University of Otago

Department of Computer Science

COSC480 Project Report

Quantifying Conceptual Density in Text

Author:
Anthony Dickson
(3348967)

Supervisor(s):
Anthony Robins
Alistair Knott

July 16, 2019

Abstract

It has been claimed that programming languages form a domain of *tightly integrated* concepts and that these tightly integrated concepts are difficult to describe or understand independently of other concepts. This tightly integrated nature of concepts in programming languages is thought to lead to the high rate of both fail and high grades observed in introductory programming courses across many institutions. This notion of the degree to which concepts are integrated in a domain is referred to as **conceptual density**. In this project we are interested in conceptual density in general, not just in the context of programming languages. In this report a definition of conceptual density and a graph-based method for quantifying conceptual density in well-structured expository text documents, such as educational textbooks, are explored.

1 Introduction

It is thought that the unusually high rate of fail and high grades observed in introductory programming courses is possibly due to concepts in programming languages being unusually tightly integrated (Robins, 2010). A tightly integrated concept is a concept that is difficult to describe independently of other concepts. Tightly integrated concepts are said to magnify the effects of learning edge momentum and lead to the polarised outcomes observed in introductory programming courses. This report is interested in conceptual density, an idea related to the degree to which concepts in a given domain are integrated.

The idea of conceptual density affecting learning difficulty is not unique to the aforementioned work. It appears in the literature on cognitive load theory, in a perhaps slightly different form, as the "element interactivity effect", which says that the number of interacting elements in a task is the main contributor to cognitive load - a measure of mental exertion required to perform a given task (Sweller, 1994). Cognitive load theory provides an explanation of aspects of learning difficulty and can be used to guide education and instructional design.

To the best of my knowledge there exists no objective measure in the current literature of the element interactivity effect, or the notion of conceptual density. Element interactivity appears to be estimated manually and it depends greatly on the knowledge of the learner (Chandler and Sweller, 1996), and as such remains subjective. In (Robins, 2010) the notion of concepts being tightly integrated stems from the observation that educators typically fail to reach a consensus on how to structure introductory programming courses (this is supposedly not the case in many other disciplines). While no measure of conceptual density was clearly defined in that work, some possible ways to do so were proposed.

In this report, I follow up on the notion of conceptual density and set out to elaborate on its definition and develop an objective measure of conceptual density in the context of well-structured expository text documents (e.g. textbooks, scientific papers). For measuring conceptual density I explore one of the methods mentioned in (Robins, 2010) - analysing a mind-map like graph structure. Whereas in the original work, this approach was suggested as a characterisation of introductory programming course material that is to be generated by experts, in this report I aim to build up and analyse mind-map like graph structures through the use of natural language processing methods and graph analysis techniques, removing the need for an expert to manually assess conceptual density for a given document.

The rest of this report is structured as follows: Section 2 covers the theoretical background of conceptual density; Section 3 defines what is considered as a concept in text and other text features that relate to conceptual density; Section 4 describes the parsing algorithm used for building a mind-map like graph structure and how a numerical score of conceptual density can be derived from a graph structure; Section 5 discusses related work and what distinguishes the work in this project from the related work; and Section 6 discusses the work-to-date and future work.

2 Background

2.1 Learning Edge Momentum

Learning edge momentum was proposed as an explanation for the unusually high rate of both fail and high grades in introductory computer science courses (Robins, 2010). The hypothesis builds upon the idea that we learn at the edges of what we know - it is easy to learn things that build upon pre-existing knowledge. The core of the hypothesis is that learning outcomes are self-reinforcing: successful learning tends to make learning closely related concepts easier, and unsuccessful learning tends to make learning closely related concepts more difficult. It is also said that concepts in programming languages tend to be tightly integrated and that these tightly integrated concepts are difficult to describe or understand independently of each other. This means that learning a new concept is greatly dependent on understanding the prerequisite concepts. And because of this the effects of learning edge momentum in introductory programming courses are particularly pronounced, leading to the polarised distribution of grades.

Conceptual density is mentioned in this paper as a possible metric for providing more concrete evidence to educators on how to structure an introductory programming course. Two methods are suggested for measuring the degree to which concepts in a domain are integrated. One method is to look at the proximity of key concept terms and see if there are areas where many concepts are mentioned close together. The other method is to generate a mind-map like graph structure A Here we represent the domain of concepts as a graph structure, where the concepts present in the material are represented with a node in the graph and edges between nodes represent a connection between concepts. This follows the main ideas behind semantic networks and knowledge graphs (Sowa, 1987; Zhang, 2002; Koncel-Kedziorski et al., 2019), an area of research concerning knowledge representation using graphs. This idea of a mind map of concepts forms the basis for the work in this project.

2.2 Cognitive Load Theory

Cognitive load theory provides a framework about learning difficulty and instructional design (Sweller, 1994). In CLT (cognitive load theory) the main task of the brain is characterised as information processing, and each person has a certain amount of working memory to process information. Performing tasks requires an amount of mental exertion and concentration, called cognitive load. When the amount of cognitive load exceeds the capacity of an individual's working memory, performing the given task becomes much more difficult.

If we are to explain this by analogy, the idea of working memory is similar in some ways to computer RAM and running out of working memory is akin to what happens when a computer runs out of RAM. When a computer runs out of RAM, it starts thrashing and all of the programs will become very slow and take many times longer to run compared to when there is enough RAM. By reducing the number of programs that are running we can reduce the memory usage and prevent the computer from thrashing. Similarly, under

```java
public class HelloWorld {
    public static void main(String[] args) {
        // Prints "Hello, World" to the terminal window.
        System.out.println("Hello, World");
    }
}
```

Figure 1: Java hello world example

CLT we can reduce cognitive load by reducing the number of things that must be considered simultaneously to perform a given task in order to make it easier to perform that task.

The main contributor to the difficulty of a task is number of things that must be considered simultaneously and the degree to which they interact. In CLT this idea is captured by the **element interactivity effect** (Sweller et al., 2011). The cognitive load imposed by element interactivity is said to be intrinsic to the material itself (**intrinsic cognitive load**), meaning that no matter how you present the material there will be a minimum level of difficulty.

How we present material can affect how difficult it is understand it. For example, consider trying to teach beginner programmers what all of the keywords mean in a Java program that simply prints a message to the screen (see Figure 1) and what each part of the code does. This would likely be too much to learn at once. Here there are many interacting elements (e.g. access modifiers, classes, types, functions, function arguments, comments, member fields, member functions, function calls, string literals) just in this simple program. Compare this to simply pointing out the line that prints the message and the fact that anything between the quotation marks is printed to the terminal window. This approach is much easier to understand since there are fewer things that must be considered. Changing how the material is presented has the effect of increasing or reducing **extraneous cognitive load**, a type of cognitive load that is not intrinsic to the material itself but dependent on how the material is presented.

Following on with the example, the details of the "Hello, world!" program could be taught more easily once the learner has learned the fundamentals and acquired a basic intuition for programming. It is theorised that this is because as complex ideas and tasks are learned, we learn more compact representation of these. These compact representations are often referred to as a **schema**; a schema is essentially a single unit of knowledge representing a concept or task (Axelrod, 1973; Abelson, 1981; Bartlett et al., 1995). By acquiring a schema, a task with many interacting elements can be reduced to a task with a few, or even a single, element. And under the element interactivity effect fewer interacting elements means less overall cognitive load. This idea of learned knowledge reducing the complexity of a task or idea is captured by **germane cognitive load**. Essentially, germane load is a type of cognitive load incurred by accessing schema. The amount of cognitive load incurred by this type of cognitive load is less than other types due to the use of schemas and their effect of reducing a complex idea/task down to a more compact representation.

To summarise CLT in a rather simplified manner, an individual has a limited amount of working memory to perform tasks and if the amount of cognitive load exceeds the capacity of this working memory, performing the given task becomes much more difficult. There is one major effect and three types of cognitive load that we are interested in: the element interactivity effect which is the main contributor to intrinsic load and overall cognitive load; extraneous load which is affected by the way in which the material/-task is presented; and germane load that is incurred when accessing schema (previously learned knowledge).

3 Conceptual Density in Text

To reiterate, conceptual density is an idea related to the degree to which concepts in a domain are integrated, and in this project we are interested in conceptual density in the context of text documents. Some of the questions that we are interested in are, for example: how are concepts related, to what degree are concepts integrated, do concepts form self-referential systems, how are concepts referenced and presented in the document? The core of conceptual density - the degree to which concepts are integrated - can be rooted in the element interactivity effect and described in terms of intrinsic cognitive load, and the way that concepts are presented in a document (e.g. how concepts are referenced between sections) can be described in terms of extraneous and germane cognitive load. In this section I will define what constitutes a concept in text in the context of this project and describe a few text features related to the idea of conceptual density.

3.1 Identifying Concepts in Text

To understand the relationships between concepts, we must first be able to identify concepts in text. A definition of a concept that could be used is a simple noun phrase - a phrase in text that denotes a thing. Using words from the previous sentence as an example, both "definition" and "simple definition" would be valid noun phrases, if we include determiners into the definition of a noun phrase "a simple definition" would also be a valid noun phrase. A more precise definition this type of noun phrase (also called a N-bar, or N′, phrase) is: an optional determiner, followed by a (possibly empty) sequence of nouns and/or adjectives, terminated by a noun. This kind of N-bar phrase can also be expressed in regex-like syntax using part of speech tags:

Grammar 1 N′: `<DT>?<NN.*|JJ>*<NN.*>`

where `DT` is a determiner (e.g. a, the), `NN.*` is any type of noun and `JJ` is an adjective.

The pattern described above describes simple noun phrases such as 'apple', 'an apple' and 'a red apple'. However, these simple noun phrases may also be joined together by prepositions (e.g. in, of) or coordinating conjunctions (e.g. 'Zeus is the *sky* ***and*** *thunder god* of the ancient Greek religion'). Here I extend the definition of a concept to be a N-bar phrase optionally followed by a sequence of preposition/coordinating conjunction and N-bar phrase pairs. We can define a noun phrase more precisely as:

Grammar 2 NP: `<N′>(<IN|CC><N′>)*`

where `N′` is Grammar 1, `IN` is a preposition and `CC` is a coordinating conjunction.

While 'things' as captured by these kinds of noun phrases make up many concepts, there are many concepts that not things but 'doing things', in other words many concepts are represented as verbs or verb phrases. For example, in programming we can talk about 'calling functions', 'looping' and 'instantiating an object' as concepts, and all of these are verbs or verb phrases. At this point in time, the only type of verb that I have included in the definition is the gerund verb, a type of verb that is typically suffixed with -ing, such as eat**ing**, sleep**ing**. This rather simple definition of a concept is in line with what is currently implemented. More rich definitions of a concept are to be explored in future work and are briefly discussed in Section 6.

3.2 Presentation of Concepts

The presentation of concepts is included in the definition of conceptual density. What I mean by presentation of concepts is the way that concepts are talked about in a document and how concepts are referenced between sections of the document. For example, a concept in section A may make reference to a concept in section B, likewise a concept in section C may reference a concept that was introduced in section B. These types of interactions between concepts are of interest, and are of relevance to the idea of conceptual density, since the interaction introduces another element that must be considered simultaneously by the reader. This increases element interactivity which is the core of conceptual density. In this section I will propose two types of concepts present in text, and four types of references that could be used to help measure the effect of presentation on conceptual density.

3.2.1 A Priori & Emerging Concepts

It is possible to categorise concepts into two types: a priori and emerging. An a priori concept is a concept that the reader of a text document would be expected to know beforehand or to be common knowledge. An emerging concept is a concept that is defined or introduced within a document. For example, a textbook on computer programming would expect the reader to know what a computer is, but possibly not the concept of recursion. In this case the concept ‘computer’ would be an a priori concept and ‘recursion’ would be an emerging concept. The importance of distinguishing between these two types of concepts is in how they contribute to cognitive load differently. Referencing an a priori concept would impose minimal cognitive load since the a priori concept would likely represent a fully acquired schema. On the other hand, an emerging concept would likely impose more cognitive load since it likely represents a schema that is yet to be fully acquired and still requires a noticeable amount of consideration in regards to the concept and its constituent parts.

```
Bread

Bread
Bread is a staple food typically prepared from a dough of
wheat flour and water, usually by baking.

Wheat
Wheat is a type of grain.
Wheat is commonly used for making wheat flour, a typical
ingredient of bread dough.
```

(a) A simple text document on the topic of bread.

```xml
<document>
    <title>
        Bread.
    </title>

    <section>
        <title>Bread</title>
        <text>
            Bread is a staple food typically prepared from a dough of wheat flour and water, usually by baking.
        </text>
    </section>

    <section>
        <title>Wheat</title>
        <text>
            Wheat is a type of grain.
            Wheat is commonly used for making wheat flour, a typical ingredient of bread dough.
        </text>
    </section>
</document>
```

(b) A version of the document in Figure 2a that has been marked up in XML.

Figure 2: A sample document with a plain text version (Figure 2a) and a version marked up with XML (Figure 2b).

3.2.2 Forward References

Forward references are where the text makes reference to a concept that is not fully explained until later in the document. These types of references introduce extraneous cognitive load since they make the reader *park* the involved concepts, without much existing knowledge to associate them with. Remembering unfamiliar terms that carry little meaning (to the reader) is more of a demanding task than remembering well integrated knowledge. For example, when teaching Java programming to beginners the meaning and function of keywords such as `class` and `static` are often not explained and are usually just rote learned until later in the course.

3.2.3 Backward References

Backward references are where the text makes reference to a concept that was explained previously in the text. These types of references are likely to be introducing a connection between concepts. The cost of backward references are relatively low when compared to forward references. With forward references we are making the readers consider an additional element that is not at all well understood by the reader, along with the current context. However, in the case of backward references we are simply asking the reader to recall a previously explained concept and possibly introduce a new relationship between the two concepts. This type of reference imposes germane load as it requires the reader to call on pre-existing knowledge.

4 Building a Graph of Concepts

The domain of concepts present in a text document can be represented with a graph structure. The graph structure that is being built is similar to a mind-map, except instead of a single central idea/concept, we are building a rather free-form graph with concepts branching off other related concepts. Representing a domain of concepts in this type of graph has precedent in semantic networks/knowledge graphs. This graph-based approach is a natural approach for the problem of quantifying conceptual density because the core concern is the degree to which concepts are integrated, in other words the relationships between the concepts. With respect to the quantification of conceptual density, well-established graph theory and related analysis techniques provide ways to derive a numerical score of conceptual density.

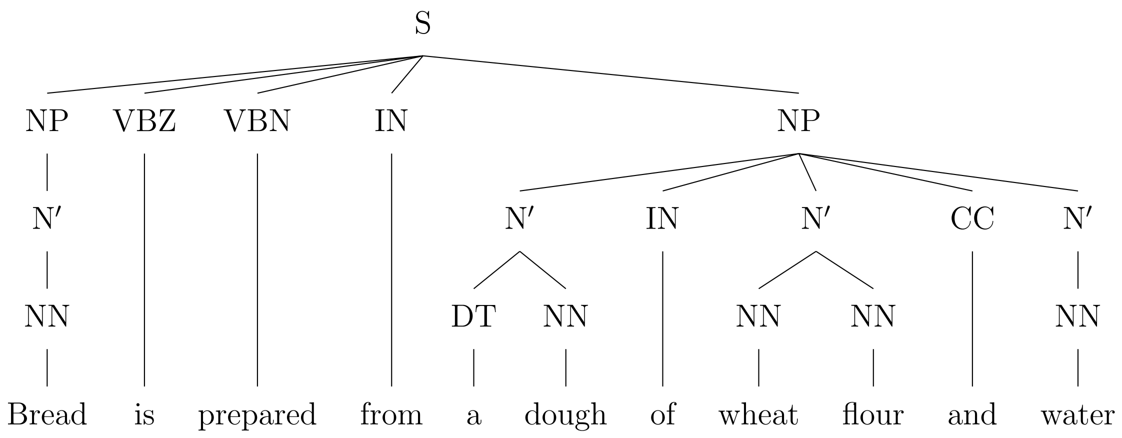

Figure 3: The parse tree resulting from tagging and chunking the sentence "Bread is prepared from a dough of wheat flour and water" using the regex grammars defined in Grammar 1 and Grammar 2. The nodes directly above the terminal nodes denote each word's part of speech tag.

4.1 Nodes & Extracting Concepts

In the graph structure we are building, a node represents a concept present in a given text document. As discussed in Section 3.1, we create a definition of what constitutes a concept in terms of patterns of parts of speech. The formal definition of the pattern given defines what is called a grammar. The basic process to extract a concept from text is to segment the document into sentences, then split each sentence into a set of tokens (words separated by white-space), then for each set of tokens we assign a tag to each token denoting the part of speech of the token. From here we chunk the sentences into groupings based on the patterns described in the grammar. An example is shown in Figure 3.

From the example in Figure 3 one might observe that noun phrases can be complex and made up of multiple N-bar phrases. For example, from the phrase "a dough of wheat flour and water" we could extract the N-bar phrases 'a dough', 'wheat flour' and 'water'. And even from the N-bar phrase 'wheat flour' we can extract two distinct nouns, 'wheat' and 'flour'. I choose to add these constituent parts to the graph as nodes in order to produce a more rich representation of the domain of concepts present in the text document.

[1] Figure generated from `https://corenlp.run/`.

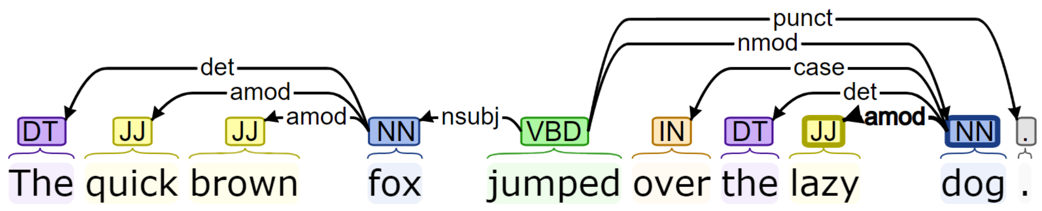

Figure 4: An example of a dependency parse. The arc labelled "nsubj" points to the subject of the sentence. For a description of what the relations on the arcs mean refer to (Martin and Jurafsky, 2018).[1]

4.2 Edges & Relating Concepts

In the graph structure we are building, a relation between concepts is represented as a directed edge between nodes. Edges are used to represent a type of relation where one concept makes reference to, or depends on, another concept. There are three criteria for which I choose to create an edge between two concepts: one concept is the subject of a sentence and the other concept appears in the same sentence; one concept is a noun phrase and the other is a N-bar phrase that is part of the noun phrase; or one concept is a N-bar phrase and the other is a noun that is part of that N-bar phrase. The subject of a sentence can be identified through a dependency parse, which marks the relationships between words in a sentence. See Figure 4 for an example.

4.3 Identifying A Priori & Emerging Concepts

A priori and emerging concepts can be identified in a graph by looking at the incoming edges. For a given node, we can look at each of the edges associated with that node. If all of the incoming edges come from nodes in the same section as the node in question, then we can say that the node represents an a priori concept. Otherwise, if there are two or more sections that the incoming edges originate from, then we can say that the node in question represents an emerging concept.

4.4 Identifying Forward and Backward References

In the graph forward and backward references can be found through traversing the graph structure and identifying when the algorithm crosses between

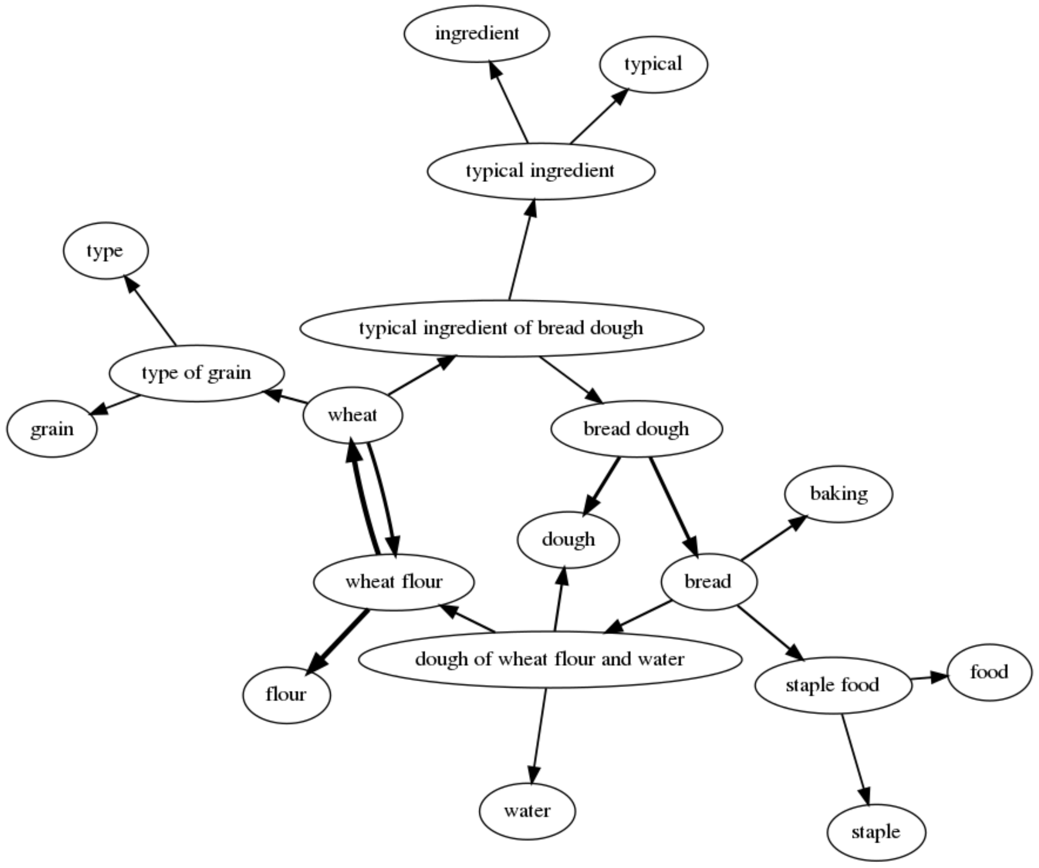

Figure 5: An example of a dependency-based graph created using the XML document from Figure 2b. Nodes represent the concepts that are referenced in the document. Edges in the graph represent the relationship 'A references B'. Blue edges represent forward references and red edges represent backward references.

sections (see Algorithm 1). A prerequisite is that we must record the sections that appear in the text, and the order in which they are introduced during the parsing of the text document. We can identify a forward reference when the traversal algorithm visits a node that is associated with a section that comes after the section of the previously visited node. Similarly, backward references can be identified when the traversal algorithm visits a node that is associated with a section that comes before the section of the previously visited node. An example of a graph where the forward and backward egdes have been coloured is shown in Figure 6.

4.5 Deriving a Numerical Score

At the centre of conceptual density is the notion of the degree to which concepts are integrated. In the graph structure this can be measured by looking at the connectivity of nodes. One useful measure may be the average outdegree:

$$S_D = \frac{1}{|V|} \sum_{v \in V} \deg^+(v) \tag{1}$$

where S_D is the score of the document D, V is the set of vertices in the graph, and $\deg^+(v)$ is the outdegree, or the number of outgoing edges from the vertex v. The interpretation of a high average degree is that the average concept is relatively difficult to learn since the concept is related to many other concepts which you must understand to understand the given concept.

We can further refine the scoring method by taking into account the strength of the edges, or in other words the weights of the edges:

$$S_D = \frac{1}{|V|} \sum_{v \in V} \sum_{e \in E[v]} e_w \tag{2}$$

where $E[v]$ is the set of edges originating from the vertex v, and e_w is the weight for a given edge. The weight for given edge can be adjusted based on the type of reference the edge represents. For example, we could set the weight of an edge to: 0.5 for edges that point to a priori concepts since they typically represent a concept that the reader is expected to know beforehand and only impose germane load; 1.5 for backward references to emerging concepts since the reader is required to recall a concept that has been introduced within the text and using a poorly integrated schema will require more mental exertion than when considering an a priori concept; and 2.0 for forward

Algorithm 1: Marking Forward and Backward References

```
Data: sections: List of sections in a given document $D$
nodes: Set of nodes in the graph $G$
adj: Adjacency list for a directed graph $G$.

visited = $\emptyset$

for $node \in nodes$ do
    markEdges(node, NULL, visited)
end

Function markEdges(curr, prev, visited)
    /* Arguments                                    */
    /* curr:  Current node                          */
    /* prev:  Previous node                         */
    /* visited:  Set of visited nodes               */

    if $prev \neq NULL\ AND\ curr.section \neq prev.section$ then
        $curr\_i \leftarrow sections.indexOf(curr.section)$
        $prev\_i \leftarrow sections.indexOf(prev.section)$

        if $curr\_i < prev\_i$ then
            Mark edge $\{prev, curr\}$ as a backward reference
        else if $curr\_i > prev\_i$ then
            Mark edge $\{prev, curr\}$ as a forward reference

    if $curr \notin visited$ then
        $visited \leftarrow visited \cup \{curr\}$

        for $neighbour \in adj[curr]$ do
            markEdges(neighbour, curr, visited)
        end
end
```

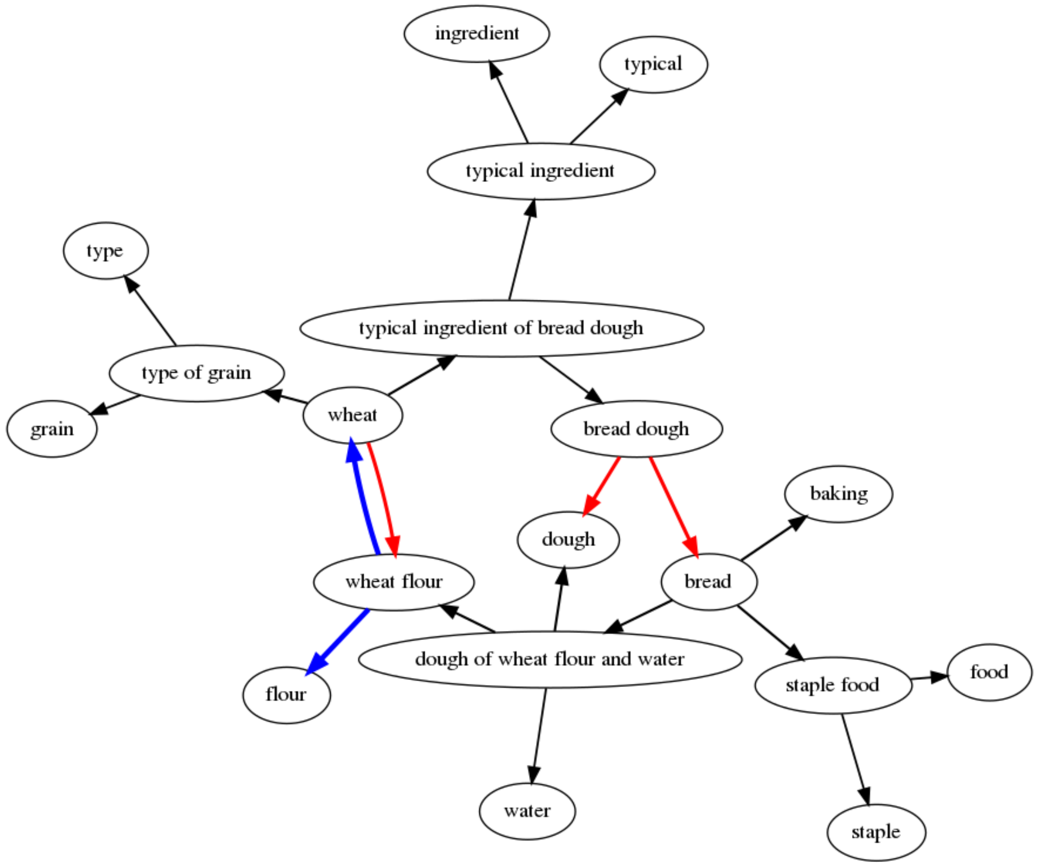

Figure 6: An example of a dependency-based graph created using the XML document from Figure 2b where forward and backward references have been coloured. Nodes represent the concepts that are referenced in the document. Edges in the graph represent the relationship 'A references B'. Blue edges represent forward references and red edges represent backward references.

references since the reader is required to *park* a new concept without anything concrete to pin it too, increasing cognitive load. The exact numbers are not important, but rather the relative scales of the numbers. References to a priori concepts should impose less cognitive load than backward references, and backward references should impose less cognitive load than forward references.

4.6 Summary

By using part of speech tags and a regex-based chunker we can extract concepts from text. We can then derive relationships between concepts by looking at dependency parses of sentence structures and by looking at the constituent parts of complex concepts represented by noun phrases. For the former method, we simply take the subject of a sentence and create edges between it and each of the other concepts that appear in the same sentence. Concepts are represented as a node in the graph and relationships between concepts are represented as edges in the graph. The different types of concepts and references can be identified by analysing the edges in the graph. Finally, we can derive a score from the generated graph by measuring the average weighted outdegree.

5 Related Work

5.1 Information Extraction

The process of identifying concepts and how they are related that I have described in this report is similar to the task of information extraction. However, there are some key differences between what I have done and what is usually done in information extraction. First and foremost, information extraction is typically concerned with named-entity recognition - the identification of named-entities such as names of people, companies, locations, and dates. In this project we are concerned with concepts in general (all types of entities), which can be thought of as a superset of named-entities. Furthermore, information extraction is also interested in the relationship between named-entities. However, in this project and the work-to-date we are not too concerned about the nature of a relationship between concepts as much as the fact that a relationship exists.

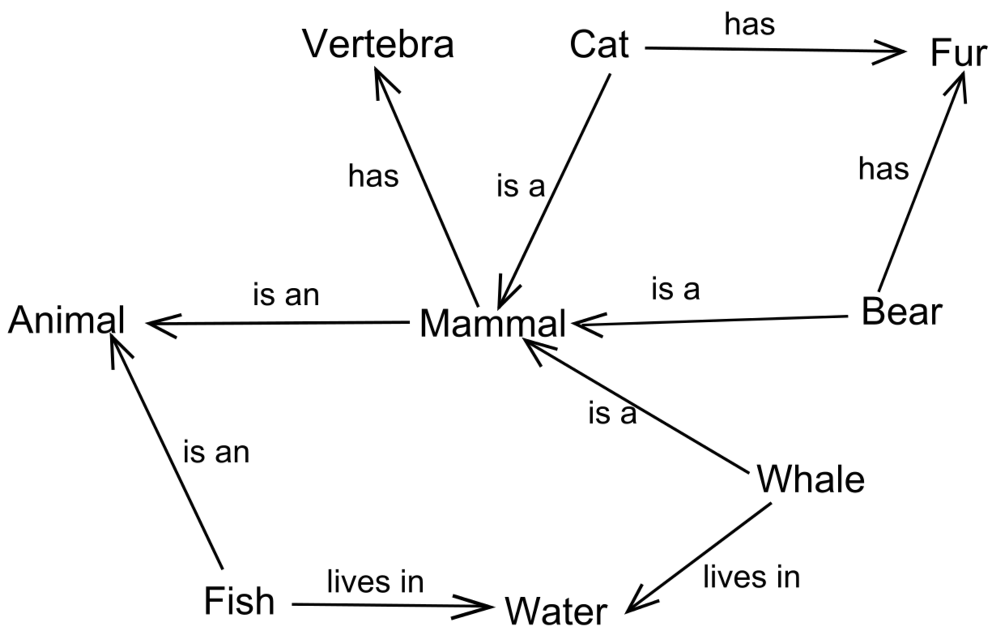

Figure 7: An example of a semantic network.[2]

5.2 Semantic Networks & Knowledge Graphs

The type of graph structure that is being built up in this project is very similar to semantic networks/knowledge graphs. Some differences between the graphs in this project and semantic networks are that: we are not annotating edges with the type of relationship (compare Figure 6 and Figure 7, for example), and the criteria for deciding what constitutes a node or edge in the graph may differ from what is common in the literature. For one, somewhat complex noun phrases are added as nodes in this project rather than just the head of a noun phrase.

[2]Figure from https://commons.wikimedia.org/wiki/File:Semantic_Net.svg.

6 Future Work

6.1 Transitioning to Unstructured Text

In the current implementation a graph can be built from semi-structured text (e.g. Figure 5), but recreating the same graph from the corresponding plain text version of a document remains difficult. At present the sections and section headers must be explicitly marked up in XML. Moving to unstructured text is desirable since using semi-structured text requires someone to manually mark up the text, which is a time consuming process. Identifying the section titles and where sections start and end can be unreliable at times when using regex pattern matching due to different section heading formats and it is liable to false positives (see Table 1 in the appendix, for example).

6.2 Improving Information Extraction

One issue with the current work is the quality of the extracted concepts and how the concepts are related. There are many cases where the extracted concepts do not align with the concepts or relations that a human would intuitively pick out. While the current quality is adequate enough to build up a graph structure, important concepts and relations between concepts are likely to be missed. So it is important that future work focuses on improving the quality of extracted concepts and relations. Improving the current implementation would include extending the definition of a concept in text to include verbs and verb phrases, and another thing to improve would be the quality of the extracted relations between concepts.

However, there exists a framework called OpenIE (Banko et al., 2007) which is made for performing information extraction, which is essentially the task that I am trying to perform. The benefits of using such a framework is that many of the issues with the current implementation for extracting concepts and relations between concepts in this project have more or less been solved in OpenIE. One issue with using OpenIE in this project is that it generates many spurious tuples (see Table 2 in the appendix for an illustrative example). So if OpenIE were to be used in this project the main thing that would need to be worked on is filtering the triples.

6.3 Additional Graph Features

There are many graph analysis techniques that have not been explored in this report. For example, see (NetworkX Developers, 2004). It would be interesting to see if incorporating features derived from these analysis techniques into the scoring scheme would improve it. For example, we could look at: identifying simple cycles in the graph using Johnson's algorithm; identifying minimum cuts which could be used to identify bottleneck concepts and to find subgraphs that are *isolated* in the sense that they are mostly separate from the main supergraph; and reachability of nodes in the graph.

6.4 Additional Text Features

There are other features that could be extracted from text. In (Robins, 2010) it is suggested that a measure of distance between keywords (concepts) could be used to help measure conceptual density and included in the scoring scheme. There are some methods that may help with this such as: using lexical chains for keyword extraction (Ercan and Cicekli, 2007); and looking at long distance dependency as a metric of reading comprehension difficulty (Liu, 2008).

6.5 Evaluation of the Model

Regarding the evaluation of the model for quantifying conceptual density, the quality of the scores have not been evaluated. Since no objective measure of conceptual density exists in the current literature, there exists no benchmarks to compare the model with. The next best thing to compare the model against is human judgements. One experiment that could be done is to gather a collection of snippets from textbooks. These snippets would consist of a few chapters from a given textbook. Then we could arrange the collection into pairs and get human judgements on whether or not the first snippet is more conceptually dense than the second snippet. From there, for each document pair we could generate scores using the model described in this report and determine which document the model judges as being more conceptually dense. Then we could compare the model's judgements to the human judgements to evaluate how 'good' the model is, or how well it follows human intuition.

7 Conclusion

In this report I have explored a graph-based approach to quantifying conceptual density - an idea relating to the degree that concepts of a particular knowledge domain, encoded by a text document, are integrated. I have discussed how this idea of conceptual density is grounded in, and related to, learning edge momentum and cognitive load theory. I have discussed the ways in which the notion of conceptual density can be defined and measured in semi-structured and unstructured text documents. By analysing the language in the documents and observing the patterns in natural language we can create an operational definition of a 'concept' in text. Then by analysing the relationships between concepts in a given document we can build up a graph structure from which we can derive a score of conceptual density.

References

Robert P Abelson. Psychological status of the script concept. *American psychologist*, 36(7):715, 1981.

Robert Axelrod. Schema theory: An information processing model of perception and cognition. *American political science review*, 67(4):1248–1266, 1973.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

Frederic Charles Bartlett, Frederic C Bartlett, and Walter Kintsch. Remembering: A study in experimental and social psychology, 1995.

Paul Chandler and John Sweller. Cognitive load while learning to use a computer program. *Applied cognitive psychology*, 10(2):151–170, 1996.

Gonenc Ercan and Ilyas Cicekli. Using lexical chains for keyword extraction. *Information Processing & Management*, 43(6):1705–1714, 2007.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. Text generation from knowledge graphs with graph transformers. *arXiv preprint arXiv:1904.02342*, 2019.

Haitao Liu. Dependency distance as a metric of language comprehension difficulty. *Journal of Cognitive Science*, 9(2):159–191, 2008.

James H Martin and Daniel Jurafsky. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Third Edition draft.* Pearson/Prentice Hall Upper Saddle River, 2018.

NetworkX Developers. Algorithms - networkx 2.3 documentation. https://networkx.github.io/documentation/stable/reference/algorithms/index.html, 2004. Accessed on 05 July 2019.

Anthony Robins. Learning edge momentum: A new account of outcomes in cs1. *Computer Science Education*, 20(1):37–71, 2010.

John F Sowa. Semantic networks. 1987.

John Sweller. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, 4(4):295–312, 1994.

John Sweller, Paul Ayres, and Slava Kalyuga. The element interactivity effect. In *Cognitive Load Theory*, pages 193–201. Springer, 2011.

Liecai Zhang. *Knowledge graph theory and structural parsing.* Twente University Press, 2002.

A Tables

Table 1: An example of the current implementation for finding section headers, start and end positions generated from (Sweller et al., 2011).

Section Header	**Start Pos.**	**End Pos.**
Chapter 15 The Element Interactivity Effect	44	3358
Empirical Evidence for the Element Interactivity Effect	3359	8850
**The secondary task was presented on a separate computer and involved a tone*	8851	9091

Section Header	**Start Pos.**	**End Pos.**
* *When instructional materials involved low or no interaction between elements* of	9092	9964
Element Interactivity and Understanding Instructions	9965	10041
* *The extent to which we understand instructions depends on levels of element*	10042	12129
Element Interactivity and the Modality Effect	12130	16477
Element Interactivity and the Expertise Reversal Effect	16478	18835
Element Interactivity and the Imagination Effect	18836	21474
Conditions of Applicability	21475	21908
* *There is also an obvious relation between these two effects as the levels of element interactivity that produce an intrinsic cognitive load are always relative to levels of*	21909	24226
Instructional Implications	24227	26005
Conclusion	26006	27782

Table 2: An example of the relation triples extracted from the document in Figure 2a illustrating how OpenIE generates many spurious triples. Notice that some entities in the column 'Entity 2' are nonsensical, e.g. "staple food by usually baking".

Entity 1	**Relation**	**Entity 2**
Bread	is	food typically prepared from dough by baking
Bread	is	staple food prepared
Bread	is	staple food typically prepared from dough of wheat flour usually by baking
Bread	is	food typically prepared from dough
Bread	is	food by baking
Bread	is	staple food typically prepared from dough usually by baking
Bread	is	staple food prepared from dough by baking

Entity 1	Relation	Entity 2
Bread	is	food typically prepared usually by baking
Bread	is	food prepared
Bread	is	food typically prepared
Bread	is	staple
Bread	is	food prepared from dough of wheat flour by baking
Bread	is	staple food typically prepared
Bread	is	staple food by baking
Bread	is	food prepared from dough by baking
Bread	is	staple food typically prepared by baking
Bread	is	staple food prepared from dough of wheat flour by baking
Bread	is	food typically prepared from dough of wheat flour
Bread	is	staple food
Bread	is	food typically prepared by baking
Bread	is	staple food typically prepared from dough of wheat flour by baking
Bread	is	food typically prepared from dough of wheat flour by baking
Bread	is	staple food typically prepared from dough
Bread	is	food prepared from dough
Bread	is	food prepared from dough of wheat flour
Bread	is	staple food typically prepared usually by baking
Bread	is	staple food prepared from dough usually by baking
Bread	is	staple food prepared from dough of wheat flour usually by baking
Bread	is	food by usually baking
Bread	is	food prepared usually by baking
Bread	is	food prepared by baking
Bread	is	staple food by usually baking
Bread	is	food typically prepared from dough of wheat flour usually by baking
Bread	is	food prepared from dough usually by baking

Entity 1	**Relation**	**Entity 2**
Bread	is	staple food prepared usually by baking
Bread	is	staple food prepared by baking
Bread	is	staple food typically prepared from dough by baking
Bread	is	staple food typically prepared from dough of wheat flour
Bread	is	staple food prepared from dough of wheat flour
Bread	is	food
Bread	is	food typically prepared from dough usually by baking
Bread	is	food prepared from dough of wheat flour usually by baking
Bread	is	staple food prepared from dough
Wheat	making	ingredient of bread dough
Wheat	making	wheat flour
Wheat	making	ingredient
Wheat	is	commonly used
wheat flour	ingredient of	bread dough
Wheat	making	typical ingredient of bread dough
Wheat	making	typical ingredient
Wheat	is	used