processing indicate different choices for the other parameters.

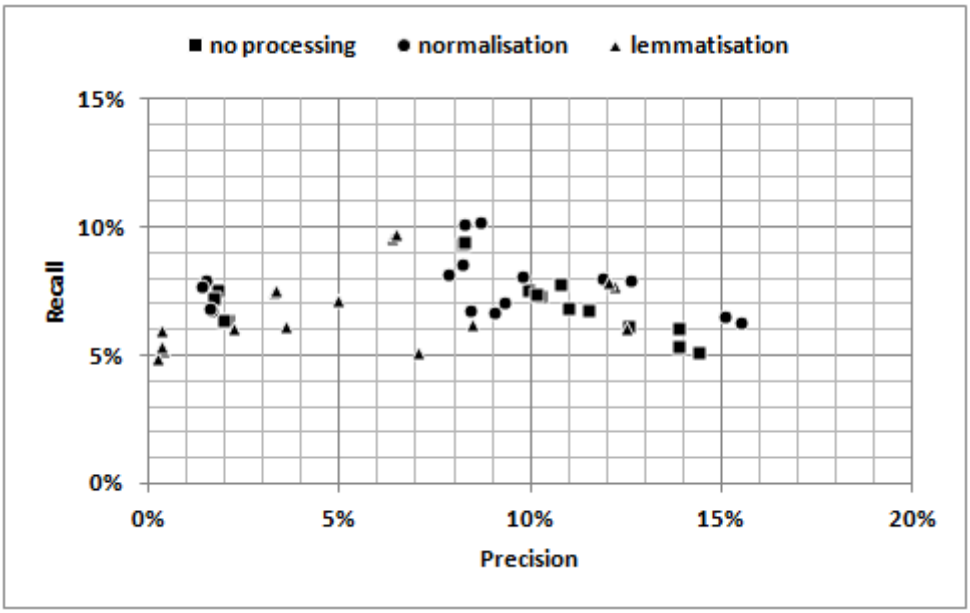


Figure 41 Results of the matching process for various levels of processing with the Philo experiment. The different points using the same level of processing indicate different choices for the other parameters.

6. CONCLUSION

In this paper we presented our work around the Biblindex project aiming to create a citation search framework in ancient Greek texts.

We presented our approach, proposed several solution on text parsing, modeling, treatment and processing.

We discussed the issue of detecting quotations within ancient documents. We took into account the specificities of language, culture (through the practices of quotation) and of the corpus itself. Using several approaches that were used in similar cases, we evaluated the main tools offered by statistical computation and found that even though they were not a sufficient solution, they worked as usual in the case of quotations that show text similarities.

This work leaves us three options to increase our performance at detecting quotations in ancient Greek texts. The first two revolve around the heuristic of merging similar terms using statistical methods. We can use our partnership with Greek experts to build a stemmer for this language, and we can use new semantic algorithms to improve our similarity measure. Increasing the amount of common terms may allow us to increase the similarity threshold of quotation detection and improve precision. The third option is finding heuristics within the metadata that can be obtained on the texts themselves. For example, if we know that some text is a commentary of some book – and there are many such cases, we can rely on this information to resolve most cases of multiple source candidates.

Another perspective is the study of the efficiency and scalability of statistico-semantic methods. We are currently working on the implementation of methods such as word2vec (Mikolov,et al.,2013) and the improvement of memory efficiency of the algorithm of (Mousselly-Sergiehet al.,2013) based on stream processing methods.

7. REFERENCES

7.1. Scientific papers

Bao, J.P., J.Y. Shen, H.Y. Liu, X.D. Liu. « A fast document copy detection model ». *Soft Computing - A Fusion of Foundations, Methodologies and Applications*. Vol. 10, n°1, p. 41-46, 2006.

Büchler M., Geßner A., Eckart T., Heyer G., « Unsupervised Detection and Visualisation of Textual Reuse on Ancient Greek Texts », *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, Vol. 1, n°2, 2010, p. 1–17.

Büchler M., Crane G., Moritz M., Babeu A., « Increasing Recall for Text Re-use in Historical Documents to Support Research in the Humanities », *In George Buchanan, Edie Rasmussen and Fernando Loizides: Theory and Practice of Digital Libraries 2012*. Lecture Notes in Computer Science, Volume 7489, pages 95-100, Springer Verlag. Paphos, Cyprus, 2012.

Büchler M., Crane G., Mueller M., Burns P., Heyer G., « One Step Closer To Paraphrase Detection On Historical Texts: About The Quality of Text Re-use Techniques and the Ability to Learn Paradigmatic Relations », *In George K. Thiruvathukal, Steven E. Jones, English: Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, Chicago, IL, USA, 2011.

Coulie, B, « La lemmatisation des textes grecs et byzantins : une approche particulière de la langue et des auteurs », *Byzantion : revue internationale des études byzantines*, Vol. 66, 1996, p. 35-54.

Dimitrios P. Lyras, Kyriakos N. Sgarbas, Nikolaos D. Fakotakis., « Applying similarity measures for automatic lemmatization: a case study for modern Greek and English? », *In Int. J. Artif. Intell. Tools* 17, 1043 (2008). DOI: 10.1142/S021821300800428X

Ernst-Gerlach A., Crane G., « Identifying quotations in reference works and primary materials », *Research and Advanced Technology for Digital Libraries*, vol. 5173, 2008, p. 78–87.

Faisal Alvi, El-Sayed M. El-Alfy, Wasfi G. Al-Khatib, and Radwan E. Abdel-Aal, « Analysis and extraction of sentence-level paraphrase sub-corpus in CS education », *Proceedings of the 2012 ACM SIGITE Conference, p. 49-54.*

Jo C., Pavel B., « New Functions for Unsupervised Asymmetrical Paraphrase Detection », *Journal of Software*, Vol. 2, n°4, 2007, p. 12–23.

Lee J., « A computational model of text reuse in ancient literary texts », *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007, p. 472–479.

Leskovec J., Backstrom L., Kleinberk J., « Meme-tracking and the dynamics of the news cycle », *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining,* p. 497-506.

Lukashenko R., Graudina V., Grundspenkis J., « Computer-Based Plagiarism Detection Methods and Tools : An Overview », *International Conference on Computer Systems and Technologies*, 2007, p. 1–6.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Computation and Language.Retrieved from http://arxiv.org/abs/1301.3781.

Mikolov, T., Sutskever, I., 0010, K. C., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and their Compositionality. In NIPS(pp. 3111–3119). Retrieved from http://dblp.uni-trier.de/db/conf/nips/nips2013.html#MikolovSCCD13

Hatem Mousselly-Sergieh, Elod Egyed-Zsigmond, Gabriele Gianini, Mario Döller, Harald Kosch & Jean-Marie Pinon (2013). « Tag Similarity in Folksonomies ». INFORSID 2013, 29 mai 2013, Paris (France), pp. 319-334. HAL : hal-01339168.

Nawab R., Stevenson M., Clough P., « Detecting Text Reuse with Modified and Weighted N-grams », *Proceedings of the ACM First Joint Conference on Lexical and Computational Semantics*, 2012, p. 54-58.

Roger T. Pédauque (Eds.). La redocumentarisation du monde.

7.2. Web pages

Archimedes Project Morphology Service, http://archimedes.mpiwg-berlin.mpg.de/arch/doc/ xml-rpc.html, accessed on 2013-02-07, updated in 2007.

Perseus Digital Library, http://www.perseus.tufts.edu, accessed on 2013-02-07, updated in 2013.

Thesaurus Linguae Graecae (TLG), www.tlg.uci.edu, accessed on 2013-02-07, updated in 2012.

Praxeme institute, www.praxeme.org/, accessed on 2013-10-04, updated in 2013.