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

October 7, 2019

Contents

Abstract

"Conceptual density" is a term that describes the degree to which concepts in a domain are integrated, or interdependent. There is a hypothesis that text documents with high conceptual density are harder to process. At present, the concept of "conceptual density" has been informally defined. In this pilot study, I investigate the idea of conceptual density in the context of expository text documents, a graph-based model for quantifying conceptual density and ways to evaluate this model. Finally, I discuss open problems and direction for future work related to conceptual density.

1 Introduction

Robins (2010) suggested that a domain of "tightly integrated concepts" magnifies the effect of early success or failure of learning, leading to polarised learning outcomes, an effect that he named "learning edge momentum". A tightly integrated concept is said to be a concept that is difficult to describe or explain independently of other concepts. This pilot study is focused on the idea of "conceptual density", an idea related to the degree to which concepts in a given domain are integrated.

The idea of conceptual density affecting the difficulty of learning is not unique to the aforementioned work. Cognitive load theory provides a framework for reasoning about aspects of learning difficulty and can be used to guide instructional design (Sweller, 1994). Conceptual density appears in the literature on cognitive load theory, in a perhaps slightly different form, as "element interactivity". Element interactivity, similar to conceptual density, is an idea related to the number of interacting elements in a task or piece of material that is to be learned. Under the element interactivity effect, high element interactivity is suggested to be the main contributor to the difficulty of a learning task.

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 pilot study, I set out to elaborate on the definition of conceptual density and develop an objective measure of this concept in the context of well-structured expository text documents (e.g. textbooks). 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 pilot study I aim to create a tool that builds up and analyses mind-map like graph structures from a text document automatically, through the use of natural language processing methods and graph analysis techniques, thus removing the need for an expert.

The main contributions of the work in this pilot study are: (i) the exploration of the notion of conceptual density (Sections 2 and 3); (ii) a graph-based model for quantifying conceptual density (Sections 3, 4 and 5); (iii) a framework for quantitatively evaluating the quality of the model (Section 6); and (iv) a discussion of open problems and direction for future work related to conceptual density (Section 8).

2 Background

2.1 Learning Edge Momentum

Robins (2010) proposed learning edge momentum as an explanation for the unusually high rate of both fail and high grades in introductory programming courses. The hypothesis builds upon the idea that we learn at the edges of what we know, or in other words, 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 of a concept tends to make learning closely related concepts easier, and unsuccessful learning tends to make learning closely related concepts more difficult. It is also suggested 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 introductory programming courses. 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. In this approach, a domain of concepts is represented 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), which are graph structures used for knowledge representation. This idea of a mind map of concepts forms the basis for the work in this pilot study.

2.2 Cognitive Load Theory and Schema

CLT (cognitive load theory) provides a framework for reasoning about learning difficulty and instructional design (Sweller, 1994). In CLT 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 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.

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

Figure 1: Example Java program that prints the message “Hello, World”.

The main contributor to the difficulty of a task is the number of things that must be considered simultaneously and the degree to which they interact. In CLT this idea is captured by **element interactivity** and 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 to understand it. For example, consider trying to teach beginner programmers what all the keywords mean in a Java program that prints “Hello, World” 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 the code of this simple program. Compare this to 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 representations of these that are committed to long-term memory. These compact representations are often referred to as **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 elements, or even a single element. And under the element interactivity effect, fewer interacting elements means less overall cognitive load.

To summarise CLT in a rather simplified manner, an individual has a limited amount of working memory to process information and if the amount of cognitive load exceeds the capacity of this working memory, then processing the information becomes much more difficult. There are three things from CLT that we are interested in: element interactivity, which is the main contributor to both intrinsic cognitive load and overall cognitive load; intrinsic cognitive load which is related to the inherent complexity of information; and extraneous load which is influenced by how the material/task is presented.

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 pilot study, we are interested in conceptual density in the context of text documents. Some 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? Conceptual density can be rooted in element interactivity 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 cognitive load. In this section, I will give a simple working definition of a concept in text 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 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 is: an optional determiner, followed by a (possibly empty) sequence of nouns and/or adjectives, terminated by a noun. This kind of noun phrase can also be expressed in regex-like syntax using part of speech tags:

Grammar 1 NP: `<DT>?<NN.*|JJ>*<NN.*>`

where the angle brackets denote a group/part of speech tag, `DT` is a determiner (e.g. a, the), `NN.*` is any type of noun and `JJ` is an adjective.

This pattern describes noun phrases such as 'apple', 'an apple' and 'a red apple'. However, these noun phrases may also be joined together by prepositions (e.g. 'in' or 'of') or coordinating conjunctions (e.g. 'and' or 'or') to form a single, complex concept. For example, consider the sentence "Zeus is the sky and thunder god of the ancient Greek religion". We can see that 'the sky and thunder god' is one concept because we are not saying that Zeus is 'the sky' and that he is separately the 'thunder god', so we have a case for considering two noun phrases joined by a coordinating conjunction as a single concept. It is also evident that we should consider 'the sky and thunder god of the ancient Greek religion' as a single, complex concept because Zeus is not the sky and thunder god of all religions, but rather of the ancient Greek religion alone. I call these types of noun phrases 'complex noun phrases'. For this pilot study, I define a concept to be a complex noun phrase, which is a noun phrase optionally followed by a sequence of preposition/coordinating conjunction and noun phrase pairs. We can define a complex noun phrase more precisely as:

Grammar 2 CNP: `<NP>(<IN|CC><NP>)*`

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

3.2 Presentation of Concepts

The 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 require a reference to a concept in section B; likewise a concept in section C may require a reference to a concept 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 cognitive load. In this section, I will propose two types of concepts present in text, and two types of references that could be used to help measure the effect of presentation on conceptual density.

3.2.1 A Priori and 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 and well-integrated knowledge. 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 mental exertion in regards to the concept and its constituent parts.

3.2.2 Forward References

Forward references are where the text refers 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 refers to a concept that was explained previously in the text. These types of references are likely to be introducing a relation between concepts. The cost of backward references is 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 asking the reader to recall a previously explained concept and possibly introduce a new relationship between the two concepts. This type of reference imposes less cognitive load as it allows 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. In this section, I will describe how we can build a graph of concepts from text, and how we can extract concepts and the text features described in the previous section.

4.1 Nodes and 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 grammars. An example is shown in Figure 2.

By definition, a complex noun phrase is made up of multiple noun phrases (Grammar 2). For example, from the phrase “a dough of wheat flour and water” we could extract the noun phrases ‘a dough’, ‘wheat flour’ and ‘water’. And even from the noun phrase ‘wheat flour’ we can extract two distinct nouns, ‘wheat’ and ‘flour’. I choose to add these constituent parts to the graph as nodes to produce a richer representation of the domain of concepts

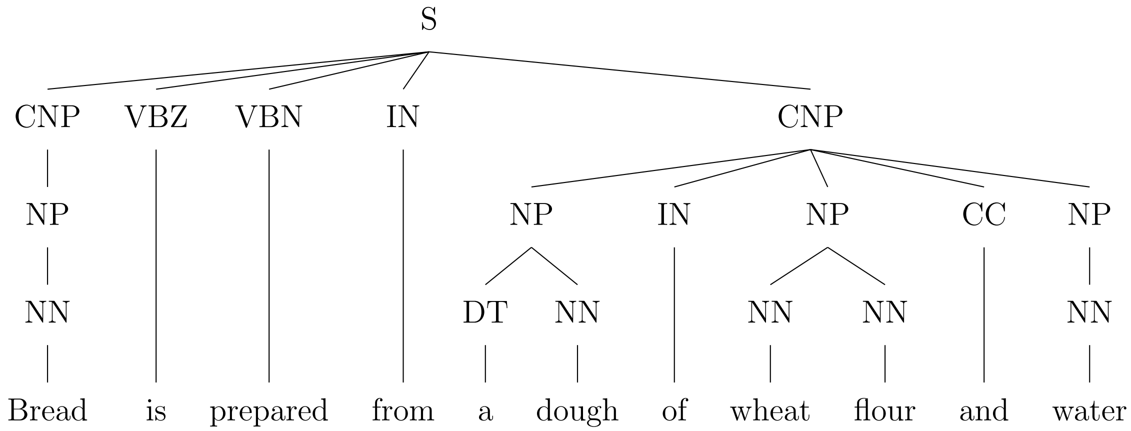


Figure 2: The parse tree resulting from first tagging the parts of speech for tokens in the sentence “Bread is prepared from a dough of wheat flour and water”, and then chunking the tagged tokens using the regex grammars defined in Grammar 1 and Grammar 2. The nodes directly above the terminal nodes denote each word’s part of speech tag. The tag ‘S’ represents a sentence, the tag ‘VBZ’ represents a verb in 3rd person singular present form and ‘VBN’ represents a verb in past participle form.

present in the text document. This means that adjectives are also considered to be concepts. Adding these constituent parts also helps match up concepts that are closely related but perhaps not mentioned together or mentioned in the same form, e.g. dough’ and ‘bread dough’ in Figure 3a.

4.2 Edges and 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 refers 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 complex noun phrase (Grammar 2) and the other is a noun phrase (Grammar 1) that is part of the complex noun phrase; or one concept is a noun phrase and the other is a noun or an adjective that is part of that noun phrase. The subject of a sentence can be identified through a dependency parse, which marks the relationships

```
Bread

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

Wheat Flour
Wheat Flour is made from ground up 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.

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

    <section>
        <title>Wheat Flour</title>
        <text>
            Wheat flour is made from ground up wheat.
            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 3a that has been marked up in XML.

Figure 3: A sample document with a plain text version (Figure 3a) and a version marked up with XML (Figure 3b).

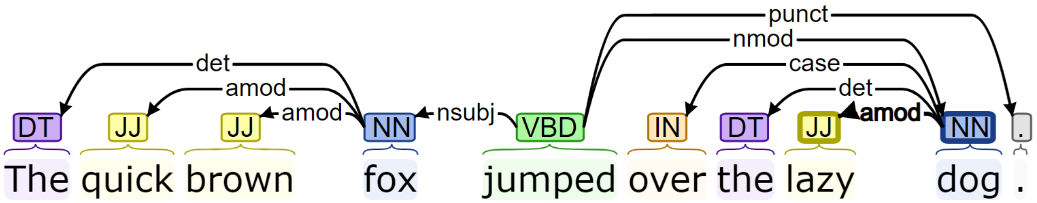


Figure 4: An example of a dependency parse.[1]The arc labelled "nsubj" points to the subject of the sentence. For a description of what the relations on the arcs mean, refer to (Martin and Jurafsky, 2018).

between words in a sentence (see Figure 4 for an example).

4.3 Identifying A Priori and Emerging Concepts

To identify a priori and emerging concepts we can look at how the concepts are mentioned. In particular, we could look for sentences that contain definitions. One type of sentence that could be considered to contain a definition could be one that follows the pattern 'X is a Y'. If a concept appears as the subject of these kinds of sentences then perhaps that could indicate that the concept is an emerging concept. For example, if the first mention of 'SGX' is "Intel Software Guard Extensions (SGX) is a set of security-related instruction codes that are built into some modern Intel central processing units (CPUs)" then perhaps it is likely that SGX is being defined here and thus is an emerging concept. We can look for other patterns in the structure of sentences that suggest that a concept is being defined, such 'X is defined as a Y'. The exact patterns that are used are described later in Section 5.3.2. A priori concepts are then defined as any concept that does not fulfil the definition of 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 traversal algorithm crosses between document sections. A prerequisite is that we must record the sections that appear in the text, the order in which they are introduced during the parsing of the text document, and associate each node in the graph with

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

a section. Each node is associated with the section that the node's corresponding concept appears in most frequently. In the case of a tie, the section that appears first is chosen. We can then 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. References are restricted to only point to nodes that have been labelled as emerging concepts since the definition of forward and backward references precludes references to a priori concepts. Pseudocode for this process is given in Algorithm 1.

4.5 Deriving a Numerical Score

At the centre of conceptual density is the notion of the degree to which concepts are integrated. In a graph structure, this could 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 a 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 e. The weight for a 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: 1.0 for edges that point to a priori concepts since they typically represent a concept that the reader is expected to know beforehand and are easier to process; 1.5 for backward references to emerging concepts since the reader is required to recall a concept that has been introduced

Algorithm 1: Marking Forward and Backward References

```
Data: $sections$: List of sections in a given document $D$
$nodes$: Set of nodes in the graph $G$
$nodes_{\text{emerging}}$: Set of nodes in $G$ labelled as emerging concepts
$adj$: 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:  The previously visited node                     */
    /* visited:  Set of visited nodes                         */

    if $prev \neq NULL\ AND\ curr.section \neq prev.section\ AND\ curr \in nodes_{emerging}$ then
        /* Find the ordering of the sections associated with
           the nodes $curr$ and $prev$.                       */
        $curr\_i \leftarrow sections.indexOf(curr.section)$
        $prev\_i \leftarrow sections.indexOf(prev.section)$

        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
```

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 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 scale of these 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.

It should be noted that the scores generated by this model do not have much meaning on their own. This is because it is difficult to come up with an interpretation of what a score of 3.1, for example, means in terms of conceptual density. As such, the main purpose of these scores is for comparison and the scores should be interpreted as an ordinal scale.

4.6 Summary

By using part of speech tagging 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 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. Figure 5 shows the graph generated using the current implementation and the sample text document in Figure 3.

5 Implementation of the Conceptual Density Model

In this section, I will describe the implementation of the model discussed across Sections 3 and 4 that is used in the experiments. The current implementation takes in a XML formatted document, parses the document and extracts concepts and relations between concepts, builds up a graph representation of these extracted entities and generates a score. The way the score

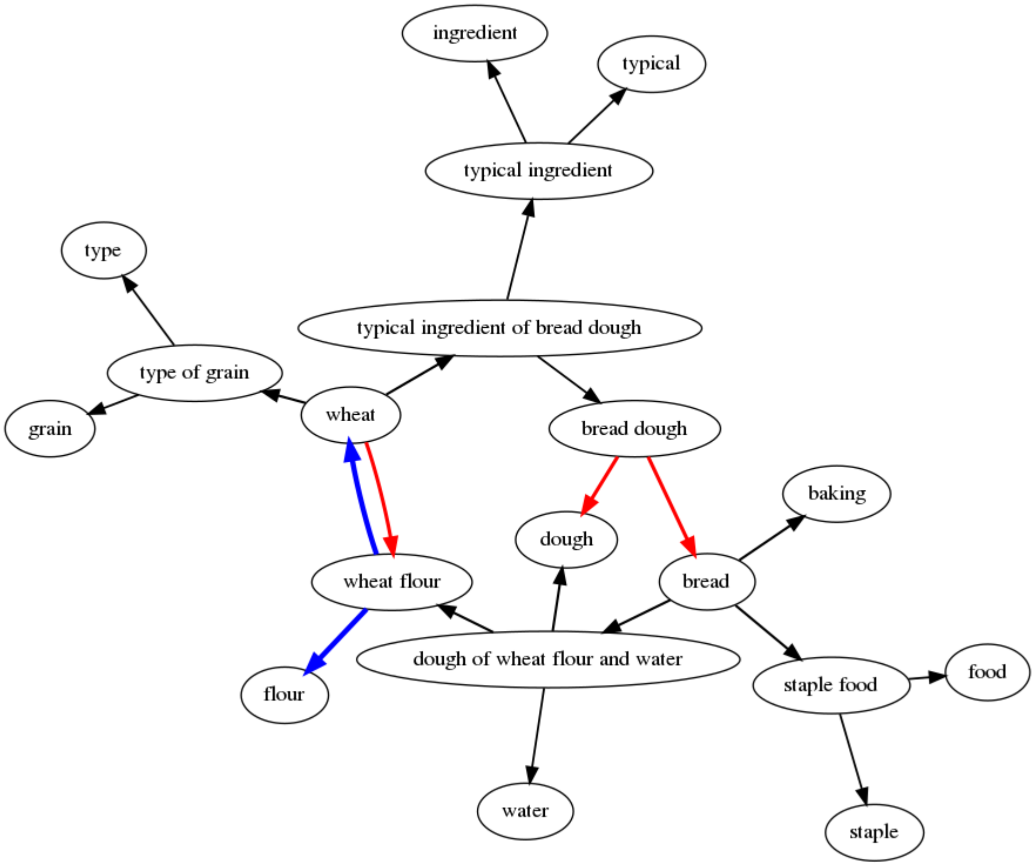


Figure 5: An example showing the graph generated from the XML document in Figure 3b where forward and backward references have been coloured. Nodes represent the concepts that are referenced in the document. Edges in the graph represent a relationship between concepts. Blue edges represent forward references and red edges represent backward references.

is generated is the same as it was described in Section 4.5, so I will only describe the other aspects below. Also, since the motivation and rationale for the different aspects of the model are explained in the previous sections, I will focus on describing the steps involved in processing a document in more detail. The source code for the model is made publicly available Dickson (2019b).

5.1 Input Format

One limitation of the current implementation is that documents must be marked up in XML. The reason why this semi-structured text format was chosen is that identifying sections and where start and end in plain text is not always reliable (this is discussed briefly in Section 8, Future Work) and it is of interest to see how the model performs assuming sections can be identified correctly. The format as is follows: the root element is a `document` element; each section is enclosed within a `section` element and nested sections are lifted up to be a child of the root node, rather than the parent section in the text (this is not by design, but rather a limitation of the current implementation); the title of each section is enclosed within a `title` element inside the corresponding `section` element; and the text of each section is enclosed within a `text` element inside the corresponding `section` element. An example is shown in Figure 3b.

5.2 Parsing Algorithm

The parsing algorithm is the part of the model that analyses the text. It extracts the concepts (nodes) and relations between concepts (edges) and inserts them into the graph. There are three main stages to the parsing algorithm: parsing the DOM (document object model) from the input XML; extracting concepts and relations between concepts from the text of each section; performing post-processing on the resulting graph to normalise/fix it. The algorithm is as follows:

1. Parse the input XML file and generate the document object model (DOM).

2. For each section element in the DOM:

   (a) Extract the title from the title element and convert it to lowercase.

(b) Extract the text from the text element and convert it to lowercase.

(c) Split the text into sentences.

(d) For each sentence:

   i. Tokenise the sentence into white-space separated words.

   ii. Tag each token with its part of speech (POS).

   iii. Normalise tokens by replacing them with their lemmas (the non-inflected/dictionary form of a word).

   iv. Analyse the dependency tree of the sentence to find the subject of the sentence.

   v. Optionally, analyse the dependency tree of the sentence and look for emerging concepts using a rule-based classifier (discussed below).

   vi. Chunk the POS tagged tokens using the grammars defined in Section 3.1 to create a constituency tree.

   vii. Extract the concepts identified by the grammars and insert the concept representing the subject and each extracted concept into the graph. Also create an edge between the subject and the extracted concepts.

   viii. Add constituent concepts of complex noun phrases to the graph in a similar way as above.

3. Perform post-processing steps on the resulting graph:

   (a) Assign a section to each concept - this represents where the concept was defined in the text.

   (b) Classify concepts as either a priori or emerging using a graph-based approach (described below). Only concepts that were not classified in the earlier steps are classified in this step.

   (c) Filter emerging concepts based on their word frequency (described below).

   (d) Mark forward and backward references using the method described in Section. 4. This method only works for edges between distinct nodes in the graph. Additional edges are added for concepts that are referenced verbatim (i.e. referenced in the same form) in preceding or following sections.

5.2.1 Assigning Sections to Nodes

In regards to step 3a, where a concept is defined in a given text is decided upon with the simple assumption that a concept is mentioned the most in the section that it is defined in (although this may not always be the case). This step is mainly for ensuring that forward and backward references are assigned more accurately in later steps, and as such is only relevant for emerging concepts. There are some cases where this method does not produce the correct results. For example, in the sample text on bread in Figure 3a, wheat flour is mentioned in the first section but not defined until the following section and the concept is mentioned once in each section. The first mention of the concept should create a forward reference. However, based on how nodes in the graph are assigned to a section (refer to Section 4.4), the concept 'wheat flour' would be assigned to the first section (due to the tie break condition) and thus the second time 'wheat flour' is mentioned it creates a backward reference to the mention in the first section. This particular case and similar cases can be corrected for by first observing that the concept appears in the title of a section, and assigning the concept to the section whose title contains the concept.

5.2.2 Filtering Emerging Concepts

For all of the methods described below, emerging concepts are filtered by their word frequency using the Zipf scale (Brysbaert et al., 2013; Van Heuven et al., 2014) to help prevent common words being misclassified as emerging concepts. If a classifier has decided to label a concept as an emerging concept, only concepts that are below the threshold value are accepted as emerging concepts, otherwise the concept is labelled as an a priori concept. This threshold value was set to 3.0. This value was found empirically by experimenting with different values and choosing the threshold value that gave the best results in experiments (see Sections 6.2 and 6.5.1 for the methodology and results of the experiment in question).

5.3 Comparing Methods for Classifying Concepts

In the experiments four different approaches to classifying concepts are evaluated: a graph-based classifier, a rule-based classifier, a classifier that combines the graph-based and rule-based classifier, and a random classifier.

These are described below.

5.3.1 Graph-Based Classifier

This approach classifies concepts using the graph topology. This approach works on the sometimes incorrect assumptions that: a concept that is only referenced from within a single section is an a priori concept; and concepts that are referenced from within multiple sections are emerging concepts. This is implemented by looking at the incoming edges for a given node and for the tail of each of these edges counting the number of distinct sections that these tail nodes belong to. Then we make a simple decision, if the number of sections is one then the concept/node is labelled as an a priori concept, otherwise if the number of sections is greater than one the concept is labelled as an emerging concept.

As alluded to above, there are cases where these assumptions do not hold. Using the Wikipedia article on closures (Wikipedia contributors, 2019b) as an example, the concept "closure operator" is referenced from multiple sections and is indeed an emerging concept. However, the concept "set" is referenced from multiple sections however it is not an emerging concept as the method might suggest.

It may be strange that this is being included even though it is based on incorrect assumptions. However, interesting results are produced when compared against an approach that is more faithful to the method described in Section 4.3.

5.3.2 Rule-Based Classifier

This approach is intended as the first steps to address the issues of the graph-based approach and improve the accuracy of the concept classification. In this approach the algorithm analyses dependency trees and matches simple patterns to pick out concepts that look like they are being defined (i.e. the concept is likely an emerging concept). The basic patterns that are matched are:

- Subjects of sentences where the main verb is 'be' and the object is a NP. For example, in the sentence "Apples are a type of fruit.", "Apples" would be labelled as an emerging concept.

- Sentences where the main verb is 'define':

  - Direct objects of sentences where the verb ‘define’ is used in the active voice. For example, in the sentence ‘These three properties define an abstract closure operator.”, “abstract closure operator” is labelled as an emerging concept.
  - Subjects of passive sentences where the main verb is ’define’. For example, in the sentence “The congruence closure of R is defined as the smallest congruence relation containing R.”, “congruence closure of R” is labelled as an emerging concept.

- Objects of passive sentences where the main verb is ‘call’. For example, in the sentence ‘This smallest closed set is called the closure of S ...”, “closure of S” would be labelled as an emerging concept.

5.3.3 Composite Classifier

This approach is a synthesis of the graph-based and rule-based classifiers. In this approach the rule-based classifier takes precedence, and any concepts not classified by it are then processed by the graph-based classifier.

5.3.4 Random Classifier

This approach labels concepts as a priori or emerging randomly with equal probability. Since there are no benchmarks to compare the other approaches for classifying concepts with, the next best thing is to compare it against is a classifier that assigns labels randomly. This approach is intended to serve as a baseline to compare the other methods against.

6 Evaluation

In this section, I will discuss how the quality of the model for quantifying conceptual density is evaluated. For evaluating the quality of the parsing algorithm, the extracted concepts and relations between concepts for a given document are compared against a set of annotations for that document. Then, we look at how the scores generated by model change as the order of the sections in a document are permuted. In the last experiment, a set of three documents are ranked in order of least to most conceptually dense based on human judgements and compared against the ranking produced by the model. First, I will describe the annotation process, then following that

I will describe the setup for the three experiments and finally I will discuss the results for each of these experiments.

6.1 Annotated Document

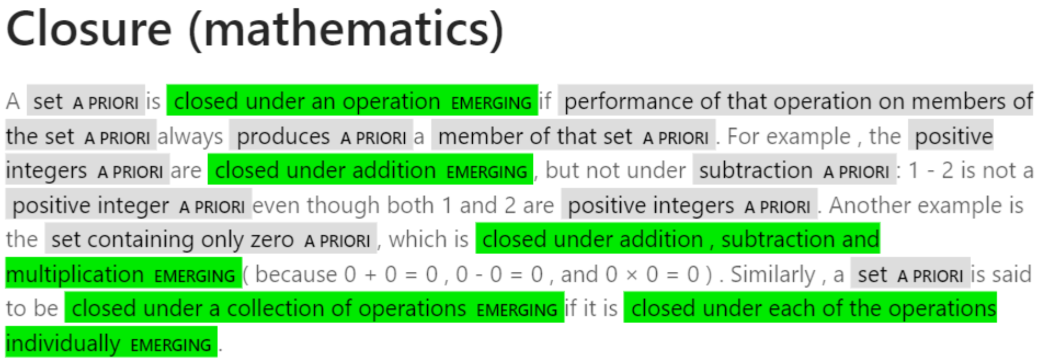
The Wikipedia article on closures (Wikipedia contributors, 2019b) was annotated with concepts (a priori and emerging) and references (forward and backward). A web-based tool was created to facilitate the creation of annotated documents and to automate the process of generating an XML version of the annotations for the evaluation scripts to use as input. The source code for this annotation tool is made publicly available (Dickson, 2019a).

Two approaches were taken when creating the annotations which targeted *low-level concepts* and *high-level concepts* separately. The idea of a low-level concept refers to an *atomic concept*, a concept that is named using a single token. Conversely, a high-level concept refers to a *compound concept*, a concept that is named using one or more tokens. For example, ' The idea of a high-level concept is similar to that of the complex noun phrases mentioned in Section 3.1. The reason why both approaches are used is that for a given phrase/sentence, there is not necessarily only one set of correct annotations. For example, in the phrase "A set is closed under an operation ..." it is clear that 'set' is a single, atomic concept. However, the same may not be able to be said about the verb phrase 'closed under an operation'. One could argue there are two concepts, 'closed' and 'operation'. However, one may also argue that 'closed under an operation' forms a single, composite concept that has a meaning distinct from the constituent concepts in isolation. How do we reconcile these different approaches? We could just choose one approach over the over, but we would risk losing potentially important information. To this end, the two sets of annotations that target low-level concepts and high-level concepts were created. These two sets of annotations are concatenated and used as the ground truth set of annotations for the purposes of evaluation. See Table 1 for the distribution of the annotations and Figure 6 for an example of some high-level concept annotations.

These annotations were checked by my two supervisors and any discrepancies were discussed and resolved accordingly in a rather ad hoc manner. However, in future work it would be best if documents are annotated independently by multiple judges so that inter-rater reliability can measured through something like Cohen's Kappa coefficient (Cohen, 1960; McHugh, 2012), for example. This would allow for the quality of the annotations to

A Priori Concepts	257
Emerging Concepts	94
Concepts Total	351
Forward References	3
Backward References	10
References Total	13
Annotations Total	364

Table 1: Counts for different types of annotations in the closures document.



Closure (mathematics)

A set A PRIORI is closed under an operation EMERGING if performance of that operation on members of the set A PRIORI always produces A PRIORI a member of that set A PRIORI . For example , the positive integers A PRIORI are closed under addition EMERGING , but not under subtraction A PRIORI : 1 - 2 is not a positive integer A PRIORI even though both 1 and 2 are positive integers A PRIORI . Another example is the set containing only zero A PRIORI , which is closed under addition , subtraction and multiplication EMERGING (because 0 + 0 = 0 , 0 - 0 = 0 , and 0 × 0 = 0) . Similarly , a set A PRIORI is said to be closed under a collection of operations EMERGING if it is closed under each of the operations individually EMERGING .

Figure 6: A sample of the annotations for the Wikipedia article on closures. The sections highlighted gray indicate a priori concepts and the section highlighted green indicate emerging concepts.

be assessed in a principled manner.

6.2 Entity Recognition

The first experiment looks at evaluating the quality of the parsing algorithm and its ability to correctly extract concepts and identify forward and backward references. We can characterise the parsing algorithm as an algorithm that tries to predict the ground truth set of annotations for a given document. The ground truth set of annotations for a given document are the distinct concepts and references from the corresponding annotated version of the document. We can evaluate the quality of the parsing algorithm by calculating metrics on the predicted set of annotations generated by it. The metrics used are precision, recall and the F_1 score (the harmonic mean of recall and precision). These metrics are calculated for each type of concept (a priori and emerging) and each type of reference (forward and backward). Additionally, these metrics are calculated for the four different approaches to classifying concepts described in Section 5.2. The results for this experiment are discussed in Section 6.5.1. Other, alternative parsing algorithms for entity recognition are also evaluated. However, these approaches are not the main focus of this report and as such these approaches and their results are briefly discussed in Appendix A. Below I will briefly define and discuss the metrics used in this experiment.

6.2.1 Definition of Metrics

Precision In binary classification, precision is defined as the proportion of samples labelled as *positive* by a classifier that are actually positive. Positive may refer to the presence of something as opposed to the absence of something (e.g. in image classification, the presence of a cat in an image from a set of images containing cats and dogs where the task is to identify cats), or more generally, it may simply refer to one of the two classes in the binary classification problem. Precision can be interpreted as how reliable a classifier is when it labels a sample as positive. Following on the from the previous example, precision could answer the question: out of the n samples some classifier labels as cats, how many samples are actually of cats? Precision can defined mathematically as the number of elements in the intersection between the set of ground truth labels and the set of predicted labels for positive samples, divided by the number of elements in the set of predicted labels for positive samples:

$$p = \frac{|t \cap y|}{|y|}$$

where t is the ground truth set of positive labels and y is the predicted set of positive labels.

Recall Recall is a measure of the proportion of positive samples that a classifier was able to identify out of all positive samples. Recall can be interpreted as how thorough a classifier is in identifying all instances of a given class. Using the example of classifying images of cats and dogs, recall could answer the question: out of the n cats in the data set, how many cats was the classifier able to identify/label as cats? Recall can defined mathematically as the number of elements in the intersection between the set of ground truth labels and set of predicted labels for positive samples, divided by the number of elements in the set of ground truth labels for positive samples:

$$r = \frac{|t \cap y|}{|t|}$$

where t is the ground truth set of positive labels and y is the predicted set of positive labels.

$\mathbf{F_1}$ Score F_1 score is the harmonic mean of precision and recall. It summarises precision and recall in a single metric and is calculated as such:

$$\mathrm{F}_1 = 2\frac{pr}{p+r}$$

where p is precision and r is recall. One characteristic of the harmonic mean is that it punishes extreme values more heavily in comparison to the arithmetic mean. For example, consider a precision and recall of 0.1 and 0.9. The arithmetic mean for these would be 0.5, however the harmonic mean would be 0.18.

6.3 Permuting the Order of Sections

The second experiment looks at evaluating the scores generated by the conceptual density model and how sensible these scores are when the order of sections in a document are permuted. The assumption is that the author of an expository text chooses to introduce concepts/topics (sections) in an order that is easy to read and understand, and by permuting the order of the sections in the text we expect the text to become more difficult to read and understand. In terms of the conceptual density model, permuting the order of sections will change the number of forward and backward references (however, it does not change the total number of references) based on the new section order (recall that whether a reference is considered a forward or backward reference is dependent on the relative order of sections). Due to the way references are weighted in the graph, the score generated by the model will also change as the sections are permuted. Hence, we can expect the model to reflect the change in readability of a given text through the scores that it generates as the sections are permuted, and by measuring these scores for different permutations we can evaluate the quality of the model without the need for human judgements. However, it should be noted that this method of evaluation does not necessarily directly measure the quality of the model in regards to conceptual density, rather it is a proxy that measures something like a readability score. The results for this experiment are discussed in Section 6.5.2.

6.4 Ranking Documents

Although there is no existing gold standard when it comes to evaluating models of conceptual density, we can evaluate the model against human judgements and intuition. In this experiment, the task is to rank a set of three Wikipedia articles (Wikipedia contributors, 2019d,a,c) from least to most conceptually dense. The documents were selected such that they all had roughly the same length and that there was a document for each of low, medium and high conceptual density.

Human judgements (rankings) were gathered by having each judge ($n = 3$) independently read the documents and rank them based on pre-theoretical assessments of conceptual density. Each judge was given a brief description of conceptual density to base their assessments on before starting. This description is as follows: "A tightly integrated concept is a concept that is difficult to describe or explain independently of other concepts". All judges reached the unanimous agreement that the ranking of the three documents based on this definition from least conceptually dense to most conceptually dense is as follows: House of Habsburg (Wikipedia contributors, 2019d), CDO (Wikipedia contributors, 2019a) and then DNA (Wikipedia contributors, 2019c). One thing to note is that the three judges were the two supervisors for this project and myself. This is perhaps not ideal since all judges were aware of the hypothesis being tested beforehand. Ideally for future work, documents would be ranked by judges with little to no exposure to the hypothesis being tested.

A ranking of the documents is derived from the model by generating a score for each document and then sorting by the scores. The results for this experiment are discussed in Section 6.5.3

6.5 Results

6.5.1 Entity Recognition

The conceptual density model was evaluated on the set of annotations for the Wikipedia article on closures as described in Section 6.1 and the results are listed in Figure 7. Denoted by 'Random' in the results table is the random classifier. Metrics that are calculated for the random classifier are reported with the mean and the margin of error representing the 95% confidence interval calculated from 40 independent trials. Denoted by 'Graph-Based' is the graph-based classifier, denoted by 'Rule-Based' is the rule-based classifier and denoted by 'Composite' is the composite classifier.

Overall, the graph-based classifier, the rule-based classifier, and the composite classifier are more accurate than the random classifier in most cases by a small margin. The only cases where the random classifier outperforms these three classifiers is for the recall and F_1 scores calculated on a priori concepts. Notably, for precision the margin between the three methods and random is small, ranging between -0.01% and 4.3%. The margin between the three methods and random is larger when it comes to recall, ranging between -9.6% and 38.9%. For F_1 score, the margin between the three methods and random is not as extreme, ranging between -6.7% and 8.9%. There is no single approach that always performs the best out of the four methods evaluated, however it appears that the composite classifier performs the best most of time.

There are a few reasons why the implementation of the model does not perform too well at identifying concepts:

- The parser does not handle actions (verbs and verb phrases) and thus fails to identify many concepts.

(a) Precision

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	0.361 (± 0.008)	**0.404**	0.396	0.361
Emerging Concepts	0.120 (± 0.005)	0.164	0.165	**0.170**
Forward References	0.024 (± 0.007)	**0.043**	**0.043**	**0.043**
Backward References	0.067 (± 0.008)	0.066	0.075	**0.085**

(b) Recall

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	**0.302 (± 0.009)**	0.237	0.222	0.206
Emerging Concepts	0.277 (± 0.014)	0.489	0.511	**0.521**
Forward References	0.278 (± 0.083)	**0.667**	**0.667**	**0.667**
Backward References	0.337 (± 0.039)	0.500	0.600	**0.700**

(c) F_1 Score

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	**0.329 (± 0.008)**	0.299	0.284	0.262
Emerging Concepts	0.168 (± 0.007)	0.246	0.249	**0.257**
Forward References	0.066 (± 0.008)	**0.080**	**0.080**	**0.080**
Backward References	0.111 (± 0.013)	0.116	0.133	**0.152**

Figure 7: Tables showing results from evaluating the parsing algorithm on annotations from the closures document. Results are rounded to three decimal places and the best method for classifying a priori and emerging concepts are emphasised in bold. The results for the 'Random' classifier calculated from 40 independent trials are reported as the mean and in parentheses the margin of error (95% confidence interval).

- The parser does not always chunk tokens in the same way as tokens are chunked in the annotations, e.g. '*thus a* subgroup of a group' produced by the model instead of 'subgroup of a group' in the annotations.

- For the rule-based classifier, the set of rules is rather simplistic and far from exhaustive. For example, it fails to recognise that the concept 'closed' is being defined in the sentence "A set is closed under an operation if ...".

- Particular to documents that contain equations, the way that how equations are parsed produces many spurious/meaningless tokens which adds noise and lowers precision. In the case of the composite classifier, about 35 of the extracted concepts are such tokens.

Despite the reasonable recall for references (although note the small samples size of 13), all methods result in low precision, at best 4.3% and 8.5% for forward and backward references, respectively. This is likely due to the fact that the current implementation identifies many more forward/backward references (116 and 141, respectively) in comparison to the actual number of references (3 and 10, respectively).

One explanation for why the parser produces too many references may be that due to the low precision of the concept classifiers, highly frequent concepts that are mentioned across many sections are erroneously marked as emerging concepts, which in turn would produce many references. One such example is the concept 'set' in the article on closures. The model erroneously marks the concept as an emerging concept and the concept is referenced from every section by a total of 43 different concepts. Of these 43 references to the concept 'set', 14 create a forward reference, 19 create a backward reference, and 10 occur in the same section and thus create neither a forward or backward reference. So due to the parsing algorithm marking 'set' as an emerging concept, 33 references are incorrectly identified. Removing these references from the predicted set of references would increase precision by at most 0.6% and 1.3% for forward and backward references, respectively. While this is the most extreme example, we can imagine how the quality of the set of forward and backward references would be adversely affected by having many concepts misclassified as emerging concepts, and how the metrics would improve if the classification of concepts was made more accurate.

6.5.2 Permuting the Order of Sections

For this experiment, the Wikipedia article on closures (Wikipedia contributors, 2019b) is used. For reference, the sections in their original order are: ‘closure (mathematics)’, ‘basic properties’, ‘closed sets’, ‘examples’, ‘closure operator’ and ‘binary relation closures’. Ideally, we would expect the model to generate the lowest score for this ordering. However, based on the current model the only thing that changes as sections are permuted is the assignment of forward and backward references. Hence, we expect the model to generate the lowest score for the ordering that minimises the ratio of forward references to backward references (forward references have larger weights than backward references). Based on these assumptions and the annotations for the closures article, the expected order of sections that give the lowest conceptual density could be: ‘closure (mathematics)’, ‘closure operator’, ‘closed sets’, ‘basic properties’, ‘examples’, ‘binary relation closures’. This ordering makes intuitive sense for the most part since sections that build upon concepts introduced in the article are placed towards the end.

The section ordering with the lowest and highest conceptual density given by the model were:

- lowest score of 2.31: ‘closed sets’, ‘examples’, ‘closure operator’, ‘basic properties’, ‘closure (mathematics)’, ‘binary relation closures’

- highest score of 2.43: ‘binary relation closures’, ‘closure (mathematics)’, ‘basic properties’, ‘closure operator’, ‘examples’, ‘closed sets’.

These results, while not exactly what was expected, are still somewhat intuitive. For the ordering with the lowest conceptual density, the section that explains the main topic/core concept is first and the section that seems to me to be the most conceptually dense is last. However, it is odd that the introductory section is placed second to last in the ordering that gives the lowest conceptual density.

We can reason about why the ordering given by the model does not match the expected ordering. The only thing that changes as we permute the sections is which references are marked as forward references and which references are marked as backward references. Thus, the reason why the output from the model does not match the expected output is that the model has not identified the set of forward and backward references that matches the set of references present in the annotations upon which the expected orderings are based on. Simply by comparing the number of references identified by the model (264) to the number of references present in the annotations (13), it is clear that this is the case. If the portion of the parsing algorithm that identifies references is improved, then the model would give results that are closer to what we would expect.

One may notice that there is only a difference of 0.12 between the ordering that gives the lowest conceptual density and the ordering that gives the highest conceptual density. The small difference is due to these main factors:

- Reordering the sections only changes the number of forward and backward references, and the weights for these only have a difference of 0.5.

- The scores are the average weighted outdegree of the generated graph, and there are only a few forward and backward references (3 forward references and 10 backward references across annotations for both low-level and high-level concepts) and many nodes in the graph (436) so the difference is small.

- The parser misses some forward and backward references which makes the difference even smaller.

6.5.3 Ranking Documents

The ranking produced by the model for the three Wikipedia articles (Wikipedia contributors, 2019d,a,c) is as follows, with scores in the parentheses: DNA (2.62), CDO (2.66), House of Habsburg (2.94). As previously stated, the expected ranking of the three articles is: House of Habsburg, CDO and then DNA. We can see that the model has produced results that are the opposite of the expected ordering. The main contributor to the model producing the opposite ranking is, by the very definition of the scoring algorithm, the average outdegree of the graph. The average outdegree, rounded to two decimal places, for the graphs generated for each document are: DNA (1.97), CDO (2.02) and House of Habsburg (2.14). Based on the rankings and definition of conceptual density, neither these numbers or the reverse order of the expected rankings are expected. Why is it the case that the document judged to have the lowest conceptual density has the highest score and average outdegree?

There are a couple of possible reasons for why the model produces the incorrect rankings that come to mind. It could be that the human judgements capture something that is not quite in line with the notion of conceptual density that the model is based on. For example, it could be that judges are ranking documents based on a notion that is most similar to a measure of readability. Another explanation could be that the writing styles for the documents differ in a way that biases the scores. For example, perhaps it is the case that in the article on the House of Habsburg is written in a way that means on average more concepts are mentioned within a sentence. This hypothesis is supported by looking at the average sentence length (in number of tokens) for each document. Rounded to one decimal place, the article on the House of Habsburg has on average 22.3 tokens per sentence, the article on CDOs 20.6 and the article on DNA 18.9. The fact that the final score reflects the structure of sentences is not surprising; the way that concepts are related is dependent on sentence structure (see Section 4.2).

This suggests two things: something in the implementation needs fixing; and the length of sentences needs to be controlled for in some way. There is some evidence for the former point. For example, in the graph generated for the sample text given in Figure 3a, an edge is created between ‘wheat’ and ‘typical ingredient of bread dough’ (see Figure 5). However, looking at the text and the structure of that sentence, it is clear that the concept ‘typical ingredient of bread dough’ is being mentioned in reference to the concept ‘wheat flour’, not ‘wheat’. Adjusting the way concepts are related to be more ‘concept-centric’ (as opposed to the current ‘subject-centric’ approach) would weaken the correlation between sentence length and the scores and perhaps shift the scores in a way that makes the expected ranking more likely to be produced by the model.

Despite producing the opposite of the expected ranking, looking at the data produced by the model there is also indication that different scoring schemes could produce the expected ranking. One case where the predicted ranking would match the expected ordering:

- ordering documents by the ratio of nodes to edges, which is like the inverse of the average outdegree (numbers in parentheses correspond to the number of nodes, edges and the ratio of nodes to edges):
  - House of Habsburg (2034/4351/0.47)
  - CDO (1778/3596/0.49)
  - DNA (2451/4823/0.51)

One alternative ranking that was also proposed was: CDO, House of Habsburg, DNA. There is one case where the predicted ranking would match the this ordering:

- ordering documents by number of emerging concepts when using just the rule-based concept classifier (numbers in parentheses correspond to the number of emerging concepts):
  - CDO (1223)
  - House of Habsburg (1506)
  - DNA (1669)

While promising, these observations may not hold as the parsing algorithm is made more accurate. However, it does raise the question of whether or not the assumptions that the current scoring algorithm is based upon are all correct.

6.6 Summary

In this section, I have described how annotations of documents can be created and methods for evaluating the conceptual density model. The current implementation of the conceptual density model was evaluated with the methods described in this section and the results were discussed. The main takeaway from the results is that much of the model is heavily dependent on how accurately concepts are identified, and as such future work should prioritise improving the parsing algorithm and the identification of concepts.

7 Related Work

7.1 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 seems to typically be concerned with named-entity recognition - the identification of named-entities such as names of people, companies, locations, and dates. In this study, we are concerned with concepts in general (all

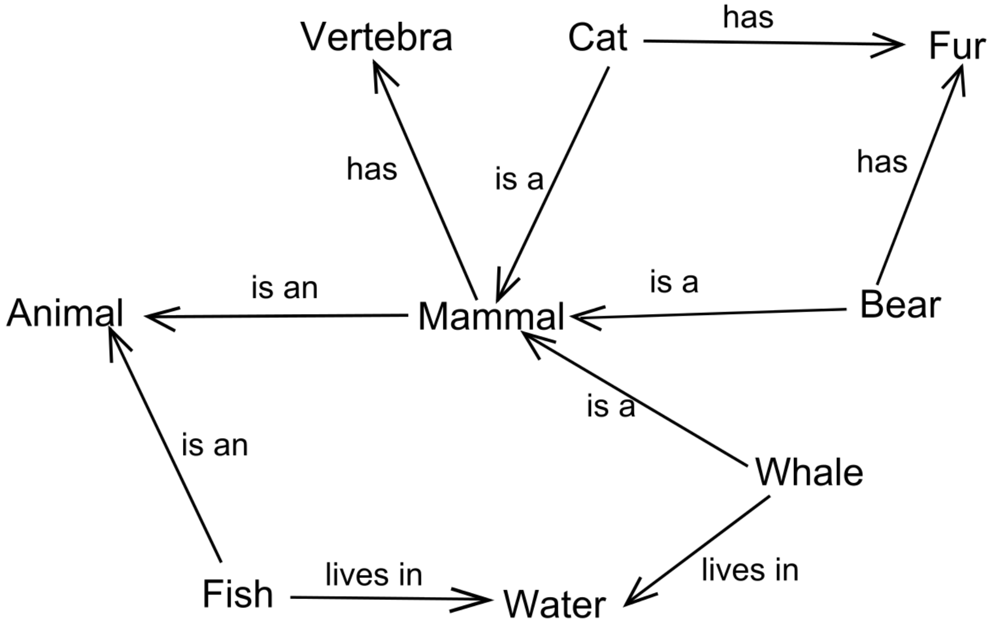


Figure 8: An example of a semantic network.[2]

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 the work-to-date, we have not been not too concerned about the nature of a relationship between concepts as much as the fact that a relationship exists.

7.2 Semantic Networks and Knowledge Graphs

The type of graph structure that is being built up in the conceptual density model is very similar to semantic networks/knowledge graphs. Some differences between the graphs produced by the conceptual density model and semantic networks are that: we are not annotating edges with the type of relationship (compare Figure 5 and Figure 8, 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 example, somewhat complex noun phrases are added as nodes in the conceptual density model rather than just simple noun phrases or the head of noun phrases.

8 Future Work and Open Problems

8.1 Suitability of Hypertext

The model of conceptual density was evaluated on articles from Wikipedia. One possible issue is that the methods explored in this report depend on the linearity of a text. Textbooks are a linear medium of text since you are expected to read each page in order and all of the necessary information for understanding the current section is given ahead of time. In contrast, hypertext documents are not necessarily designed in the same way. Concepts are often hyperlinks which can be followed, and when they are followed it is fair to consider the flow of the original document broken and the original document no longer linear. One case where hypertext may be considered linear is where the reader is one who understands all of the prerequisite concepts and thus follows very few or even zero links when reading the hypertext.

8.2 Type-Token Distinction

Another issue is the distinction between types and tokens. We can think of types as classes and tokens as instances of a given type/class, e.g. Dunedin is a token/instance of the type/class 'city'. An interesting question that arises is: should tokens be weighted differently from types - i.e. should tokens increase conceptual density as much types? Much of history talks about historical figures, is it difficult to understand what is going on even if we do not know exactly who the people being talked about are? In contrast, how likely is one to understand what a collateralised debt obligation or a mortgage backed security is if they do not know what an asset backed security is? It is more likely that one would be able to understand the interactions between historical figures (instances of people) than understand types of different tradeable financial assets even if they are already familiar with both people and asset backed securities.

Another issue is whether or not a token can be considered a concept at all. A concept is sometimes defined as an abstract idea or "a mental representation of the essential or typical properties of something, considered without regard to the peculiar properties of any specific instance or example" (Oxford University Press, 2015). Assuming we accept this definition of a concept, then we could argue that tokens are in fact not concepts.

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

8.3 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. 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 and requires them to adhere to a standard format, which is bound to be error-prone. At present, the sections and section headers must be explicitly marked up in XML. Identifying the section titles and where sections start and end automatically can be unreliable at times when using simple regex pattern matching due to different section heading formats and it is liable to false positives (see Table 2 in the appendix, for example).

8.4 Improving Entity Recognition

One issue with the current implementation is the quality of the extracted concepts and how the concepts are related. There are many cases where the extracted concepts and relations do not align with what a human would intuitively pick out. For example, under the current set of grammars, concepts that are captured by noun phrases are well covered but many concepts are actions (verbs or verb phrases), not things. For example, in programming, we can talk about 'calling functions', 'looping' and 'instantiating objects' as concepts, and all of these are examples of verbs or verb phrases. While the current implementation is adequate to build up a graph structure, important concepts and relations between concepts are likely to be missed.

Another issue is with accuracy of the classification of a priori and emerging concepts. While an attempt has been made at improving the accuracy of the parsing algorithm in this regard by utilising dependency parses, the results show that there is much work to do on the rule-based classifier. Previous work on identifying definitions in text by Fahmi and Bouma (2006); Saggion (2004); Iftene et al. (2008); Borg et al. (2007); Borg (2007) may provide direction in addressing this. It is possible that a model-based classifier trained for the task of entity linking may also provide a way of doing this. The advantage of this approach is that the knowledge-based approach could disambiguate difficult cases (word sense disambiguation).

As pointed out in the discussion of the results, it is likely that simply relating concepts to the subject of the sentence leads to degraded performance of the parsing algorithm. Making further use of dependency parses (e.g. for identifying appositive phrases) may enable the parsing algorithm to relate concepts in a more accurate way. This idea was investigated briefly (see Appendix A.1).

Overall, it is important that future work focuses on improving the quality of extracted concepts and relations by: extending the definition of a concept in text to include actions (e.g. verbs, verb phrases, nouns modified by verb phrases); improving the accuracy of the concept classifiers; and improving the quality of the extracted relations between concepts.

8.5 Weighting Repetition

One question that arises when analysing text documents under the lens of conceptual density is how should repetition of concepts be weighted when generating a numerical score? For example, which document would be more conceptually dense: one that mentions one emerging concept 50 times, or one that mentions three emerging concepts three times each? Arguments could be made in either direction: more repetitions gives more examples that help the reader understand the concept better, so the 50 repetitions of one emerging concept would be less dense than the alternative; or more repetitions may force the reader to consider the many interactions between the concept and other concepts, increasing cognitive load and increasing how tightly integrated the concept appears and thus the 50 repetitions of one emerging concept is more dense than the alternative. For example, how would this apply to the Wikipedia article on the House of Habsburg? There are a few members of the dynasty that play a prominent role in the narratives told in the article, and there are many members of the dynasty that are just mentioned in passing in a list or family tree. Surely, the few, central members of the dynasty have a much more rich history (at least in the context of the article) and much more complex relationships to other entities.

It is difficult to decide on which of these arguments is correct, assuming only one of these correct. However, it is likely that both are correct but more information is needed to judge how a high number of mentions for a given concept affects conceptual density.

8.6 Additional Graph Features

Many graph analysis techniques have not yet been explored in this pilot study (see (NetworkX Developers, 2004), for example). It would be interesting to see if incorporating features derived from these analysis techniques into the scoring algorithm 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.

8.6.1 Bottlenecks

Another interesting possible feature in the graph topology is the idea of a bottleneck vertex/vertices that might be able to be found by identifying the minimum cuts of the graph. One could imagine a bottleneck in a graph structure being visually similar to a hourglass. The vertex at the centre of the hourglass shape would be the bottleneck vertex. This would correspond to a concept that is defined in terms of other concepts and is used to define a set of other concepts. For example, in programming polymorphism and inheritance cannot be explained if classes are not explained, similarly classes cannot be explained if variables and functions are not explained. While in this example we could imagine back references from polymorphism to functions, to explain polymorphism we must first explain classes. Therefore, directionality must included when defining and identifying bottleneck concepts. Bottleneck concepts are important to the idea of conceptual density because they make learning more difficult.

8.7 Additional Text Features

There are other features that could be extracted from text. In (Robins, 2010) it is suggested that a measure of proximity of key concept terms could be used to help measure conceptual density. Some methods that may help with this are: various keyword extraction techniques (Mihalcea and Tarau, 2004; Matsuo and Ishizuka, 2004; Ercan and Cicekli, 2007; Rose et al., 2010); and long-distance dependency as a metric of reading comprehension difficulty (Liu, 2008).

8.8 Feature Importance Analysis

Following the process for ranking documents described in Section 6.4, we could repeat this process on a larger collection of documents to create a labelled data set. From this we could train a machine learning algorithm such as a simple linear regression or random forest. Then through feature importance analysis we could investigate which features out of all of the extracted features (from text and the graph structure) are the most salient in regards to conceptual density. And with a large enough data set, we could extract as many features as possible, regardless of whether or not they seem relevant to conceptual density, and then use this feature importance analysis to filter out the unimportant features in a principled way.

8.9 Word Embeddings

Word embedding models such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) could provide interesting avenues of research into how the latent semantics relate to conceptual density. These models may also provide ways of building up concept graphs for a subject (rather than for a single text) by facilitating easier interpolation of concepts (e.g. translating between different writing styles and different names for identical concepts, resolving synonyms and word sense disambiguation).

9 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 how the notion of conceptual density can be defined and measured in 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. This preliminary investigation brought forth many questions and methodological complexities which were discussed, and could be used to guide future work.

References

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

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, 2015.

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

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

Claudia Borg. Discovering grammar rules for automatic extraction of definitions. *Doctoral Consortium at the Eurolan Summer School*, pages 61–68, 2007.

Claudia Borg, Michael Rosner, and Gordon J Pace. Towards automatic extraction of definitions. *Proceedings of CSAW'07*, page 52, 2007.

Marc Brysbaert, Emmanuel Keuleers, Michael Stevens, Lise Van der Haegen, Ark Verma, Maaike Callens, Wim Tops, Vatsala Khare, Paweł Mandera, Heleen Vander Beken, et al. The zipf-scale: A better standardized measure of word frequency. 2013.

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

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Anthony Dickson. eight0153/Conceptual-Density-Text-Annotator: Initial Release, September 2019a. URL https://doi.org/10.5281/zenodo.3458618.

Anthony Dickson. eight0153/Quantifying-Conceptual-Density-in-Text v1.0.1, September 2019b. URL https://doi.org/10.5281/zenodo.3458623.

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

Ismail Fahmi and Gosse Bouma. Learning to identify definitions using syntactic features. In *Proceedings of the Workshop on Learning Structured Information in Natural Language Applications*, 2006.

Adrian Iftene, Ionut Pistol, and Diana Trandabat. Grammar-based automatic extraction of definitions. In *2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 110–115. IEEE, 2008.

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.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL http://www.aclweb.org/anthology/P/P14/P14-5010.

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.

Yutaka Matsuo and Mitsuru Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169, 2004.

Mary L McHugh. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282, 2012.

Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

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

Oxford University Press. Oxford english dictionary third edition — concept, 2015. URL https://www.oed.com/view/Entry/38130. Accessed on 19 September 2019.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

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

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20, 2010.

Horacio Saggion. Identifying definitions in text collections for question answering. In *LREC*, 2004.

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.

Walter JB Van Heuven, Pawel Mandera, Emmanuel Keuleers, and Marc Brysbaert. Subtlex-uk: A new and improved word frequency database for british english. *The Quarterly Journal of Experimental Psychology*, 67(6): 1176–1190, 2014.

Wikipedia contributors. Collateralized debt obligation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Collateralized_debt_obligation&oldid=916031111, 2019a. [Online; accessed 19-September-2019].

Wikipedia contributors. Closure (mathematics) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Closure_(mathematics)&oldid=907097840, 2019b. Accessed on 05 August 2019.

Wikipedia contributors. Dna — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=DNA&oldid=915559295, 2019c. [Online; accessed 19-September-2019].

Wikipedia contributors. House of habsburg — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=House_of_Habsburg&oldid=914872141, 2019d. [Online; accessed 19-September-2019].

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

A Alternative Parsing Algorithms for Entity Recognition

Aside from the parser described in the main portion of this report, which will be referred to as ‘XMLParser’ in this section, three other approaches were explored. The first, ‘CoreNLPParser’ uses dependency parses produced by the Stanford NLP group’s CoreNLP package (Manning et al., 2014). The second parser, ‘OpenIEParser’, leverages OpenIE, the open information extraction implementation from the same CoreNLP package. The third parser, ‘EnsembleParser’, uses all three parsers, ‘XMLParser’, ‘CoreNLPParser’ and ‘OpenIEParser’.

A.1 CoreNLP Parser

CoreNLPParser uses Stanford NLP group’s CoreNLP package to perform dependency parses and using the resulting tree structure it relates concepts in what is intended to be a more intuitive way. This is in contrast to the XMLParser which uses a regex-based chunker and the grammars defined in Section 3.1 to extract concepts and simply relates each concept in a sentence to the subject. The results for evaluating this parser on the annotations described in Section 6.1 are shown in Figure 9.

(a) Precision

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	0.284 (± 0.009)	0.345	**0.460**	0.358
Emerging Concepts	0.090 (± 0.007)	0.112	**0.116**	0.112
Forward References	0.085 (± 0.037)	0.100	**0.111**	**0.111**
Backward References	0.138 (± 0.023)	0.091	0.136	**0.143**

(b) Recall

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	**0.110 (± 0.004)**	0.074	0.089	0.074
Emerging Concepts	0.095 (± 0.008)	0.170	**0.191**	0.181
Forward References	0.144 (± 0.060)	**0.333**	**0.333**	**0.333**
Backward References	0.163 (± 0.026)	0.200	**0.300**	**0.300**

(c) F_1 Score

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	**0.159 (± 0.005)**	0.122	0.150	0.123
Emerging Concepts	0.092 (± 0.007)	0.135	**0.145**	0.138
Forward References	**0.244 (± 0.013)**	0.154	0.167	0.167
Backward References	0.153 (± 0.022)	0.125	0.188	**0.194**

Figure 9: Tables showing results from evaluating an alternative parsing algorithm on annotations from the closures document. The algorithm uses the Stanford's CoreNLP package for dependency parses. The numbers are rounded to three decimal places. The results for the 'Random' classifier calculated from 40 independent trials are reported as the mean and in parentheses the margin of error (95% confidence interval).

A.2 OpenIE Parser

This parser uses the OpenIE (Open Information Extraction) framework that has been implemented by the NLP group at Stanford University (Manning et al., 2014; Angeli et al., 2015). Entity-relation-entity triples are extracted from a text by the tool and the entities are inserted into the graph in a similar way to that of XMLParser. The benefits of using OpenIE is that it is an established tool and likely has already solved some the issues with the current implementation of the parsing algorithm for the conceptual density model. One issue with this OpenIE implementation is that it generates many spurious relation triples (see Table 3 in the appendix for an illustrative example). The results for evaluating this parser on the annotations described in Section 6.1 are shown in Figure 10.

(a) Precision

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	0.206 (± 0.010)	0.300	0.316	**0.333**
Emerging Concepts	0.060 (± 0.006)	0.073	0.074	**0.075**
Forward References	0.027 (± 0.007)	**0.036**	0.033	0.032
Backward References	0.076 (± 0.013)	0.065	0.077	**0.078**

(b) Recall

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	0.156 (± 0.008)	**0.163**	0.117	0.128
Emerging Concepts	0.122 (± 0.012)	0.191	**0.234**	**0.234**
Forward References	0.322 (± 0.080)	**0.667**	**0.667**	**0.667**
Backward References	0.217 (± 0.034)	0.300	**0.400**	**0.400**

(c) F_1 Score

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	0.178 (± 0.008)	**0.212**	0.170	0.185
Emerging Concepts	0.080 (± 0.008)	0.106	0.112	**0.113**
Forward References	0.064 (± 0.008)	**0.068**	0.062	0.062
Backward References	0.115 (± 0.017)	0.107	0.129	**0.131**

Figure 10: Tables showing results from evaluating an alternative parsing algorithm on annotations from the closures document. The algorithm uses the OpenIE parser from Stanford's CoreNLP package to extract entity-relation-entity tripes. The numbers are rounded to three decimal places. The results for the 'Random' classifier calculated from 40 independent trials are reported as the mean and in parentheses the margin of error (95% confidence interval).

A.3 Ensemble Parser

EnsembleParser, uses all three parsers: XMLParser, CoreNLPParser and OpenIEParser. The graph that EnsembleParser produces can be expressed as the union of the sets of nodes and edges produced by each constituent parser. The results for evaluating this parser on the annotations described in Section 6.1 are shown in Figure 11.

(a) Precision

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	0.193 (± 0.006)	**0.256**	0.244	0.230
Emerging Concepts	0.083 (± 0.004)	0.096	**0.099**	0.097
Forward References	0.016 (± 0.004)	**0.019**	**0.019**	0.018
Backward References	**0.048 (± 0.005)**	0.036	0.045	0.043

(b) Recall

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	**0.243 (± 0.007)**	0.132	0.128	0.109
Emerging Concepts	0.290 (± 0.014)	0.532	**0.553**	**0.553**
Forward References	0.344 (± 0.073)	**0.667**	**0.667**	**0.667**
Backward References	0.303 (± 0.036)	0.400	**0.500**	**0.500**

(c) F_1 Score

	Random	Graph-Based	Rule-Based	Composite
A Priori Concepts	**0.215 (± 0.006)**	0.174	0.168	0.148
Emerging Concepts	0.129 (± 0.006)	0.163	**0.168**	0.165
Forward References	**0.038 (± 0.005)**	0.036	0.037	0.035
Backward References	0.082 (± 0.009)	0.066	**0.083**	0.080

Figure 11: Tables showing results from evaluating the parsing algorithm on annotations from the closures document. The algorithm is an ensemble of the parser described in the main report, the CoreNLP-based parser and the OpenIE-based parser which output the union of the sets of nodes and edges in the graphs produced by each of the parsers. The numbers are rounded to three decimal places. The results for the 'Random' classifier calculated from 40 independent trials are reported as the mean and in parentheses the margin of error (95% confidence interval).

B Additional Tables

Table 2: An example of the current regex-based implementation for finding section headers and where sections start and end, using the text of the paper by Sweller et al. (2011). Asterisks mark cases where portions of text that are not section headers have been erroneously picked out.

Extracted Section Header
Chapter 15 The Element Interactivity Effect
Empirical Evidence for the Element Interactivity Effect
*The secondary task was presented on a separate computer and involved a tone
*When instructional materials involved low or no interaction between elements of
Element Interactivity and Understanding Instructions
*The extent to which we understand instructions depends on levels of element
Element Interactivity and the Modality Effect
Element Interactivity and the Expertise Reversal Effect
Element Interactivity and the Imagination Effect
Conditions of Applicability
*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
Instructional Implications
Conclusion

Table 3: An example of the relation triples extracted from the document in Figure 3a illustrating how the OpenIE implementation in Stanford CoreNLP generates many spurious triples. Notice that some of the relation triples are nonsensical, e.g. ('Bread', 'is', 'staple food prepared') in the second row.

Entity 1	**Relation**	**Entity 2**
Bread	is	food typically prepared from dough by baking
Bread	is	staple food prepared

Entity 1	**Relation**	**Entity 2**
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
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

Entity 1	**Relation**	**Entity 2**
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
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