Context & Common Knowledge

Using an entity linker and knowledge base to detect hard entities in need of an explanation

Wessel Poelman

Bachelor thesis
Information Science
Wessel Poelman
S2976129
December 18, 2020

ABSTRACT

Current automatic text simplification research is not very focused on named entities. These can definitely be a source of difficulty for a reader. A grammatically simplified sentence with a hard entity can still be difficult to understand. This project focusses on explaining difficult named entities in texts *when it is needed*. The following question is answered: *How can a system decide if named entities in a text need an explanation?* This is done by using an entity linker and knowledge base to retrieve information about an entity. The decision, if an explanation is needed or not, is made by using a list of 'common knowledge' entities and the context of a given entity in a text. Human annotators got the same task of making this decision and agreed in 76% of cases with the system. In conclusion, by comparing the context of a named entity to information in a knowledge base and by counting entity occurrences in a suitable corpus to determine which are common knowledge, a system can make a decent decision if a named entity needs an explanation or not.

The source code is available at: `https://github.com/WPoelman/thesis_is`

CONTENTS

PREFACE

This thesis was quite special for me. Almost exactly a year ago I completed the bachelor Communication and Information Sciences. While that was definitely interesting, I knew it was not what I wanted to pursue a career in. The Information Science field suits me a lot better and now I know I want to go further in the Information Science / Computer Science field.

The current project involved of a lot of trial & error and experimentation, which caused some headaches. When I found a configuration of parts that worked and combined them all, it was very satisfying and the fun part started. I enjoyed the building of the system the most and looking at the interesting, strange and sometimes funny output from it.

I am quite happy with the results considering it is something that (to my knowledge and research) has not been done before. Before starting this thesis I hardly knew that the field of 'automatic text simplification' existed. While working on this thesis I realized it is a surprisingly broad and interesting research area.

I would like to thank my supervisor Gertjan van Noord. Compared to my CIS thesis supervisor, there was not a lot of interaction, but the talks we had were useful and helped with some tough parts. This topic was also well suited for trying things on your own and did not need hand holding. I would also like to thank my mother for her proof reading (second time!) and general support.

I hope you enjoy reading this thesis.

Wessel Poelman

1 INTRODUCTION

Determining the difficulty of a text or sentence is not an easy task. Intuitively we have some ideas about what makes a text simple or hard. To explain or pinpoint what causes this, is not as trivial as it might seem. Is text difficulty caused by word choice, sentence length, topic, structure? Is it a combination of all of the above? As Shardlow put it in a 2014 automatic text simplification overview article:

> Simplicity is intuitively obvious, yet hard to define. (p.59)

This intuition is straightforward for humans, but not for a computerprogram. Making the decision if a text is difficult or not and what to do about it, can be done on many different levels. Altering the size of a text, rewriting sentences, restructuring information, replacing difficult terms with simpeler variants to name a few (Alva-Manchego et al., 2020). One of these levels is explaining difficult parts of a text. With this approach, the text is not made shorter to decrease the amount of information, but longer to provide more clarity and increase understandability. Or, more concisely put by Alva-Manchego et al. in their automatic text simplification overview article from 2020:

> (...) simplifying a text could also involve further explaining complex terms or concepts. This is not merely replacing a word or phrase for a simpler synonym or its definition, but to elaborate on the concept in a natural way that keeps the text grammatical, is meaning preserving, and is simple. (p.177)

This quote highlights two important challenges of explaining 'complex terms or concepts'. First, a resource is needed from which to create or get explanations. Second, a decision has to be made which terms are in need of an explanation. To solve the first problem there are numerous large and open resources (knowledge bases) that can be used for this. Wikipedia is a great example of a general knowledge base. For more domain specific knowledge there are also various resources available. The second problem is harder to solve. To illustrate, this problem has an effect on all aspects of the previous quote. Every recognized term from a knowledge base could be explained, this would 'elaborate on the concept', but not in a 'natural way'. There needs to be a method of deciding when an explanation is *needed*. This also touches on the other points from the quote. It is hard to imagine a grammatical or readable text in which every term is explained. This would lead to the opposite of the goal of making the text simpler. The question is *how* a system can make a decision when an explanation is needed or not.

The scope of this question is too broad. In practical terms, what is considered a 'term or concept'? This could be individual words, sentences or even chapters. The level of detail needs to be such that the term is easy to find in a knowledge base and has a short and simple explanation. This touches on several natural language processing fields, the most useful being 'entity linking', which comes close to the goal here. This field focusses on automatically recognizing entities from a knowledge base in a given text. This is a good starting point, but knowledge bases, such as Wikipedia, contain entities that are not suitable for simplifying using explanations. Take articles for individual years or dates for example. These most likely do not need explaining and how can a year even be 'explained'? To restrict the question, this project focusses on *named entities*. These include persons, geographical locations, brands or organizations. Luckily, entity linkers and general knowledge bases predominantly target named entities. In automatic text simplification research, named entities are also somewhat neglected. A good opportunity to try something new. The main research question is the following:

How can a system decide if named entities in a text need an explanation?

To make that decision, two factors have to be taken into account: context & common knowledge. A named entity can be explained by the context of the sentence or text it occurs in. Some entities are not explained in texts because they are considered to be common knowledge. A named entity such as 'Amsterdam' is generally not explained in texts. Two sub questions address these factors:

1. How can context be captured and used to decide if a named entity needs an explanation?
2. How can common knowledge be captured and used to decide if a named entity needs an explanation?

To answer these questions, a system was made that generates explanations for named entities, makes a choice if the explanation is needed and annotates an input text with the needed explanations. The system also produced versions with and without explanations of all sentences in which entities were recognized. These were shown to human annotators who had to make the same 'needed' or 'not needed' decision.

2 BACKGROUND

This chapter outlines where the current project ties into existing 'automatic text simplification using explanations' research. Next, the role of named entities & entity linkers is described. Finally, some disambiguation methods used by entity linkers are explained to illustrate the approach of dealing with the context of an entity in a text.

2.1 SIMPLIFYING USING EXPLANATIONS

As stated in the Introduction, explaining difficult terms is one of many ways to simplify a text. Some research has been done, but "this is an important research area in SS (sentence simplification) where limited work has been published" (Alva-Manchego et al., 2020). Two fields in this area will be highlighted: *healthcare* and *reading assistance*.

2.1.1 Healthcare

Healthcare is a field where hard terms contributing to the difficulty of a text is clearly visible. A decent amount of research has gone into explaining difficult medical terms. The systems developed for this often have a secondary screen or interface showing the explanation (Alfano et al., 2020; Alsheref and Fattoh, 2020; Qenam et al., 2017). A user needs to explicitly point out to the system which term they want explained. These systems use vocabularies of medical terms with so called 'consumer explanations' to find an explanation for a given term. These explanations are manually created by experts. This research and the systems are highly domain specific and have some drawbacks. The first being the fixed nature of the vocabularies. When a new term is added, the vocabulary needs to be updated. An expert needs to manually create a new explanation for that term. The same problem applies when updating existing terms with new explanations. This means that the system is quite hands on when it comes to extendibility and limited in the terms it can find.

The current project, while not applied to healthcare specifically, uses Wikipedia as a general knowledge base. The advantage of this, is that the system retrieves information in real time. At the moment of generating the explanation, the information about an entity is as recent as the last update on that entity's article. Another advantage is that, theoretically, all entities in the Wikipedia database can be explained. This is, however, limited by the recognition of those terms and the language used. This project is mainly limited by the entity linker. This real time approach is more extendible than the fixed vocabulary. The downside of using Wikipedia is that the quality of the information is not guaranteed. Expert explanations in the vocabulary approach are likely more consistent than the open editing nature of Wikipedia.

Aside from the vocabularies, the systems in healthcare use an all or nothing approach. Both the systems that use an external screen for the explanations and the systems that insert them into the source text overlook an essential part of explaining a term, which is context. The external method somewhat solves this by letting the user choose what they want explained. If the user understands the term they simply do not click on it. This is a convenience of the design, rather than a solution to the problem. If a term gets an inline explanation, but gets naturally explained in the next sentence, it arguably makes the text more confusing. The point of the current project is to explain a term only when it is *needed*. If a term gets explained in context, the system should be smart enough to *not* explain it and vice versa. What is considered 'context' will be clarified later.

As mentioned, a lot of the findings in the healthcare text simplification field are quite domain specific. Some of the discoveries are more broadly applicable though, such as a study from Gu et al. from 2018. This study looked at the use of parentheses when providing explanations in texts. The main experiment consisted of inserting the explanation in the text and surrounding the difficult term with parentheses, or vice versa. They showed that for already simple texts, it is best to enclose the explanation with parentheses and in difficult texts, the other way around. The current project uses online news articles, which are not considered to be difficult texts (more on this later). This is why the system in the current project encloses the explanations in parentheses behind the term that needs to be explained, following the recommendations of Gu et al..

2.1.2 Reading assistance

There have been some attempts at generating explanations for reading assistance. A study done by Watanabe et al. from 2010 looked at providing explanations for terms in web texts through the use of a browser plugin. The user could click on terms and a highlighted popup with an explanation would appear. Something similar has been done by Eom et al. in 2012 to disambiguate words for struggling readers and by Azab et al. in 2013 to help people with learning a foreign language. All three studies have some major drawbacks. They use tools or interfaces that need to be open outside of the actual text during reading. This arguably hinders the reader as they need to switch to the system and back to the text while reading. That 'break' in the reading process could result in needing to reread a section or needing to find where they were. While these seem like minor inconveniences, the tools are aimed at struggling readers. Any additional obstacles only make reading harder. Additionally, the systems are reliant on user input, usually in the form of clicking on text elements. This also interrupts the reading process. Finally, as with the healthcare systems, there is no regard for context, it is all or nothing. The current project aims to provide explanations for named entities *when needed*, based on context and common knowledge. Explanations are inserted into the original text and sourced from a general knowledge base. No external system needs to be open alongside reading the text.

2.2 NAMED ENTITIES

In current automatic text simplification research, named entities are somewhat neglected. In the previously mentioned paper by Alva-Manchego et al. from 2020, for example, a sentence splitting model is presented that "strips off named entities and properties" (p.164). When replacing difficult words with simpeler synonyms, named entities are often discarded or replaced by a dummy word in order to reduce the size of vocabularies (Nisioi et al., 2017; Alva-Manchego et al., 2020). Considerable research is dedicated to shortening sentences or grammatically restructuring texts. These approaches are important when simplifying, but a simplified sentence with difficult terms is not necessarily simple. Named entities are a good candidate to simplify because they can generally be explained in a couple of words. Take for example a recent news article from the University of Groningen about Ezinge[1]. In the article there is no clear explanation of what Ezinge is. This is assumed to be common knowledge and for most people from northern provinces of the Nether-

[1] https://www.rug.nl/news/2020/10/ezinge-revisited

lands, this would probably be true. This is, however, not guaranteed and a simple inline explanation in the first sentence could help with understandability:

> *Dutch:* De opgraving in de wierde van Ezinge **(Gronings dorp)**, tussen 1923 en 1934 uitgevoerd door prof. Albert Egges van Giffen, behoort tot de bekendste opgravingen in de geschiedenis van de Nederlandse archeologie.
> *English:* The excavation in the Mound of Ezinge **(village in Groningen)**, carried out between 1923 and 1934 by Prof. Albert Egges van Giffen, is one of the most famous excavations in the history of Dutch archeology.

Named entities themselves are a large research area within natural language processing. This field looks at recognizing named entities in a text, disambiguating those entities and linking them to a knowledge base (Shen et al., 2015; van Hulst et al., 2020). All these areas are important for the current project. One of the main components of the system is an entity linker, which performs all of them. Entity linkers are mostly used to assist components of information retrieval systems (Delpeuch, 2019; van Hulst et al., 2020). General entity linkers have not been used in text simplification applications using explanations. The discussed healthcare systems do link a term to an explanation, but they do not use a *general* knowledge base. Disambiguation is also rarely needed in those systems. Some of the reading assistance systems do use a form of entity linking, but there the system does not need to find the entities in the text, the user does that by clicking.

Current state of the art entity linkers use a variety of knowledge bases and approaches. For the current project a Dutch, pre-trained entity linker is needed. There are some recent attempts at Dutch entity linkers[2], but they are not as well maintained or recent as their English counterparts. Pre-trained systems for Dutch are even harder to come by. Additionally, some linkers are very domain specific, targeting only historical documents for example. For this reason, the system used in this project is DBPedia's Spotlight with their 2016 Dutch language model, created by Daiber et al. in 2013 (updated in 2016). This is not a state of the art system, but it provides a clear and usable API and comes with a pre-trained model. Spotlight provides entity recognition, disambiguation and linking using DBPedia as a knowledge base. The main knowledge base this project uses is Wikipedia. DBPedia provides methods to convert DBPedia entries to Wikipedia entries. More detailed information on Spotlight will be provided in the Method chapter.

2.3 INSPIRATION FROM DISAMBIGUATION METHODS

The disambiguation methods in entity linkers give a good starting point of how to look at the context of a named entity. Disambiguation involves looking at which variant of an entity is the right one *given the context*. For example, the sentences 'the Miami Heat beat the LA Clippers last weekend' and 'the Miami heat was tough for farmers this year' both contain the possible named entity *Miami heat*, thus being ambiguous. When looking at the context, the first sentence has another basketball team in it. When a system has access to a knowledge base, it can use that information to guess that the first one is probably referring to the basketball team and the second to weather conditions, based on the context.

This process is often done by providing a confidence score of which variant the system thinks is the closest. There are many disambiguation methods and techniques. The current project has been influenced by methods that leverage a

[2] This project started with experiments using:
- DAC from the Dutch National Library `https://github.com/KBNLresearch/dac`
- An entity linker for historical Dutch articles from the VU Amsterdam `https://github.com/cltl/entity-identification-from-scratch`

knowledge base to figure out what is happening in the context of an entity. Specifically, the current system looks at how *different* the context is to information in the knowledge base.

Looking at that context can be done in several ways, such as word vectors (embeddings), counting previous mentions or using the 'relatedness' of Wikipedia links (Delpeuch, 2019; Ma et al., 2019; van Hulst et al., 2020). Disambiguation algorithms and methods often do not use long form 'semantic' information from a knowledge base. They use data that allows for graph calculations or to enhance features of an entity (Rao et al., 2013). Nowadays, this data is often transformed into machine learning features. These include categories, identifiers, edges of related entities in graph representations or popularity. There is, however, a lot more information in knowledge bases that could be used in analyzing context. Take Wikipedia for example, short descriptions, summaries, even full articles provide a lot of information. The system in the current project uses this type of information to make the decision if an explanation is needed.

In a paper by van Hulst et al. from 2020, they used special `Wikipedia2Vec` word and entity embeddings. These encoded word embeddings as well as Wikipedia metadata. A context window of 50 words surrounding an ambiguous entity was used. This context was transformed to embeddings using `Wikipedia2Vec` and similarity was calculated between the context and entity embeddings. Other steps were used, but this particular method can also be used in deciding if an entity needs an explanation. In some way, disambiguation already gives information about how well the entity 'fits' in its context by using embeddings. This is not the same as explaining the entity in the context, but it does provide a starting point. The current project uses a context window consisting of the sentence in which the entity occurs and the sentences left and right from it. The context window is described further in the Method Chapter.

The current system uses the first sentence of a Wikipedia abstract from the knowledge base for comparison. This is done in order to keep the size of the context window relatively the same as the size of the information from the knowledge base. This also generally provides the most information about an entity in a short and succinct way. The abstract, along with the description of the entity, get transformed into word embeddings and compared to the context, similar as what van Hulst et al. did. When this information from the knowledge base is *different enough* from the context, an explanation is needed. The specific approach and how the system deals with common knowledge is discussed in the Method Chapter.

The following chapters explain the data used in the project, the validation procedure, the specifics of the system and finally the results and conclusions.

3 DATA AND MATERIAL

3.1 COLLECTION

3.1.1 Dataset

The corpus used for this project is the `DutchWebCorpus`, created by Wietse de Vries. This dataset consists of all online news articles from 2015 to mid 2019 from four major Dutch news outlets: NOS, NU.nl, de Telegraaf & de Volkskrant. The total number of articles is roughly 990.000, totalling more than 270 million words. This dataset is chosen because news articles contain a large number of named entities. Online news articles are also of suitable length. The system uses a decently sized context window to make the decision. Named entities in short texts, such as tweets, generally do not have sufficient context. News articles are also suitable because of their textual structure. Since they are not that long, they have to be compact and concise with presenting information. If the author thinks a named entity needs an explanation, it has to be done quite close to the first time that named entity is mentioned. Simply because there is not enough space to do this in a later chapter for example. Furthermore, the structure generally does not use distant explaining of terms. With books or long form articles, footnotes or registers can be used to explain complex terms 'far away' from the sentence they occur in. Online news articles do not have this luxury. The dataset is used for two purposes:

- Extracting information about all named entities to capture which named entities are common knowledge.
- To annotate texts with explanations and present the intermediate results to validate the choices of the system.

3.1.2 Entity linker

As mentioned in the Background Chapter, the entity linker used is DBPedia's Spotlight. This is by no means a state of the art entity linker, but this is not a big issue for the current project. The main focus is the decision if a named entity needs an explanation, not if every entity is recognized. The Method chapter elaborates on the configuration and use of Spotlight.

3.1.3 Knowledge base

The knowledge base used is Wikipedia. For the purposes of this project, it is a very good and large resource for general information about named entities. It also provides an excellent open API which serves two functions:

- Retrieving an explanation for a given entity.
- Retrieving information about the entity to compare its context to.

3.2 ANNOTATION

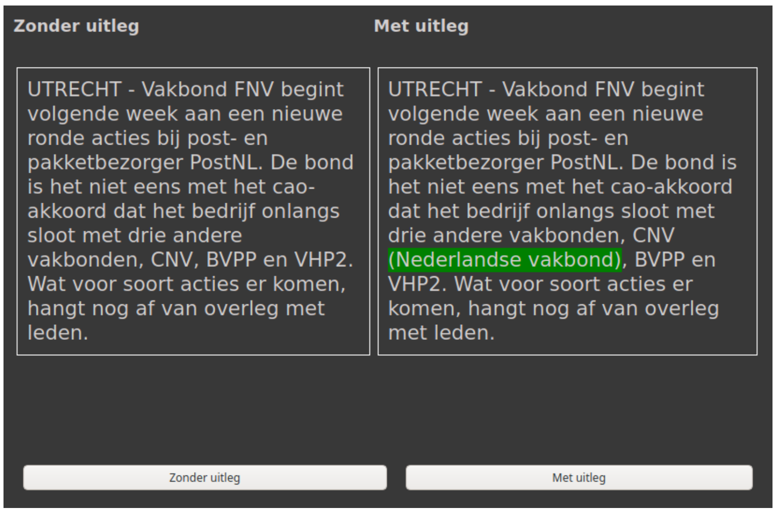
In the project there are two steps where texts are annotated, by the system and by human annotators.

3.2.1 System

The system receives an input text and decides which named entities need an explanation. When needed, the system inserts the explanation between parentheses behind the entity in question. The final output text is the original text with explanations. During this process, it produces detailed intermediate, annotated data. It creates a version of the sentence with and without the explanation for all named entities it recognized. These include the context sentences the system used as well for the decision. Human annotators needed to annotate this output.

3.2.2 Human annotators

The human annotators got to see two versions of the entity in context, with and without explanation. Figure 1 shows an example of this. The annotators got instructions to choose the version which makes the text more understandable and 'simple'. In other words, they needed to choose the one where the explanation was needed and supported the understanding of the text. The labels used were 'with explanation' and 'without explanation'. The full instructions can be found in the `validation.html` file in the GitHub repository.



Zonder uitleg

UTRECHT - Vakbond FNV begint volgende week aan een nieuwe ronde acties bij post- en pakketbezorger PostNL. De bond is het niet eens met het cao-akkoord dat het bedrijf onlangs sloot met drie andere vakbonden, CNV, BVPP en VHP2. Wat voor soort acties er komen, hangt nog af van overleg met leden.

Met uitleg

UTRECHT - Vakbond FNV begint volgende week aan een nieuwe ronde acties bij post- en pakketbezorger PostNL. De bond is het niet eens met het cao-akkoord dat het bedrijf onlangs sloot met drie andere vakbonden, CNV (Nederlandse vakbond), BVPP en VHP2. Wat voor soort acties er komen, hangt nog af van overleg met leden.

Figure 1: Example of what the human annotators needed to choose between.

3.3 PROCESSING

The final data gets processed and analyzed in several steps. First, interesting output of the system is analyzed, such as strange decisions or inaccuracies. This is done by looking through the output texts, the scores assigned to the decisions and the intermediate output of the system. Second, the human annotated data is compared with the system annotated data. This consists of first calculating inter-annotator agreement to check how similar the human annotators performed. Lastly, the agreement between the system and annotators is evaluated.

4 METHOD

Since there are a lot of moving parts, this section is structured as follows: First, the approaches used for the sub questions are explained. Next, the individual components of the system are clarified, along with a full walk through of the system. Finally, the validation process is outlined.

4.1 CONTEXT

This section explains the method for the first sub question:

> How can context be captured and used to determine if a named entity needs an explanation?

A named entity can be explained in the sentence it occurs in or in the surrounding text. As mentioned, in news articles this explaining is often close to the first mention of the entity. News articles do not have the length to explain an entity later on. Because of this, the context window used is the sentence the entity occurs in (S = sentence), one sentence before that (L = left sentence) and one after (R = right sentence). L and R do not always exist if S is the first or last sentence of an article.

The system needs compare the context (LSR) to something to decide if an explanation is needed. This is where the knowledge base comes into play. The following information about the entity is collected from the Wikipedia API:

- The `description` field (D)
- The `abstract` field (A)

D is a short description of the entity. For example, Amsterdam has 'capital of the Netherlands' as its description. The project uses the Dutch version of Wikipedia, but the approach is not language dependent. D is used in evaluating the context and also serves as the explanation that is inserted into the text.

The abstract is (a shortend version of) the first paragraph of the Wikipedia article about the entity. Here, the system only selects the first sentence (A). This is done because the first sentence of a Wikipedia article generally presents the most important information about the topic in a compact and direct way. For the purposes of this project, A can be considered the longer and more detailed version of D.

D and A are both used because, while D can give a decent amount of information about the entity, it is limited in scope. A gives a lot more information and casts a wider net of how the entity could be explained in the context. The larger amount of information in A does bring the risk of noise and useless information, more on that later. Next, some pre-processing is done for L, S, R, D and A:

- Stop words are removed
- Mentions of the named entity are removed

When 'context' is mentioned in the following sections, it refers to a combination of the cleaned versions of L, S & R. From now on D and A also refer to their cleaned versions. This cleaning is done in order to not 'taint' the similarity between L, S & R and D & A with noise. Stop words can wrongly make it seem like two very different sentences are quite similar. The entity is removed because it says nothing about *it being explained*, it is not part of the 'context'.

Next, L, S, R, D & A are converted to word embeddings using the large Dutch language model (`nl_core_news_lg`) from Spacy[1]. Spacy uses FastText (Bojanowski et al., 2016) for their word embeddings. These are trained on large web texts and Wikipedia, which is conveniently appropriate. In Spacy terms, L, S, R, D & A are `Spans` of `Tokens`, which roughly means a `collection` of `words`. These `Tokens` contain the word embedding vector for themselves and `Spans` can access these.

Finally, the similarity between the context and D & A is computed and a score is assigned. The built in Spacy method `similarity` on `Spans` is used. This method calculates the cosine similarity of the average of the word vectors in the two `Spans` that are being compared. The advantage of using this method is that it captures a lot of semantic information. This means that word order, synonyms or structure do not matter that much. It is all captured quite well by the individual word embeddings and the average of those embeddings. For example, if the entity is 'Washington D.C.', S is 'city America' and D is 'capital USA', the similarity between S and D is 0.63.

Calculating the similarities is done in pairs, consisting of an item from the context and D or A. These get summed for D and A and averaged to the number of items in the context. As mentioned, L and R might not exist and this deals with the difference in context length. Finally, a weight is assigned to the average similarity of D and A. This is done because (the uncleaned version of) D is inserted into the text. If that similarity is high, it is likely that the entity is explained in the context. A contains more information and is thus more prone to noise having an effect on the similarity, even when stop words are removed. With some experimentation the final weights are 0.7 for D and 0.3 for A. The sum of these is the final score and a threshold of 0.5 has been used. This means, with a score of 0.4, that the context is *not similar enough* to D and A. The entity, therefore, requires an explanation. With a score of 0.8, the context is *similar enough* to D and A. The entity is likely already explained in the context and does not need an explanation. The pseudocode in Algorithm 1 shows the process of calculating the score with all three items present in the context.

Algorithm 1 Pseudocode for producing similarity score

```
1:  $contextItems = [L, S, R]$
2:  $knowledgeItems = [D, A]$
3:
4:  for $contextItem = [L, S, R]$ do
5:      for $knowledgeItem = [D, A]$ do
6:          $x$ = average of word vectors $contextItem$
7:          $y$ = average of word vectors $knowledgeItem$
8:
9:          Compute cosine similarity of $x$ & $y$
10:     end for
11: end for
12:
13: $avgD$ = Sum similarities grouped by $D$ / length of $contextItems$
14: $avgA$ = Sum similarities grouped by $A$ / length of $contextItems$
15:
16: $finalScore = (0.7 * avgD) + (0.3 * avgA)$
```

[1] https://spacy.io/models/nl#nl_core_news_lg

4.2 COMMON KNOWLEDGE

The previous method tries to capture knowledge about the entity *inside* the text to decide if the explanation is needed. That decision is fine if the entity gets explained enough in the context window. When an entity does not get explained enough, however, the system would always decide to insert the explanation. This does not take the reason for not explaining an entity into account, which often is *common knowledge*. The previously mentioned example of Amsterdam illustrates this. Especially in news articles, this entity can be considered common knowledge. Something similar can be said about large sports clubs, celebrities, countries, world leaders, artists, brands and so on. The approach to tackle the second sub question is explained here.

> How can common knowledge be captured and used to decide if a named entity needs an explanation?

Since the dataset contains almost a million recent news articles, spanning almost four years, from four different sources, it is a suitable place to look for common knowledge. To 100% decide if a named entity is considered 'common knowledge', is almost impossible since it dependens on a lot of different factors. Current trends or the background of a person for example.

The method used in this project is the straightforward approach of counting all occurrences of named entities. This is done using the Spacy NER (named entity recognition) model, also from `nl_core_news_lg`. This model does not have state of the art performance, but is good enough for the purposes of this project. The process of counting was done by running the entire corpus of roughly one million articles through this NER model and keeping a count of all entities it encountered. The entities were normalized to all lowercase to make the count more accurate. This lengthy process resulted in a dictionary of 1.826.103 unique entities. The model was not restricted in categories it was allowed to find. This included entities such as dates and percentages. This was done on purpose since the approach of the other components was not finalized at this point. Categories other than named entities might have been useful as well. The resulting dataset has been added to the GitHub repository of this project[2]. It took a long time to create and might be useful for other projects. As an illustration, Table 1 shows the ten most common entities and their counts.

Table 1: Top ten most frequent entities in the DutchWebCorpus

Lowered entity	Occurrences
twee	353.610
eerste	329.076
nederland	278.561
nederlandse	200.816
amerikaanse	187.679
drie	171.847
tweede	148.224
amsterdam	135.706
één	130.717
vrijdag	106.078

Next, a cutoff point had to be chosen for when an entity was considered 'common knowledge'. After experimenting with numerous configurations, the cutoff was set at the top 1% most common entities. This created a list of 18.261 entities, with the least frequent entity occurring 118 times. If the system encounters an entity from that top 1%, it knows it does *not* need to explain it. Entities are first lowered

[2] The files to look for are *all_entity_counts.csv* & *all_entity_counts.pickle*

before checking if they are present in the list[3]. In the `support` folder of the source code, the scripts for creating this list are included. Another cutoff point can easily be chosen.

When looking through the list, the following categories of named entities are very common: cities, countries, athletes, world leaders, technology and clothing brands, artists and large companies. This captures quite a lot of common knowledge. The only 'type' of named entity that is not captured well with this method, is historic figures / events. These generally do not occur in news articles, unless a museum or something similar is mentioned. In this project the same dataset is also used for annoting, common knowledge from the `DutchWebCorpus` is therefore captured well enough. This method is specific to a corpus and use case.

4.3 SYSTEM COMPONENTS

This section will provide a walk through the system, starting with an article from the corpus as input. The descriptions will be brief and explain *why* and *how* the system works, not what it does in detail.

The article text first gets sent to the entity linker Spotlight, which is configured to look for names, organizations, persons and places[4]. Spotlight uses a confidence value between 0 and 1 for how sure it is about a found entity. After some experimentation, the allowed confidence for this project has been set at 0.4. This might seem low, but in testing some obvious entities, such city names, were ignored. From Spotlight a list of all recognized entities is returned with a DBPedia URI and the index of the entity in the text.

The URI gets translated to a Wikipedia title. DBPedia provides conversions for this and a cached lookup is created before the system is ran. With this title the system retrieves the description (D from previous section) and abstract (A) from the Wikipedia API[5]. The response from Wikipedia gets cached in an effort not to 'abuse' the API. The entity 'Amsterdam' alone would create more than 150.000 requests without caching if all articles were annotated.

If the previous steps were completed successfully, the system calculates the score for the entities to see if the explanation is needed. First, the system checks if the entity is in the 'common knowledge' list from section 4.2. If that is the case it returns a score of 1. If not, the system calculates the score using the context method explained in section 4.1. Next, the score is compared to the threshold of 0.5 (can be changed). If the score is *higher* than the threshold, the explanation is *not* needed. This is also why 'common knowledge' entities get a score of 1.

Next, the explanation, if needed, gets inserted behind the entity in parentheses into a copy of the original text. Finally, the annotated output text is produced. During the entire process, metadata is saved which is used for the validation task and to analyse what the system is doing. This metadata, per entity, consists of:

- The score
- The reason for the decision (common knowledge or context)
- The context window with and without explanation
- A version of the context window with the explanation highlighted
- The abstract (A)
- The explanation / description (D)

[3] A Python set is used in practice for efficiency, but a 'list' is easier to imagine here.
[4] Specifically the following DBPedia types: DBpedia:Name, DBpedia:Organisation, DBpedia:Person and DBpedia:Place
[5] https://nl.wikipedia.org/api/rest_v1/#/Page%20content/get_page_summary__title_

The final output is a file with the input text, output text and lists of annotated & ignored entities. The entity lists consist of the above mentioned metadata per entity.

4.4 VALIDATION PROCESS

In the corpus there are 20 'raw' text files with a large number of articles in them. An entire raw file of the corpus was ran through the system to generate a decent amount of output data. A file was randomly choses from those 20, resulting in `12.txt`, consisting of 51.558 articles. If an article did not have entities (or Spotlight did not recognize any) it was not added to the output. The final output consists of 36.245 articles. From these, a random selection for validation was made. The distribution of that selection is shown in Table 2.

Table 2: Distribution of sample sentences.

	With	Without	**Total**
Context	141	38	**179**
Common knowledge	n.a.	180	**180**
Total	**141**	**218**	359

There are no samples of 'common knowledge' & 'with' since common knowledge only filters out entities that *do not* need explanations. In other words, a common knowledge entity never receives an explanation.

These samples were inserted in an database (`select_data_for_validation.py`), along with some columns for keeping count of validations. A simple API (`api.py`) was created to get a random entity from the database which has less than two validations. It also stored the choice of 'with' or 'without'. As shown in Figure 1, people got shown the entire context window with and without explanation.

A page for the validations was made (`validation.html`) and both the page and API were put live. The link to the page was shared with friends, family and colleagues.

5 RESULTS AND DISCUSSION

This chapter presents some interesting findings in the system's output data. This will provide some insight in good elements of the system and areas of improvement. The second part presents the human annotation results and how they compare to the decisions of the system. When looking at the results, two baselines can be considered for the task:

1. Explain all entities
2. Explain no entities at all

The first becomes an unreadable mess, the opposite of the goal of making texts more understandable and simple. The second baseline ignores the possibility of entities being difficult or, from another point of view, considers all named entities 'common knowledge'. Of course there can be entities that are difficult or hard to understand in a text. To consider all named entities common knowledge is too large of a generatization and requires nuance. For both baselines, there is an approach in the middle and the results of the system hopefully show that.

5.1 GENERAL OBSERVATIONS

The majority of the system's output seems correct in terms of accuracy of the entity linker and the quality of the explanations. There were some strange outliers and mistakes.

Sometimes the description or abstract field was filled with the data of a Wikipedia disambiguation page. This is due to wrong DBPedia URI to Wikipedia title conversions. This was a rare occurrence and came to light when one of the human annotators emailed about it. These errors are quite detrimental to the system since it propagates to all components. The description and abstract give nonsense information in that case, which causes a wrong assesment of the context. Finally, the explanation itself is also wrong which makes the text more confusing. The system did have measures in place to filter out bad Wikipedia results. Checks such as skipping entities if the description or abstract were empty, but this type of error also needs filtering.

Some entities were wrongly recognized by the entity linker. The Dutch past tense of *open* is *opende*, which is also the name of a village in the Netherlands. This one was particularly often wrongly recognized. This ties into the same weakness of the system as the previous error. When an entity gets wrongly recognized or the accompanying information in the knowledge base is wrong, the system falls apart.

These errors are unfortunate, but a positive point from all this is that the system, theoretically, improves when the external parts improve. Better entity linkers translate to more entities, better accuracy and better disambiguation. Higher quality knowledge bases mean better explanations and abstracts. Both help to make the system perform better.

Apart from the errors, the system did a lot right. When reading through the output sentences, 'common knowledge' seems to be captured surprisingly well for such a simple approach. Most of the ignored 'common knowledge' entities truly seem obvious. The context part of the system also worked quite well, but is more difficult to judge since it is so text dependent. A couple of interesting examples:

```
Context: "In Callantsoog in de provincie Noord-Holland heeft
  vrijdagavond een grote uitslaande brand gewoed in drie
  grote bollenschuren. Daarbij zijn geen slachtoffers gevallen."
Entity: "Callantsoog",
Explanation: "plaats in Noord-Holland",
Extract: "Callantsoog is een dorp aan de Noordzee,
  in de gemeente Schagen in de provincie Noord-Holland.",
Score: 0.5329768361773242,
```

The score is higher than the threshold of 0.5 and there are indeed several indicators that this entity does not need an explanation. Both the description and abstract mention 'Noord-Holland', which also occur in the context. The abstract mentions 'provincie', which is also present in the context. This example also indicates an area of improvement. The score is relatively low for how similar the description and abstract seem. The description mentions 'plaats' and nothing similar is present in the context. This could be the reason for a relatively low description score, which is amplified by the high weight (0.7) it gets. It might have been better to restrict the context window for the description to more accurately account for this.

An example of the system deciding an explanation is needed, based on context:

```
Context: "Wie nog een leuk optrekje voor de aankomende zomervakantie
  zoekt, kan nu de villa van Michael Douglas op Mallorca kopen. De
  aanstaande koper dient wel over een gevulde portemonnee te beschikken,
  want de acteur zet zijn huis in Valldemossa te koop voor 32,5 miljoen
  dollar. Dit meldt Variety."
Entity: "Valldemossa",
Explanation: "gemeente in Majorca",
Extract: "Valldemossa is een gemeente in de Spaanse provincie en
  regio Balearen met een oppervlakte van 43 km.",
Score: 0.25121903982206717,
```

This named entity is not considered common knowledge. It is also not explained in the context that Valldemossa is a municipality (gemeente). The explanation does provide extra, possibly useful, information. As a reader you know it is a geographical location, but what exactly is not clear. This example does show a weakness of the knowledge base, the writer uses the more common spelling 'Mallorca', while the knowledge base uses 'Majorca'. In this case this is not such an issue and with well trained word embeddings, these two should be very similar (Spacy reports a similarity of 0.75). Interestingly, the system found the named entities Mallorca and Michael Douglas as well in this same text and considered them common knowledge. 'Variety' was not found, this could be due to it not being recognized by Spotlight or missing Wikipedia information.

5.2 VALIDATION

From the 359 output samples, 255 sentences were annotated, 48 of those were annotated by two annotators for a total of 303 annotations. The link for the annotation page was distributed among friends, family and colleagues. No personal information was collected about these annotators. It was not needed and the task was made to be as simple and straightforward as possible. This was done in an effort to get more annotations. All annotators are native Dutch speakers and the estimated age range is between 17 and 56 years old.

The inter-annotator agreement for the two time annotated sentences is 0.402, which is mediocre. This could be due to the relatively low number of double annotations or because there were some errors in the data. One notable error that slipped through the cracks is the previously mentioned disambiguation page information. Another reason might be some surprising or 'suspicious' annotations that indicate some annotators not taking the task seriously, not understanding the task or something else. Some examples of these 'suspicious' annotations are the following, where the annotator thought an explanation was needed:

- Eindhoven
- Amsterdam (3x)
- Emmen
- Arnhem
- Canada
- ProRail
- Apple
- Geert Wilders
- Donald Trump
- Femke Halsema

Maybe the task was too easy and people were just clicking the same button over and over again. It is a bit of a shame that a portion of the annotations is not that great, but there are still enough interesting findings. Table 3 shows the distribution of the annotated sentences.

Table 3: Distributions of system choices of annotated sentences.

	With	Without	**Total**
Context	58	17	**75**
Common knowledge	n.a.	180	**180**
Total	**58**	**197**	**255**

The 'common knowledge' category only determines 'without' explanation as it skips the entity if it occurs in the list. If it is not in the list, the entity goes to the 'context' part of the system. These sentences were randomly selected. The number of sentences where the system decided 'without explanation' is a lot higher. This is due to more of those types in the original output. In hindsight, this split could have been more balanced.

The total distribution of what category the annotators chose is 125 'with explanation' and 178 'without explanation'. This is a surprisingly high number of occurrences where the explanation was needed, almost two times as many as the system. This might tie into the suspicious annotations as those were also in the 'with' category.

Table 4 shows how many of the annotations labeled a sentence the same as the system. Sentences without human annotations are left out.

Table 4: Human and system agreement.

	With	Without	**Total**
Context	$\frac{44}{58}$ (76%)	$\frac{10}{17}$ (59%)	$\frac{54}{75}$ **(72%)**
Common knowledge	n.a.	$\frac{141}{180}$ (78%)	$\frac{141}{180}$ **(78%)**
Total	$\frac{44}{58}$ **(76%)**	$\frac{151}{197}$ **(77%)**	$\frac{195}{255}$ **(76%)**

These results show the system and the human annotators are in decent agreement overall. Of all annotated sentences, the system and annotators agreed in 76% of cases. This shows that, even with the possible suspicious or bad annotations in the 'with' category, the system performs decently. The agreement between the system and annotators is worst when the system decides that an explanation is not needed based on context. This could be due to the relatively low number of annotated sentences were this was the case (17). It could also be due to the system giving the sentence too high of a score when this is not justified. Here are two examples of those 17 'context & without' cases where the system and annotators disagreed:

```
Context: "Gezinsuitbreiding voor Jessica Simpson en haar man
  Eric Johnson. Dinsdag zette de zangeres hun derde kindje
  op de wereld.",
Entity:  "Jessica Simpson",
Explanation: "Amerikaans zangeres",
Extract: "Jessica Ann Simpson is een Amerikaanse zangeres en actrice.",
Score: 0.568518963445846,
```

The context mentions that Jessica Simpson is a singer (zangeres) and this is repeated in both the explanation and abstract. The explanation adds that she is an American singer, which might be why the annotator thought it could be necessary to explain it.

The next example shows another disagreement between the system and annotator in the 'context & without' category. It also includes an interesting quirk of the knowledge base:

```
Context: "De IJslandse maatschappij is niet de enige
  vliegtuigmaatschappij die de laatste maanden in zwaar
  weer is geraakt. Onder meer het Belgische VLM, het
  Duitse Germania en de Britse Flybmi moesten de
  deuren sluiten."
Entity: "VLM",
Explanation: "luchtvaartmaatschappij uit Verenigd Koninkrijk",
Extract: "VLM Airlines was een Belgische luchtvaartmaatschappij",
Score: 0.5997754807602497,
```

'Airline company' (vliegtuigmaatschappij) is mentioned in the context, which also occurs in a slightly different form in the abstract and explanation as 'luchtvaartmaatschappij'. These two have a word embedding similarity of 0.95 according to Spacy, which definitely gets picked up. An interesting quirk of the knowledge base is that the explanation states that the entity is Belgian, but the extract states British. Both are not false, but not the whole truth either. VLM was originally founded in Belgium, but was later purchased by a British airline company. This shows how tricky the quality and accuracy of a knowledge base can be for making the decision. In this case, the system also recognized that Belgian (Belgische) was both in the context and sentence. The decision to not explain the entity seems justified.

All disagreements in the 'context & without' category have a score between 0.503 and 0.599. They hover just around the threshold of 0.5. Some mistakes are to be expected that in the area.

When looking at the 'common knowledge & without' category, the system and annotators agree 78% of the time. This is quite high and shows that the system pretty accurately captures 'common knowledge' entities. This is, again, highly dependent on the corpus and might not translate well to texts outside of it.

Finally, the 'context & with' category shows the system and annotators agreeing 76% of the time. This is a decent result and this category also the highest number of disagreeing annotators. This shows that it is not as black and white to decide if an explanation is needed or not. An interesting example of human and system disagreement in the 'context & with' category:

```
Context: "Het Britse muziekduo Lighthouse Family heeft na achttien
  jaar stilte nieuwe muziek aangekondigd. Hun comebacksingle"
Entity: "Lighthouse Family",
Explanation: "Britse band",
Extract: "Lighthouse Family (1993-heden) is een Britse muziekgroep,
  gevormd door Tunde Baiyewu (zang) en Paul Tucker (keyboard).",
Score: 0.24097628545364733,
```

The system gives this a surprisingly low score considering 'muziekduo' and 'band' or 'muziekgroep' are very similar. This could be due to the strange cutoff point of the last context sentence, in the full text this sentence is the following:

> Hun comebacksingle My Salvation gaat donderdag op BBC Radio 2 in premiere, zo hebben zanger Tunde Baiyewu en toetsenist en producer Paul Tucker bekendgemaakt.

This shows that all parts of the system can make mistakes. In this case it was the sentence splitting algorithm of Spacy making a mistake. This resulted in a strange context sentence, which resulted in a low score. When the correct sentence is fed to the system, it gets a score of 0.48. This is a lot more understandable, but still too low considering the information given in the context. This, again, shows the difficulty of figuring out the correct threshold. Just as with the previous examples, this score hovers right around 0.5. The system is close, but everything needs to be 'good enough' for it to perform well: accuracy of entity linking, confidence of entity linker, sentence splitting, knowledge base, weights, threshold and more. This is a delicate balance. Considering the 72% agreement between system and annotators for context, 78% for common knowledge and 76% overall, it seems that this balance was quite decent for the system created in this project.

6 CONCLUSION

6.1 RESEARCH QUESTIONS

This project started with the following research question and sub questions:

> How can a system decide if named entities in a text need an explanation?
>
> 1. How can context be captured and used to decide if a named entity needs an explanation?
> 2. How can common knowledge be captured and used to decide if a named entity needs an explanation?

The results showed that information from a knowledge base can be used to capture how well the context of a named entity explains it. A short description and the first sentence of the abstract from the knowledge base were used to check their similarity to the entity's context. This context window was set at three sentences and were stripped from the entity itself and stop words in an effort to only capture relevant information. Weights of 0.7 and 0.3 were assigned to the description and abstract. This accounted for the description being used as the explanation in the text and the abstract possibly containing more noisy information. The results show that when the system made a decision based on context, human annotators agreed with that decision in 72% of cases.

The second sub question was tackled by counting all occurrences of all named entities in the corpus. The top 1% of the most common entities were deemed common knowledge and ignored by the system. The results show that here the system and human annotators made the same decision in 78% of cases. The answer to the main research question can be summarized as follows:

> By comparing the context of a named entity to information in a knowledge base and by counting entity occurrences in a suitable corpus to determine which are common knowledge, a system can make a decision if a named entity needs an explanation or not.

6.2 REFLECTIONS AND SUGGESTIONS

Overall, the system preformed quite well and made the same decision as the human annotators in 76% of cases. As shown in the results, there are improvements to be made. First, there should be more measures in place to catch inaccuracies or errors from the external parts of the system. Currently, there are some, but these are not enough in some cases. Examples of external errors that the system could try to catch:

- Disambiguation pages instead of an entity
- Wrongly recognized entities
- Strange information in knowledge base
- Sentence splitting

The system should theoretically improve the better the external components perform. Since there are not a lot of Dutch entity linkers, it might be interesting to alter the system to use a state of the art English entity linker. The approach in this project is not language dependent, to apply the system to another language the system needs:

- An entity linker
- A corpus from which to extract common knowledge entities and to create samples for validation from
- Access to a version of Wikipedia (or another knowledge base with similar fields)
- A Spacy model (or another NLP method) with sentence splitting and word embeddings

These parts are not hard to swap out. The most important part is mapping the responses from the different components to the correct datatypes for the system.

To get to the current system, a lot of trial & error and experimentation was needed since nothing similar has been done before. This means that there are most certainly improvements to be had when trying different combinations of weights, thresholds and parameters. The performance of current configuration is decent and can be changed easily.

Experiments with different Dutch word embeddings were also done, a large `word2vec` model, a large `GloVe` model and a standalone `FastText` model. From all these, the built in Spacy one was the smallest, but preformed very similarly. Considering the computational cost of the other models, the choice fell on Spacy. For further research, other embedding approaches might be better for the task. The `wikipedia2vec` model used by van Hulst et al. (2020) could be an interesting alternative since it possibly captures more information from the knowledge base that might improve performance. The current Spacy embeddings are also trained on Wikipedia, but not on metadata.

Finally, the source code of the system contains more explanations and details and can be found at: `https://github.com/WPoelman/thesis_is`. It is a shame that the validations are not of ideal quality. It seems that the system performed quite well and that the project combined some approaches that have not been done before in the automatic text simplification field. The current approach is language independent and more configurations and methods are possible for the task. In closing, this might be an interesting area to explore further.

BIBLIOGRAPHY

Alfano, M., B. Lenziti, G. Lo Bosco, C. Muriana, T. Piazza, and G. Vizzini (2020, 03). Design, development and validation of a system for automatic help to medical text understanding. *International Journal of Medical Informatics 138*, 104109.

Alsheref, F. K. and I. E. Fattoh (2020). Medical text annotation tool based on ibm watson platform. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 1312–1316. IEEE.

Alva-Manchego, F., C. Scarton, and L. Specia (2020). Data-driven sentence simplification survey and benchmark. *Computational Linguistics 46*(1), 135–187.

Azab, M., A. Salama, K. Oflazer, H. Shima, J. Araki, and T. Mitamura (2013). An nlp-based reading tool for aiding non-native english readers. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pp. 41–48.

Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2016). Enriching word vectors with subword information. *CoRR abs/1607.04606*.

Daiber, J., M. Jakob, C. Hokamp, and P. N. Mendes (2013). Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

Delpeuch, A. (2019). Opentapioca: Lightweight entity linking for wikidata. *CoRR abs/1904.09131*.

Eom, S., M. Dickinson, and R. Sachs (2012, 06). Sense-specific lexical information for reading assistance. pp. 316–325.

Gu, Y., G. Leroy, and D. Kauchak (2018, 04). When synonyms are not enough: Optimal parenthetical insertion for text simplification. *Annual Symposium proceedings. AMIA Symposium 2017*, 810–819.

Ma, H., Y. Li, and Y. Li (2019). An entity linking method based on entity category and word embedding. In *Journal of Physics: Conference Series*, Volume 1284, pp. 012028. IOP Publishing.

Nisioi, S., S. Štajner, S. P. Ponzetto, and L. P. Dinu (2017). Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 85–91.

Qenam, B., T. Kim, M. Carroll, and M. Hogarth (2017, 12). Text simplification using consumer health vocabulary to generate patient-centered radiology reporting: Translation and evaluation. *Journal of Medical Internet Research 19*, e417.

Rao, D., P. McNamee, and M. Dredze (2013). Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, multilingual information extraction and summarization*, pp. 93–115. Springer.

Shardlow, M. (2014). A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing 2014 4*(1).

Shen, W., J. Wang, and J. Han (2015, 02). Entity linking with a knowledge base: Issues, techniques, and solutions. *Knowledge and Data Engineering, IEEE Transactions on 27*, 443–460.

van Hulst, J. M., F. Hasibi, K. Dercksen, K. Balog, and A. P. de Vries (2020). Rel: An entity linker standing on the shoulders of giants. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20. ACM.

Watanabe, W. M., A. C. Jr., M. A. Amâncio, M. D. Oliveira, T. A. S. Pardo, R. P. M. Fortes, and S. M. Aluísio (2010). Adapting web content for low-literacy readers by using lexical elaboration and named entities labeling. *New Review of Hypermedia and Multimedia 16*(3), 303–327.