Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 7179–7186
Marseille, 11–16 May 2020

CoCo: A Tool for Automatically Assessing Conceptual Complexity of Texts

Sanja Štajner[1], Sergiu Nisioi[2], Ioana Hulpuş[3]
[1]ReadableAI, Cologne, Germany
[2]Human Language Technologies Research Center, University of Bucharest, Romania
[3]Data and Web Science Group, University of Mannheim, Germany
sanja.stajner@readableai.com, sergiu.nisioi@gmail.com, ioana@informatik.uni-mannheim.de

Abstract

Traditional text complexity assessment usually takes into account only syntactic and lexical text complexity. The task of automatic assessment of conceptual text complexity, important for maintaining reader's interest and text adaptation for struggling readers, has only been proposed recently. In this paper, we present CoCo - a tool for automatic assessment of conceptual text complexity, based on using the current state-of-the-art unsupervised approach. We make the code and API freely available for research purposes, and describe the code and the possibility for its personalization and adaptation in details. We compare the current implementation with the state of the art, discussing the influence of the choice of entity linker on the performances of the tool. Finally, we present results obtained on two widely used text simplification corpora, discussing the full potential of the tool.

Keywords: conceptual complexity, spreading activation, text analysis tool

1. Introduction

To actively engage readers in the text they are reading and maintain their interest in the topic, texts needs to be on an adequate level for the reader in terms of its lexical, syntactic, as well as its conceptual complexity (McNamara et al., 2006). Lexical complexity refers to lexical choices used in the text, e.g. the frequency of words in everyday life, richness of vocabulary, use of some specific terminology or foreign words, etc. Syntactic complexity refers to the length of the sentences and their syntactic structures, e.g. the number of subordinate or coordinate clauses, use of passive voice, unusual sentence structures, depth of the syntactic tree, etc. Conceptual complexity refers to the model of reading comprehension proposed by Kintsch and van Dijk (1978). According to this model, the reader needs to understand both individual propositions and concepts in the text, as well as their relations, in order to make a coherent story and fully understand the text. Text difficulty could thus be seen as the amount of gaps in the text coherence, and the effort required by the reader to repair them by inference making (Arfé et al., 2017).

Assessment of text complexity has a long history in education and psycholinguistic research (DuBay, 2004). It is used to assess the reading level required by the students to understand the materials in the schoolbooks, complexity of army manuals, or complexity of documents available for general public, such as healthcare documents for example. Only in the second half of the last century, over 200 readability formulae were proposed with the aim of measuring text complexity (DuBay, 2004). Despite many criticisms of their simplicity (DuBay, 2004), some of them gained high popularity, e.g. the Flesch Reading Ease score (Flesch, 1949), Flesch-Kincaid readability formulae (Kincaid et al., 1975), SMOG formula (McLaughlin, 1969), or Fog readability formulae (Gunning, 1952), due to their high correlation with variables shown to be indicative of reading difficulty (Bormuth, 1966) and their easy automatic computation.[1]

Recently, we have witnessed a growing number of commercial tools for assessing text complexity, e.g. TextEvaluator[2], Lexile Analyzer[3], and many others. All those tools claim to capture text complexity along several dimensions, but those dimensions focus only on text layout, vocabulary, and syntactic structures. None of them tries to assess the cognitive effort required from the reader to understand the concepts mentioned in a text and connect them in a coherent way. The same criticism stands for readability formulae that can be computed automatically.

To bridge this gap, Štajner and Hulpuş (2018) proposed the task of automatic assessment of conceptual text complexity, defining it as the amount of background knowledge necessary to understand how the mentioned concepts are interconnected. They explored the possibility of using DBpedia (DBpedia, 2014) as a proxy to background knowledge for this task. A year later, an unsupervised method for tracking conceptual complexity of texts using the spreading activation over DBpedia knowledge graphs was proposed (Hulpuș et al., 2019). The unsupervised method outperformed the previously proposed supervised method showing excellent results when tested in two different automatic text assessment scenarios: (1) finding conceptually simpler of the two versions of the same news story; and (2) ranking the five versions of the same news story according to their conceptual complexity (Hulpuș et al., 2019).

Given that such framework offers many possibilities for further explorations and adaptations to particular user/research needs, but is, at the same time, very difficult to build from scratch, we here offer the full implementation with some additional functionalities, both as an API and as the full code (Section 3). The entity linker used in the originally proposed framework (Hulpuș et al., 2019) is not freely available. Therefore, in the current implementation, we use DBpedia Spotlight (Daiber et al., 2013) which is freely available for several languages (DBpediaSpotlight, 2019). At the same time, this enable us to explore how much the

[1]Freely available through *style* package in Linux.

[2]https://www.ets.org/c/23491/

[3]https://lexile.com

choice of the entity linker has on the final results of conceptual complexity (Section 4). The current implementation is offered in such a way that it is easy to replace the currently used entity linker (DBpedia Spotlight) with any other (see Section 3). To further emphasise the importance and capabilities of this tool, we show how it can be used to detect if conceptual simplification is present in a given text simplification corpus (Section 5).
The contributions of our current work can be summarized as follows:

- We open source CoCo - the first automatic system which measures conceptual complexity of texts (Section 3);
- We offer CoCo via API endpoints, and as a full code with modular architecture (Section 3);
- We discuss in details all the parameters of CoCo, emphasising on how they can be adjusted to adapt the system to different target audiences and/or research purposes (Section 2);
- We evaluate our implementation comparing it to the state of the art (Section 4);
- We show that the choice of the entity linker can significantly influence the results, however still achieving high performance on the benchmark dataset (Section 4);
- We run the current implementation on two text simplification corpora, showing how CoCo can be used to detect if the corpus contains conceptual simplification (Section 5).

2. Spreading Activation Framework for Tracking Conceptual Complexity of Texts

The main idea behind the proposed framework (Hulpuș et al., 2019) is that concepts are activated in working memory during reading process. DBpedia knowledge graph is used as a proxy to **long-term memory** over which **spreading activation** processes run and bring concepts into the **working memory**. The framework assumes that text is processed sequentially. During that process, each mention of a DBpedia concept in a text triggers a tide of spreading activation over the DBpedia knowledge graph. At the same time, the activation of the concepts which are already in the working memory decreases as reading progresses, modelling thus a **forgetting process** which naturally happens during reading. However, if the same concepts, or their related concepts (related in DBpedia knowledge graph), are encountered in text, the activation of those concepts (and their related concepts) in working memory increases. The cumulative activation (CA) of the mentioned concepts is tracked at different points in time: at the encounter (AE), at the end of sentences (AEoS) and at the end of paragraphs (AEoP). Those values are then used to estimate the conceptual complexity of texts. The hypothesis is that *a higher activation of text concepts in working memory indicates more accessible texts (already activated concepts in working memory are easier to understand and interconnect).* As it has been shown that AEoS has the best correlation with the Newsela complexity levels (Hulpuș et al., 2019), in the demo we present, the conceptual complexity (CoCo) value is based on AEoS only.
A number of parameters allows further specifications of processes that happen in long-term memory, and in working memory. Here we give just some brief intuitions behind each of them, which are necessary to understand parameters of the CoCo tool, and its full potential for adaptation to different tasks and reading populations. For a detailed description of spreading activation framework for tracking conceptual complexity of texts, we refer readers to the original paper (Hulpuș et al., 2019).

2.1. Long-term Memory Parameters

The spreading activation happens over DBpedia knowledge graph which is used as a proxy for the long-term memory. It follows the rules common to all spreading activation models in literature. During sequential reading process, when a mention of a concept is encountered in the text, that concept becomes active in the long-term memory with an activation value of 1.0 and it *fires* spreading its activation to its neighbours in the knowledge graph (long-term memory), which can then further *fire* and activate their neighbours as long as some preset conditions are met. Those multiple iterations (firings of newly activated nodes) that all come from a single mention of a concept in text are called pulses (*p*). In the proposed framework, four parameters control the spreading activation process in the long-term memory:

- **distance decay parameter** (α)
- **firing threshold** (β)
- **exclusivity** (exc)
- **popularity** (pop)

The first two parameters, distance decay parameter α and firing threshold β are used to compute the value of the **output function** which defines how much activation is output by a concept at pulse $p + 1$, given its activation at pulse p. **Distance decay parameter (α)** reduces the activation going out of each node exponentially with respect to p. **Firing threshold (β)** is the minimum activation that a node needs to have in the pulse p in order to be able to fire. Those two parameters offer a possibility for personalization of the approach according to the memory capacity of the reader. Lower values of α and higher values of β increase the number of nodes activated in each pulse p, corresponding thus to higher memory capacity of the reader.
The other two parameters, exclusivity (exc) and popularity (pop) are used to compute the value of the **input function**, which defines how much activation flows into a node from its firing neighbours. Together they define the **accessibility** of the target concept, given the source concept, based on how strong is the semantic relation between them, and on how familiar the target concept is to the reader. Each of the two parameters can be either turned on, or turned off, during the conceptual complexity computation. **Exclusivity** (Hulpuş et al., 2015) measures the strength of the semantic relatedness between two nodes in a knowledge

graph, and has been proved to be particularly effective for computing semantic relatedness (Zhu and Iglesias, 2017). Imagine, for instance, the following example. In a knowledge graph, we have the nodes *John Smith*, *US*, and *Donald Trump*. The nodes *John Smith* and *US* are connected with the relation *is_a_citizen_of*, while the nodes *Donald Trump* and *US* are connected with the relation *is_the_president_of*. In such a scenario, the exclusivity of the relation between the nodes *Donald Trump* and *US* will be much higher than the exclusivity of the relation between the nodes *John Smith* and *US*, as in the full knowledge graph, the relation *is_the_president_of* is much less common than the relation *is_a_citizen_of*. In other words, the stronger the exclusivity of the relation between the two nodes, the higher their semantic relatedness. In terms of conceptual complexity assessment, higher exclusivity of the relation between the two nodes indicates lower level of cognitive effort needed to infer the connection between the two concepts. **Popularity** of a given concept is calculated as the normalized node degree in the knowledge graph. It is used as a proxy for the reader's familiarity with the mentioned concept. In terms of conceptual complexity, the higher the popularity/-familiarity of the concept, the lesser the cognitive effort for the reader to recognize it and put it in the right place when connecting the encountered concepts in a coherent story, the step necessary for fully understanding the text.

To avoid to possibility to enter in a never-ending loop in which, for one mention read in the text, concepts recursively activate each other and activation constantly spreads throughout the knowledge graph, the proposed framework puts a condition that each node/concept can only fire once (at the first pulse in which its activation is over the threshold β). After that, it becomes *burnt* and cannot fire again even if it, during further pulses, accumulates activation higher than β.

2.2. Working Memory Parameters

Before reading starts, the working memory is considered to be empty. During reading, two processes happen simultaneously in the working memory: **bringing the concepts from the long-term memory**, and **forgetting process**.

The concepts activated in the long-term memory through the spreading activation process described above are brought into working memory with an activation value computed according to one of the two possible functions: ϕ^1 or ϕ^A. **The function** ϕ^A brings the activated concepts into working memory with the activation value equal to that obtained through the spreading activation process which happened in the long-term memory. **The function** ϕ^1, in contrast, brings every concept that was activated in the long-term memory, giving it the same activation value of 1, regardless of its activation value in the long-term memory at that point. The function ϕ^1 is a simplified version of the function ϕ^A which only cares about whether the concept is activated in the long-term memory or not, and does not care about the value of its activation. In other words, the function ϕ^1 partially neutralizes the impact of the knowledge graph features, once the concept is brought in the working memory.

The forgetting process is controlled by three parameters:

- **decay at the end of word/token** (γ_w);
- **decay at the end of sentence** (γ_s);
- **decay at the end of paragraph** (γ_s).

The combination of the three forgetting parameters allows to personalization according to 'reading buffer' of each reader. Those parameters allow to separately control how much forgetting happens at the end of each word, each sentence, and each paragraph. For those readers that only have problem with the capacity of their short-term memory, i.e. the amount of the text/words read, the other two parameters (γ_s and γ_p) can be set to 1. For those readers who have problems with the structural organization of the text, but not with the capacity of their short-term memory, the first parameter (γ_w) can be set to 1, and the other two set to lower value depending on how much the sentence organization, or paragraph organization, influences the reader's understanding of a given text.

3. Implementation and Code

Our implementation is designed as an HTTP API having two endpoints: one for text complexity assessment, and another one for comparing the complexity of two given texts. The code, together with data and running instructions, are provided at: `https://github.com/ioanahulpus/cocospa`. The architecture of the system consists of three services: (1) a Redis[4] instance, (2) a DBpedia Spotlight instance, and (3) our conceptual complexity HTTP endpoint. Redis is used as a cache to store the activations under different thresholds for all the entities that are found over the input texts. It makes the processing faster for future inputs. We provide such a cache to be downloaded as a resource from the repository. DBpedia Spotlight, is the entity linker serving a configurable HTTP endpoint with which our complexity calculator communicates. We maintain a separation between the linker and our code, to be able to easily test the conceptual complexity estimator with other entity linkers only by implementing the necessary connector class. Finally, our spreading activation-based conceptual complexity calculator requires the knowledge graph in the form of a DBpedia dump compatible with Neo4j[5] graph platform and a standard dump of DBpedia in HDT[6] format. We provide both files in our release,[7] together with a script that downloads and configures the necessary paths to execute the application services.

Furthermore, we provide docker containers for each service and instructions on how to run the containerized application. We believe that the usage of docker containers facilitates code compilation, distribution and the maintenance of third-party dependencies and therefore it becomes an important mean to reproduce our research over time and on different platforms.

[4] `https://redis.io/`
[5] `https://neo4j.com/`
[6] `http://www.rdfhdt.org/`
[7] `https://github.com/ioanahulpus/cocospa/releases/tag/Data`

3.1. API Endpoints

The HTTP API, together with the swagger documentation of the endpoints, the corresponding input types, and the potential responses run by default on port 8080. We currently support two endpoints: /compare and /complexity which can be tested with our public demo hosted at: `http://demaq3.informatik.uni-mannheim.de:8080/swagger-ui.html`. Both endpoints expect a HTTP POST message with a json payload and the header *'Content-Type: application/json'*. The content of the message uses the following configuration inspired by the long-term memory and working memory parameters:

- **graphDecay -** α
- **firingThreshold -** β
- **useExclusivity** - exc (binary)
- **usePopularity** - pop (binary)
- **tokenDecay** - γ_w
- **sentenceDecay** - γ_s
- **paragraphDecay** - γ_s
- **phiTo1** - ϕ^1 if true; else ϕ^A (binary)

In addition, we added a parameter to control the entity linker recall, returning entities that have a confidence score above **linkerThreshold**, and a parameter to store the text to be analyzed. The following is a cURL example of calling the /complexity endpoint.

```
curl -X POST --header 'Content-Type:
    application/json' -d '
{
  "firingThreshold": 0.01,
  "graphDecay": 0.25,
  "linkerThreshold": 0.35,
  "paragraphDecay": 0.5,
  "phiTo1": true,
  "sentenceDecay": 0.7,
  "tokenDecay": 0.85,
  "useExclusivity": true,
  "usePopularity": true,
  "text": "..."
}' http://demaq3.informatik.uni-mannheim.de
    :8080/complexity
```

For an easy use of the demo, we provide those that achieved best correlations with the Newsela levels (Hulpuș et al., 2019) as the default parameters (see the example above). Therefore, the only required parameter is the text itself. To have an easy method of comparing two texts, we have added an extra endpoint named /compare that takes exactly the same configuration parameters with *text1* and *text2* instead of *text* and returns a comparison json with the result.

3.2. Complexity Assessment Endpoint

The HTTP endpoint for */complexity* assessment responds with a number that represents the complexity score, e.g. for the text *conference in Marseille* the response is: $\{"complexityScore" : 1\}$. The response values range freely above or below 1 with larger values indicating texts that are more complex than others. In some situations, the system cannot compute the complexity score. This happens either because no entities have been found, or because the activation values of the entities did not reach the required threshold. In those cases, the returned value is -1. For example, the text *pay on Amazon*, with default parameters, will produce a -1 complexity score. In this situation, it can be helpful to have a local instance of our service running in order to look at the logs which contain the actual DBpedia Spotlight entities and the activation values at different levels in the text:

```
{   "text":"pay on Amazon",
    "confidence":"0.35",
    "Resources":[{
          "URI":"http://dbpedia.org/resource
              /Amazon.com",
          "surfaceForm":"Amazon",
          "similarityScore":"0.919"
       }]
}
```

For this particular example, if the entity linker threshold is below 0.35, the linker only returns *Amazon.com* as an entity, but that is not enough to assess the complexity score. If the entity linker threshold value is decreased to 0.2 in the request message, DBpedia Spotlight increases the recall and returns an entity linked to surface form *pay*. In our experiments we set the threshold to 0.35 provided as a default by the authors (Daiber et al., 2013). Alternatively, it is possible to directly check the public DBspotlight demo [8] and set the appropriate threshold to observe in greater detail which entities are linked in the text with each threshold.

3.3. Complexity Comparison Endpoint

The second endpoint is an extension of the first and has the sole purpose to facilitate the comparison of two texts. For a request of this type:

```
{
  "linkerThreshold": 0.2,
  "text1": "pay on Amazon",
  "text2": "ride on the Amazon"
}
```

The response is a json that shows the complexity scores of the texts and a message resulting from the comparison. In this instance, because the context is short, the word *Amazon* is linked to the company. Other entities are found related to the surface forms of *pay* (Pay_television) and *ride* (List_of_amusement_rides). Therefore, it is expected that the activation values related to amusement rides and the company are lower than the activation values related to payments and Amazon, making the second text more conceptually complex than the first one.

```
{
  "text1ComplexityScore": 1,
  "text2ComplexityScore": 1.176,
```

[8] `https://www.dbpedia-spotlight.org/demo/`

System	0-1	0-2	0-3	0-4	1-2	1-3	1-4	2-3	2-4	3-4	any
Original (Hulpuș et al., 2019)	.61	.89	.98	1	.92	.97	.99	.90	.97	.88	.91
Current implementation	.65	.84	.94	.96	.83	.93	.95	.83	.90	.80	.86

Table 1: The accuracy of binary classification according to the conceptual complexity (CoCo) scores for ten different level pairs. Comparison of the original implementation with the current one (different entity linker used).

```
  "comparison": "text2 is more complex than
      text1"
}
```

Messages in the comparison field show whether the texts are similar in their conceptual complexity and whether the score could be computed or not according to the spreading activation framework.

In the repository, in addition to the dataset and instructions on how to run the API, we provide some simple python tools that can be used to call the API on arbitrary texts using the command line. The scripts can be used to test different parameters and to reproduce our results on Wikipedia and Newsela corpora (Section 4).

4. Evaluation

To evaluate CoCo, we use one of the tasks described in the original paper (Hulpuș et al., 2019). As the originally used entity linker, KanDis (Hulpuş et al., 2015), is not publicly available, we used DBpedia Spotlight entity linker instead.

4.1. Task

The task consists in predicting the conceptually simpler of the two versions of the same news story, based solely on the value of conceptual complexity score assigned by CoCo. We use the same 200 original news stories from Newsela (Newsela level 0) and, for each of them, the four corresponding simpler versions (Newsela levels 1-4) as those in the original paper (Hulpuș et al., 2019). This results in a total of 1000 texts. Given this is an unsupervised task, it leaves us with 200 test instances for each of the ten Newsela level pairs, e.g. 0-1, 0-2, 0-3, 0-4, 1-2, 1-3. We use the same dataset and evaluation measure (accuracy) as in the original paper (Hulpuș et al., 2019) so that we can directly compare the two implementations that differ only in the choice of the entity linker.

4.2. Entity Linker

In the current implementation, we used DBpedia Spotlight entity linker instead of the originally used KanDis entity linker (Hulpuş et al., 2015) to link the texts to DBpedia, as KanDis is not freely available and DBpedia Spotlight is. In linking the entities/concepts (common nouns and named entities) to DBpedia, KanDis showed comparatively good results at linking both types of entities (disambiguation accuracy between 0.88 and 0.89 on news items). To minimize the effect of wrong linking, the outliers, i.e. entities that have very weak semantic relatedness (Hulpuş et al., 2015) to other entities in the text, are removed.[9] This strategy should eliminate some of the wrongly linked concepts and corner cases, and might be one of the reasons why using KanDis gives better results than using DBpedia Spotlight on our task (Table 1). Another reason might be the lower disambiguation accuracy of the current implementation of DBpedia Spotlight that we use, between 0.72 and 0.85 depending on the version and test corpus (Daiber et al., 2013).

[9]For more details on KanDis performances we refer the reader to the original paper (Hulpuş et al., 2015).

4.3. Results

As the results in Table 1 show, the current implementation that uses DBpedia Spotlight instead of KanDis for entity linking step, on average achieves a 5% lower accuracy on the unsupervised binary task than the original implementation (Hulpuș et al., 2019). However, it achieves a 4% better accuracy on the binary classification between original articles and their first level of simplification (*0-1*). This could, however, also be the effect of the entity linker threshold (currently set to 0.35). We did not do an extensive search for the best entity linker threshold for this specific task. Instead, we used the value provided as default by the authors of the software, and that has been used in literature before (Daiber et al., 2013; Manrique et al., 2018).

Significantly lower accuracy scores in binary classification among the most complex Newsela levels (Levels 0 and 1) should not be taken as a bad sign for our tool. On the contrary, they just reflect the fact that in the first simplification step, human editors mostly apply lexical and syntactic simplification, while conceptual simplification becomes more important at later stages of simplification, as indicated by Hulpuş et al. (2019).

In the pairwise comparison of CoCo values across different Newsela levels, in both implementations, there is a statistically significant difference at a 0.001 significance level (paired t-test).

5. Detection of Conceptual Simplification

In this section, we show how the tool can be used for detecting the presence of conceptual simplification in a corpus.

5.1. Datasets

We use two different simplification corpora: the Newsela corpus (Newsela, 2016) described in more details by Xu et al. (2015), and the English Wikipedia - Simple English Wikipedia (Coster and Kauchak, 2011), described in more details by Coster and Kauchak (2011). Simplification of the news articles in Newsela is aimed at children and language learners, and has the goal of maintaining reader's interest in text. Furthermore, simplification is performed by train human editors under strict quality control. Due to those, texts in Newsela are simplified not only at the lexical and syntactic levels, but also at the conceptual level. Articles

Statistic	Newsela					EW-SEW	
	0	1	2	3	4	Original	Simple
Mean	2.39	2.16	1.79	1.46	1.23	1.25	1.30
Stdev	0.81	0.69	0.58	0.44	0.37	0.75	0.76
Max	8.42	5.88	4.31	4.79	4.12	15.5	15.8

Table 2: The average conceptual complexity (CoCo) scores on Newsela and EW-SEW corpora (lower scores indicate conceptually simpler texts).

Feature	Statistic	Newsela					EW-SEW	
		0	1	2	3	4	Original	Simple
Sentence length	Mean	21.71	18.71	15.69	12.87	10.37	25.19	16.68
	Stdev	3.34	1.61	1.30	1.31	1.36	4.48	4.52
	Max	35.84	24.20	19.91	17.63	13.96	59.0	50.6
Lexical richness	Mean	1027	904	850	731	584	3270	380
	Stdev	364	161	146	132	168	3088	582
	Max	3364	1854	1747	1741	1587	22335	5884

Table 3: The average sentence length and the average lexical richness on Newsela and EW-SEW corpora (lower scores indicate conceptually simpler texts).

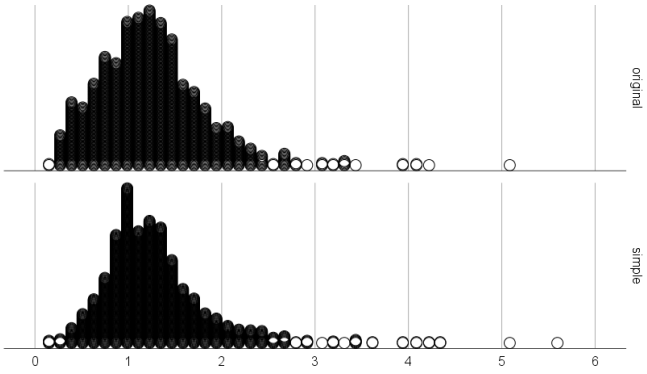
in Simple Wikipedia, in contrast, are not written by trained human editors, do not go through any quality control, and are not aimed at any target population in particular, but rather for everyone. The main Simple English Wikipedia page instructs authors to write articles using simple English words (basic English words) and to write short sentences with simple syntactic structures.[10] It does not mention how the articles should be structured, how complex should be the concepts used, or how difficult the inferences that reader needs to make in order to understand the text should be. In other words, we can expect Simple English Wikipedia to be lexically and syntactically simpler than the original English Wikipedia, but not necessarily conceptually simpler.

After removing articles for which at least one version received -1 for conceptual complexity (due to reasons mentioned in Section 3), we ended up with a total of 1880 original Newsela articles (and, for each of them, the four corresponding simplified versions), and a total of 1504 original Wikipedia articles (and, for each of them, the corresponding Simple English Wikipedia article).

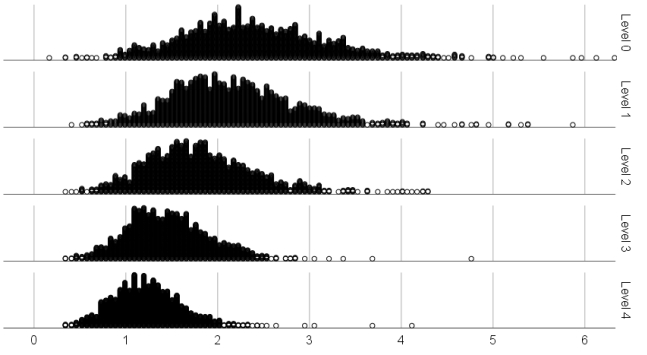
5.2. Results

The statistics of the conceptual complexity computed on both corpora are presented in Table 2. The distributions of the CoCo scores are plotted in Figure 1.

As expected, one can notice the differences in the distribution of conceptual complexity scores across the five Newsela complexity levels in terms of mean value, standard deviation, and maximum observed CoCo value. While the distribution of the CoCo scores is widely spread among the original articles (Level 0), during manual simplification, it becomes more uniform and lower on average. In the English Wikipedia - Simple English Wikipedia (EW-SEW) corpus, in contrast, we do not see any significant differences in the distribution of the conceptual complexity scores across the original Wikipedia vs. Simple English



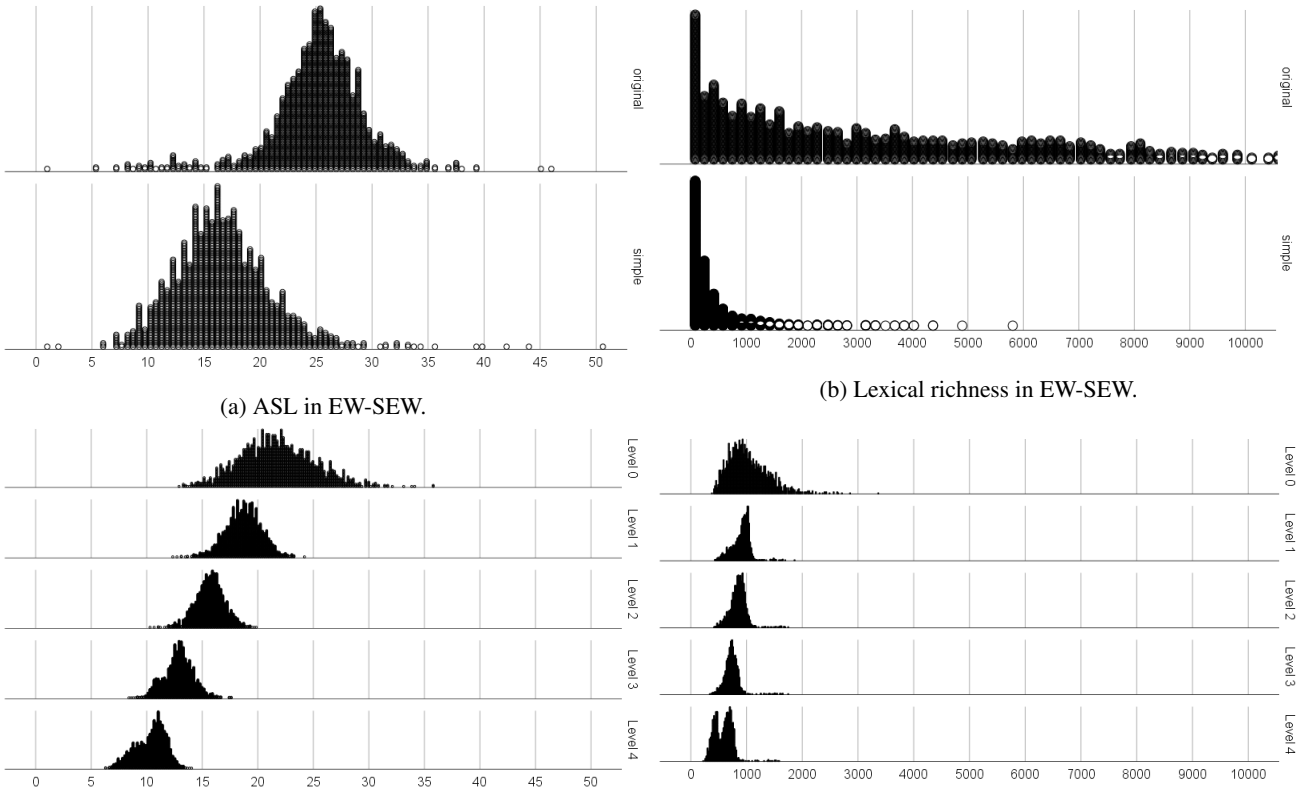
(a) EW-SEW.



(b) Newsela.

Figure 1: Distribution of conceptual complexity (CoCo) scores on Newsela and EW-SEW corpora.

[10] https://simple.wikipedia.org/wiki/Wikipedia:How_to_write_Simple_English_pages



(a) ASL in EW-SEW.

(b) Lexical richness in EW-SEW.

(c) ASL in Newsela.

(d) Lexical richness in Newsela.

Figure 2: Distribution of average sentence length (ASL) and lexical richness on EW-SEW and Newsela corpora.

Wikipedia articles. Those results prove our hypotheses that in Simple English Wikipedia, no conceptual simplification is used, unlike in the case of the Newsela articles.

Furthermore, if we wish to choose the conceptually simpler article in each pair of EW-SEW articles according to their CoCo scores (similar to the experiments performed on the Newsela articles, presented in Table 1), we end up with the accuracy of 0.49. This finding again supports the hypothesis that articles in Simple English Wikipedia are not necessarily conceptually simpler than their counterparts in the original English Wikipedia.

If, in contrast, we compute the statistics for the average sentence length and the average lexical richness (the number of unique lemmas)[11] for the Newsela and Wikipedia articles on different complexity levels (Table 3), we see that both simplification types – the one applied in Newsela, and the one applied in Wikipedia – include sentence shortening (one of the main features of syntactic simplification), and decreasing the number of unique lemmas used (one of the main features of lexical simplification). However, we see that the maximum values for both features, as well as their standard deviations, are significantly higher in Simple English Wikipedia than in any of the Newsela levels (Table 3 and Figure 2), indicating the absence of quality control in simplicity of articles in Simple English Wikipedia, as pointed out by a number of previous studies, e.g. (Amancio and Specia, 2014; Xu et al., 2015; Štajner et al., 2015).

[11]We count the number of types using the WordNet lemmatized form extracted from each POS-tagged token using the PerceptronTagger and the resources available in NLTK (Bird et al., 2009; Fellbaum, 2010). The Wikipedia corpus is already tokenized and split into sentences, therefore the average sentence length on Wikipedia articles is easily extracted. However, on Newsela corpus, we used the punkt sentence spliter and tokenizer available in NLTK to compute the average sentence length.

6. Conclusions and Outlook

Automatic assessment of text complexity plays an important role in education and in making written information accessible to wider populations. While many research and commercial tools have been proposed so far for automatically measuring lexical and syntactic text complexity, no tool has been proposed for automatically assessing conceptual text complexity which is on of the main factors for: (1) maintaining reader's interest in text; (2) improving reader's general knowledge by giving him/her a text on properly adjusted complexity level; (3) better social inclusion of people with various intellectual impairments.

To bridge this gap, we presented CoCo - the first automatic system to measure conceptual complexity of texts. We described the provided hosted demo of the system and instructions on how to call the HTTP API. We also provided the full code to allow for hosting it locally and adjusting it as necessary. We modelled our system architecture using docker containers which facilitates the deployment of the model, evaluation and reproduction of our results over time and across platforms.

We evaluated our CoCo tool and compared it against the state of the art. The results indicated that the performance of the system depends on the quality/choice of the entity

linker. Furthermore, we showed how CoCo can be used to detect whether conceptual simplification is present in a given text simplification corpus or not, taking the example of the two most widely used text simplification corpora, Newsela and English Wikipedia - Simple English Wikipedia (EW-SEW) corpora. As expected, the results indicated that articles in Simple English Wikipedia are not necessarily conceptually simpler than those in the original English Wikipedia, a fact that is corroborated both by previous research and by the instructions provided to the authors of the Simple English Wikipedia.

We hope that, by open sourcing our CoCo tool, we will encourage more research on automatic assessment of conceptual text complexity, a much needed field that has not received much attention from the NLP community so far, probably due to the complexity of the task and the lack of frameworks and open source tools. Furthermore, we envision that high number of parameters and their easy adjustment in the CoCo tool will inspire further research into more personalized automatic assessment of conceptual text complexity.

7. Bibliographical References

Amancio, M. A. and Specia, L. (2014). An Analysis of Crowdsourced Text Simplifications . In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 123–130.

Arfé, B., Mason, L., and Fajardo, I. (2017). Simplifying informational text structure for struggling readers. *Reading and Writing*, Oct.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Bormuth, J. R. (1966). Readability: A new approach. *Reading Research Quarterly*, 1(3):79–132.

Coster, W. and Kauchak, D. (2011). Simple English Wikipedia: a new text simplification task. In *Proceedings of ACL&HLT*, pages 665–669.

Daiber, J., Jakob, M., Hokamp, C., and Mendes, P. N. (2013). Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, pages 121–124, New York, NY, USA. ACM.

DuBay, W. H. (2004). The Principles of Readability. *Impact Information*.

Fellbaum, C. (2010). Wordnet. In Roberto Poli, et al., editors, *Theory and Applications of Ontology: Computer Applications*, pages 231–243. Springer Netherlands.

Flesch, R. (1949). *The art of readable writing*. Harper, New York.

Gunning, R. (1952). *The technique of clear writing*. McGraw-Hill, New York.

Hulpuş, I., Prangnawarat, N., and Hayes, C. (2015). Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In *The Semantic Web - ISWC 2015*, pages 442–457, Cham. Springer International Publishing.

Hulpuș, I., Štajner, S., and Stuckenschmidt, H. (2019). A spreading activation framework for tracking conceptual complexity of texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3878–3887, Florence, Italy, July. Association for Computational Linguistics.

Kincaid, P. J., Fishburne, R. P., Rogers, R. L., and Chissom, B. S. (1975). Derivation of new readability formulas for Navy enlisted personnel. *Research Branch Report 8-75*.

Kintsch, W. and van Dijk, T. A. (1978). Towards a model of text comprehension and production. *Psychological Review*, 85:363–394.

Manrique, R., Grévisse, C., Mariño, O., and Rothkugel, S. (2018). Knowledge graph-based core concept identification in learning resources. In Ryutaro Ichise, et al., editors, *Semantic Technology - 8th Joint International Conference, JIST 2018, Awaji, Japan, November 26-28, 2018, Proceedings*, volume 11341 of *Lecture Notes in Computer Science*, pages 36–51. Springer.

McLaughlin, G. H. (1969). SMOG grading – a new readability formula. *Journal of Reading*, 22:639–646.

McNamara, D. S., Ozuru, Y., Graesser, A., and Louwerse, M. (2006). Validating coh-metrix. In *Proceedings of the Conference of the Cognitive Science Society*, pages 573–578.

Štajner, S. and Hulpuş, I. (2018). Automatic assessment of conceptual text complexity using knowledge graphs. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 318–330, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Štajner, S., Bechara, H., and Saggion, H. (2015). A Deeper Exploration of the Standard PB-SMT Approach to Text Simplification and its Evaluation. In *Proceedings of ACL&IJCNLP (Volume 2: Short Papers)*, pages 823–828.

Xu, W., Callison-Burch, C., and Napoles, C. (2015). Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics (TACL)*, 3:283–297.

Zhu, G. and Iglesias, C. A. (2017). Computing semantic similarity of concepts in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):72–85, Jan.

8. Language Resource References

Coster, William and Kauchak, David. (2011). *English Wikipedia - Simple English Wikipedia (document aligned corpus)*. Freely available at: `https://cs.pomona.edu/~dkauchak/simplification/`, Version 2.0 document-aligned data.

DBpedia. (2014). *DBpedia dump 2014*. Freely available at: `https://wiki.dbpedia.org/`.

DBpediaSpotlight. (2019). *DBpedia Spotlight*. Freely at: `https://dbpedia-spotlight.org`, Version: 2016-01-29.

Newsela. (2016). *Newsela Corpus*. Freely available for research purposes upon request at: `https://newsela.com/data`, Version: 2016-01-29.