LESS is More: LEan Computing for Selective Summaries

Magnus Bender[1][0000−0002−1854−225X], Tanya Braun[2][0000−0003−0282−4284], Ralf Möller[1][0000−0002−1174−3323], and Marcel Gehrke[1][0000−0001−9056−7673]

[1] University of Lübeck, Institute of Information Systems, Ratzeburger Allee 160, 23562 Lübeck, Germany
{bender, moeller, gehrke}@ifis.uni-luebeck.de
[2] University of Münster, Computer Science Department, Einsteinstr. 62, 48155 Münster, Germany
tanya.braun@uni-muenster.de

Abstract. An agent in pursuit of a task may work with a corpus containing text documents. To perform information retrieval on the corpus, the agent may need annotations—additional data associated with the documents. Subjective Content Descriptions (SCDs) provide additional location-specific data for text documents. SCDs can be estimated without additional supervision for any corpus of text documents. However, the estimated SCDs lack meaningful descriptions, i.e., labels consisting of short summaries. Labels are important to identify relevant SCDs and documents by the agent and its users. Therefore, this paper presents LESS, a LEan computing approach for Selective Summaries, which can be used as labels for SCDs. LESS uses word distributions of the SCDs to compute labels. In an evaluation, we compare the labels computed by LESS with labels computed by large language models and show that LESS computes similar labels but requires less data and computational power.

1 Introduction

An agent in pursuit of a task, explicitly or implicitly defined, may work with a corpus of text documents as a reference library. From an agent-theoretic perspective, an agent is a rational, autonomous unit acting in a world fulfilling a defined task, e.g., providing document retrieval services given requests from users. We assume that the corpus represents the context of the task defined by the users of the retrieval service. Further, documents in a given corpus might be associated with additional location-specific data making the content nearby the location explicit by providing descriptions, references, or explanations. We refer to these additional location-specific data as Subjective Content Descriptions (SCDs) [10].

Associating documents with SCDs supports agents in the task of information retrieval. SCDs may cluster similar sentences across the agent's—possibly tiny—corpus [1]. The agent then uses the clusters to answer requests from users by retrieving the sentences in a cluster matching the request. However, a cluster is only a set of similar sentences and it is difficult to describe such set to a human

user in a comprehensible way. For this purpose, a label or a short description helps the agent to present retrieved clusters and its SCDs more comprehensibly to users. Therefore, each SCD needs a label in addition to the referenced similar sentences. The computation of such labels shall require less computational resources and shall work in an unsupervised way even with tiny corpora.

State-of-the-art Large Language Models (LLMs) might be applied to compute the labels. However, LLMs like Bidirectional Encoder Representations from Transformers (BERT) [5] need specialized hardware to run fast and huge amounts of computational resources resulting in high energy consumption [13]. In addition, BERT uses a pre-trained model that must be trained beforehand on large amounts of data while we are interested in an approach working even with less data, e.g., tiny user supplied corpora.

Existing approaches solve the problem of tiny corpora, e.g., by unifying training data for multiple tasks, like translation, summarization, and classification. Together, the unified data is sufficient to train the model and the model becomes multi-modal for solving the different tasks of the training data [6]. However, we are interested in an unsupervised approach requiring no training data at all. In general, a label for an SCD can be estimated by a summarization, i.e., the label is the summarization of a cluster of similar sentences. TED [16] is an unsupervised summarization approach, but it uses a transformer architecture [14] which is the basis for most LLMs. Thus, TED still needs specialized hardware to run fast and huge amounts of computational resources. More lightweight approaches exist for topic models. There, a topic consist of representative words for which a label needs to be computed. However, many approaches need supervision [3, 9, 11].

The main problems with the above approaches are the need for extensive training data and the need to compute on specialized hardware. As a solution, this paper presents LESS, an unsupervised LEan computing algorithm for Selective Summaries. LESS uses the previously mentioned clusters of similar sentences to create selective summaries which then can be used as labels. LESS builds on the Unsupervised Estimator of SCD Matrices (UESM) [1], which provides the clusters of similar sentences needed by LESS. We assume that the concept of each SCD is implicitly defined by the content of the sentences referenced and each label describes the concept of an SCD: So, LESS identifies the best fitting sentence. Together, LESS and UESM associate any corpus with labelled SCDs, where each SCD references similar sentences of the same concept, has a label describing its concept, and an SCD-word distribution. LESS in conjunction with UESM neither needs specialized hardware nor additional training data. In the evaluation, LESS computes labels with less time and computational resources while providing similar results as BERT.

The remainder of this paper is structured as follows: First, we recap the basics of SCDs and UESM. Second, we formalize the problem of computing labels for SCDs and provide our solution LESS. Afterwards, we evaluate the performance of LESS against the well-know BERT and demonstrate that LESS is on par with BERT, while being lean and requiring less resources and no pre-trained models. Finally, we conclude with a summary and short outlook.

2 Preliminaries

This section specifies notations, recaps the basics of SCDs and describes UESM.

2.1 Notations

First, we formalize our setting of a corpus.

- A word w_i is a basic unit of discrete data from a vocabulary $\mathcal{V} = \{w_1, \ldots, w_L\}$, $L \in \mathbb{N}$.
- A sentence s is defined as a sequence of words $s = (w_1, \ldots, w_N)$, $N \in \mathbb{N}$, where each word $w_i \in s$ is an element of vocabulary $\mathcal{V}$. Commonly, a sentence is terminated by punctuation symbols like ".", "!", or "?".
- A document d is defined as a sequence of sentences $d = (s_1^d, ..., s_M^d)$, $M \in \mathbb{N}$.
- A corpus $\mathcal{D}$ represents a set of documents $\{d_1, \ldots, d_D\}$, $D \in \mathbb{N}$.
- An SCD t is a tuple of the SCD's additional data $\mathcal{C}$, i.e., containing the label l of the SCD, and the referenced sentences $\{s_1, ..., s_S\}$, $S \in \mathbb{N}$. Thus, each SCD references sentences in documents of $\mathcal{D}$, while in the opposite direction a sentence is associated with an SCD.
- A sentence associated with an SCD is called SCD window, inspired by a tumbling window moving over the words of a document. Generally, an SCD window might not be equal to a sentence and may be a subsequence of a sentence or the concatenated subsequences of two sentences, too. Even though, in this paper, an SCD window always equals a sentence.
- For a corpus $\mathcal{D}$ there exists a set g called SCD set containing K associated SCDs $g(\mathcal{D}) = \left\{t_j = \left(\mathcal{C}_j, \bigcup_{d \in \mathcal{D}} \{s_1^d, ..., s_S^d\}\right)\right\}_{j=1}^{K}$. Given a document $d \in \mathcal{D}$, the term $g(d)$ refers to the set of SCDs associated with sentences from document d.
- Each word $w_i \in s^d$ is associated with an influence value $I(w_i, s^d)$ representing the relevance of w_i in the sentence s^d. For example, the closer w_i is positioned to the object of the sentence s^d, the higher its corresponding influence value $I(w_i, s^d)$. The influence value is chosen according to the task and might be distributed binomial, linear, or constant.

2.2 Subjective Content Descriptions

SCDs provide additional location-specific data for documents [10]. The data provided by SCDs may be of various types, like additional definitions or links to knowledge graphs. In this paper, we focus on computing labels for SCDs and adding these labels as data to the SCDs. However, before we can compute labels, we need the SCDs themselves.

Kuhr et al. use an SCD-word distribution represented by a matrix when working with SCDs [10]. The SCD-word distribution matrix, in short SCD matrix, can be interpreted as a generative model. A generative model for SCDs is characterized by the assumption that the SCDs generate the words of the documents. We assume that each SCD shows a specific distribution of words of the referenced sentences in the documents.

Before we describe LESS, we outline the details of SCD matrices and UESM, which trains an SCD matrix $\delta(\mathcal{D})$. UESM works unsupervised on a corpus $\mathcal{D}$. In particular, UESM does not require an SCD set $g(\mathcal{D})$ containing initial SCDs.

The SCD matrix $\delta(\mathcal{D})$ models the distributions of words for all SCDs $g(\mathcal{D})$ of a corpus $\mathcal{D}$ and is structured as follows:

$$\delta(\mathcal{D}) = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_K \end{array} \begin{array}{c} \begin{array}{ccccc} w_1 & w_2 & w_3 & \cdots & w_L \end{array} \\ \begin{pmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \cdots & v_{1,L} \\ v_{2,1} & v_{2,2} & v_{2,3} & \cdots & v_{2,L} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{K,1} & v_{K,2} & v_{K,3} & \cdots & v_{K,L} \end{pmatrix} \end{array}$$

The SCD matrix consists of K rows, one for each SCD in $g(\mathcal{D})$. Each row contains the word probability distribution for an SCD. Therefore, the SCD matrix has L columns, one for each word in the vocabulary of the corpus $\mathcal{D}$.

The SCD matrix can be estimated in a supervised manner given the set $g(\mathcal{D})$ for a corpus $\mathcal{D}$. The algorithm used for supervised estimation iterates over each document d in the corpus $\mathcal{D}$ and each document's SCDs. For each associated SCD t, the referenced sentences $s_1^d, ..., s_S^d$ are used to update the SCD matrix. Thereby, the row of the matrix representing SCD t gets incremented for each word in each sentence by each word's influence value.

2.3 Unsupervised Estimator for SCD Matrices

This subsection describes UESM [1], which estimates an SCD matrix $\delta(\mathcal{D})$ without needing the SCD set $g(\mathcal{D})$ of a corpus $\mathcal{D}$. UESM only has a corpus of text documents as input for which the SCD matrix has to be estimated. Commonly, a sentence is associated with an SCD and each SCD references one or multiple sentences. UESM initially starts by associating each sentence to one unique SCD, which leads to an initial SCD matrix consisting of a row for each sentence in the document's corpus. The SCD-word distribution of each SCD then only contains the words of the referenced sentence.

The next step is to find the sentences that represent the same concept and group them into one SCD. There are three different methods to identify similar sentences, namely K-Means [12], greedy similarity, and DBSCAN [7], which need to be chosen depending on the corpus. Each method uses the SCD-word distribution to identify similar sentences, combined with the cosine similarity or Euclidean distance. The SCDs of identified similar sentences form a cluster and are then merged to become one row of the SCD matrix.

Summarized, UESM estimates the SCD matrix $\delta(\mathcal{D})$ for any corpus $\mathcal{D}$. However, the SCDs estimated by UESM miss a label or short description of their represented concept. Next, we present LESS, which computes the missing labels.

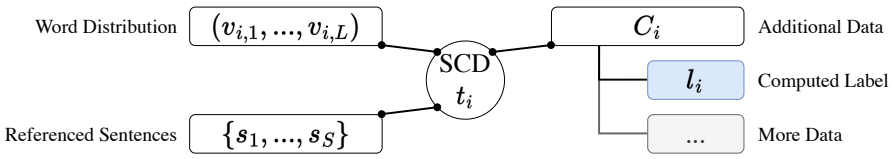

Fig. 1. An SCD t_i with its inter-SCD relations to various parts forming the SCD. LESS uses the word distribution and the referenced sentences to compute labels.

3 Computing Labels for SCDs

Before we introduce how LESS solves the problem of computing an SCD's label, we take a look at the different relations among SCDs for a better understanding of the proposed approach.

3.1 Relations and SCDs

There are two different types of relations among SCDs. First, there are relations between SCDs, e.g., to model complementarity between two SCDs [2]. Second, and faced in this paper, are the relations within an SCD to its various parts, which together form the SCD. The SCD t_i has the SCD-word distribution $(v_{i,1}, ..., v_{i,L})$, the matrix' row, and the referenced sentences $\{s_1, ..., s_S\}$. In this paper, we are interested in the label l_i. The label l_i is part of SCD's additional data $\mathcal{C}_i$, in this paper the only element in $\mathcal{C}_i$. The various parts which form the SCDs together can be seen in Figure 1.

Therefore, our setting is that there are intra-SCD relations to various parts and inter-SCD relations to other SCDs. Additional, relations can be added by storing data or references in $\mathcal{C}_i$, e.g., inter-SCD relations may be added as references to other SCDs [2].

3.2 Labels to Select From

Given that LESS builds upon UESM, the SCD t_i, containing the SCD-word distribution $(v_{i,1}, ..., v_{i,L})$ and the referenced sentences $\{s_1, ..., s_S\}$, is the input LESS has to compute a label. Thus, the label of the SCD needs to be computed only based on these two parts or additional supervision would be needed. In general, a good label could be a short summary given the word distribution of the SCD, since we assume that the word distribution generated the sentences. Therefore, we look for a short sentence, i.e., without many filler words, that is close to the word distribution. We define a *utility* function in the next Subsection 3.3 to measure how well a candidate for a label fits the concept described by an SCD.

The problem to be solved can be formulated as follows:

$$l_i = \underset{l_j \in \text{ all possible labels}}{\arg\max} \; Utility(l_j, t_i = ((v_{i,1}, ..., v_{i,L}), \{s_1, ..., s_S\}))$$

The computed label l_i for t_i (currently consisting of the word distribution and the referenced sentences) is the label with the highest utility. Now, there are two points to address (i) is it not possible to iterate over *all possible labels* and (ii) what is a label with a high utility.

For the first point, we have to specify how a label should look like. A label is a sequence of words like a short description. We argue that a sentence straight to the point, i.e., without many filler words, is a good description and thus a good candidate for a label. Furthermore, each SCD has a set of referenced sentences which together represent the concept which is represented by the SCD. Thus, we use the referenced sentences $\{s_1, ..., s_S\}$ as set of possible labels for each SCD.

Using sentences from the corpus and not generating sentences with, e.g., Generated Pre-Trained Transformer (GPT) [4], has multiple benefits: No pre-trained model or training data is needed while the sentences used as labels will still match the style of writing in the corpus. Additionally, no computational resources for GPT are needed and the sentences must not be checked for erroneous or other troublesome content, e.g., LLMs may degenerate into toxic results even by seemingly innocuous inputs [8].

The problem can now be reformulated as:

$$l_i = \underset{s_j \in \{s_1,...,s_S\}}{\arg\max} \; Utility(s_j, (v_{i,1}, ..., v_{i,L}))$$

The computed label l_i for SCD t_i is the referenced sentence of the SCD which provides the highest utility. The utility function now only gets the word distribution as input because the referenced sentences are already used as set of possible labels. Thus, LESS computes for each sentence its utility given the word distribution and takes the best. Again, an LLM like BERT may be used to calculate the utility. However, we are interested in a lean computing approach.

3.3 Utility of Sentences as Labels

The utility shall describe by a value between 0 and 1 how well a sentences fits the concept described by the SCD. The referenced sentence with the highest utility is assumed to be a good label for the SCD in human interception.

Recall that we assume that each SCD's word distribution generates the referenced sentences, then the best label for this SCD is a sentence that is most similar to the word distribution. Thus, the cosine similarity allows to determine the similarity between two vectors and a word distribution can be interpreted as a word-vector. Thus, we define the utility function as the cosine similarity between the SCD-word distribution and the referenced sentence's word-vector. In addition, the cosine similarity has proven to be a good choice for identifying sentences with similar concepts by using their word distributions: The Most Probably Suited SCD (MPS²CD) algorithm [10] uses the cosine similarity to identify the best SCD for a previously unseen sentence based on its word-vector. Thus, we use the most similar referenced sentence by cosine similarity as the computed label of an SCD.

Algorithm 1 LEan computing for Selective Summaries

1: **function** LESS($\mathcal{D}$)
2: **Input**: Corpus $\mathcal{D}$
3: **Output**: SCD matrix $\delta(\mathcal{D})$; SCD set $g(\mathcal{D})$ containing labels l_i for SCDs t_i
4: $\delta(\mathcal{D}) \leftarrow$ UESM($\mathcal{D}$) ▷ Run UESM [1]
5: $g(\mathcal{D}) \leftarrow \{\}$ ▷ Initialize empty SCD set $g(\mathcal{D})$
6: **for** each row of matrix $i = 1, ..., K$ **do**
7: $v_i \leftarrow g(\mathcal{D})[i]$ ▷ Extract SCD-word distribution
8: $\{s_1, ..., s_S\} \leftarrow$ ReferencedSentences(i) ▷ Get referenced sentences
9: $l_i \leftarrow \arg\max_{s_j \in \{s_1,...,s_S\}} \frac{\vec{s_j} \cdot v_i}{\|\vec{s_j}\|_2 \cdot \|v_i\|_2}$ ▷ Compute label
10: $t_i \leftarrow (\{l_i\}, \{s_1, ..., s_S\})$ ▷ Compose associated SCD with computed label
11: $g(\mathcal{D}) \cup \{t_i\}$ ▷ Add to SCD set
12: **return** $\delta(\mathcal{D})$, $g(\mathcal{D})$

Altogether, the lean computation of a label for an SCD can be formulated as:

$$l_i = \underset{s_j \in \{s_1,...,s_S\}}{\arg\max} \frac{\vec{s_j} \cdot v_i}{\|\vec{s_j}\|_2 \cdot \|v_i\|_2}$$

The word-vector of each referenced sentence is represented by $\vec{s_j}$ and the SCD-word distribution by v_i. In general, the cosine similarity yields results between -1 and 1, in this case all inputs are positive and thus the utilities are between 0 and 1 only.

Finally, the computed sentence to become the label my be slightly post-processed, s.t., it becomes a sentence straight to the point without many filler words. This can be achieved by removing stop words.

3.4 Algorithm LESS

Based on the two previous subsections, LESS is formulated in Algorithm 1. LESS first estimates the SCD matrix using UESM, a step that might be skipped if an SCD matrix is supplied, and initializes an empty SCD set $g(\mathcal{D})$. In Lines 6-11, the label l_i is computed for each of the K SCDs t_i iterating over the rows of the SCD matrix. First, the SCD-word distribution v_i is extracted and also all candidates for the label—the referenced sentences of each SCD—are fetched from the corpus. In Line 9 the label is computed as described in Subsection 3.3. Finally, the associated SCD t_i is composed, containing the additional data C_i including the computed label l_i and the referenced sentences, and added to the SCD set $g(\mathcal{D})$.

Next, we present an evaluation of LESS and compare the results to an approach using BERT for computing labels for SCDs.

4 Evaluation

After we have introduced LESS, we present an evaluation. First, we describe the used corpus, two approaches using BERT to compute labels, and the evaluation metrics. Afterwards, we present the results of the evaluation and show the performance and runtime of LESS in comparison to BERT.

4.1 Dataset

In this evaluation we use the Bürgerliches Gesetzbuch (BGB)[3], the civil code of Germany, in German language as corpus. The BGB does not provide enough training data to train a specialized BERT model. Additionally, it does not provide labels which can be used for supervised training.

Given the vast amount of text written in English and the fact that English is the language of computer science, most natural language processing techniques work better with English language. The BGB is therefore a *difficult* dataset and a good example to use with approaches such as LESS that require only little data and no supervision. To apply BERT on the BGB, we have to rely on external pre-trained models that have been trained on other data.

The BGB is freely available and can be downloaded as XML file. Therefore, it is easily parsable and processable. As the corpus is a law text it consists of correct language, i.e., punctuation and spelling follow the orthographic rules. Thus, less preprocessing and no data cleaning is needed.

The entire corpus consists of 2 466 law paragraphs and overall 11 904 sentences which are used as SCD windows. Each law paragraph contains between 1 and 49 sentences with an average of 4.83 sentences. The vocabulary consist of 5 315 words, where each sentence is between 1 and 35 words long with an average of 7.36 words.

4.2 Approaches using BERT

We evaluate LESS against two approaches using BERT to compute labels for SCDs. Thereby, BERT is compared to LESS in terms of runtime and actual content of the labels.

We use BERT as different utility function to select the best sentence as label from the set of referenced sentence for each SCD. We do not use freely generated texts, e.g., by GPT, as labels because these labels need to be checked for erroneous content as already stated in Subsection 3.2. Additionally, comparing freely generated text to a label selected from a set of referenced sentence is like comparing apples and oranges.

The two approaches using BERT work as follows:

[3] `https://www.gesetze-im-internet.de/bgb/`, English translation `https://www.gesetze-im-internet.de/englisch_bgb/`

BERT Vectors works similar to LESS, but uses the embeddings produced by BERT instead of the word distributions of each sentence. Throughout all referenced sentences an average embedding of each SCD is calculated. The the referenced sentence with the most similar embedding to the the average embedding in terms of cosine similarity is used as label. Thereby, the embedding of the `CLS` token for each sentence from the pre-trained model `bert-base-german-cased`[4] is used.

BERT Q&A uses the ability of BERT to answer a question. Thereby, BERT gets a question and a short text containing the answer. The assumed answer is then highlighted by BERT in the short text. We use the fine-tuned model `bert-multi-english-german-squad2`[5] and compose our answer and short text by concatenating all referenced sentences of each SCD. Hence, BERT highlights a referenced sentence or a part of one while the question consisting of all referenced sentences asks BERT to represent all sentences.

As we do not have supervision for our corpus, we cannot fine-tune BERT models for label computation.

4.3 Hardware and Metrics

LESS and both approaches using BERT run in a Docker container. The evaluation with off-the-shelf hardware is done on a machine featuring 8 Intel 6248 cores at 2.50GHz (up to 3.90GHz) and 16GB RAM, referred as CPU. However, this virtual machine does not provide a graphics card for fast usage of BERT. Thus, all experiments using BERT are run as well on a single NVIDIA A100 40GB graphics card of an NVIDIA DGX A100 320GB, referred as GPU. Beneath, the NVIDIA Container Toolkit is used to run our Docker container with NVIDIA CUDA support.

The runtime of the approaches is measured in seconds needed to compute all labels for the BGB. Thereby, the initialization of the BERT models is excluded but the necessary transformations of the referenced sentences are included. This transformations include the tokenization for BERT and composition of word distributions for LESS. SCDs referencing only a single sentence do not require a computation, and the single sentence is used as label.

The performance is measured by the accuracy between LESS and the results of each of the two BERT-based approaches. Specifically, for one BERT-based approach, all SCDs where BERT and LESS compute the same same label are counted and divided by the total number of SCDs. We distinguish between considering all SCDs, including those referencing only one sentence, or excluding those SCDs with only one referenced sentence.

[4] `https://huggingface.co/bert-base-german-cased`

[5] `https://huggingface.co/deutsche-telekom/bert-multi-english-german-squad2`

4.4 Workflow and Implementation

LESS and the BERT-based approaches are implemented using Python. The implementations use the libraries Gensim[6], NumPy[7], and Huggingface Transformers[8]. LESS is optimized to run on a single core and does not offer multi-core capabilities. BERT uses all available cores or a graphics card.

We evaluate LESS on ten similar SCD matrices. The SCD matrices are estimated by UESM with methods greedy similarity and K-Means. Each method is run with five different hyperparameters. The evaluation workflow for each of the ten matrices follows:

(i) Run UESM and thus estimate the SCD matrix and all SCDs, lacking labels. This step includes fetching the BGB's XML file and running preprocessing tasks [15].
(ii) Run LESS to compute labels for the SCDs. Here, the runtime of LESS is captured.
(iii) Run BERT Q&A and BERT Vectors to compute two more sets of labels for the SCDs. Again, the runtime is captured, separately for each approach and for CPU and GPU.
(iv) Calculate the accuracy of LESS compared to BERT Q&A and BERT Vectors as described in Subsection 4.3. Thereby, use the set of labels computed by each BERT-based approach and LESS. There is practically no difference between the labels computed by BERT on the GPU and on the CPU. Thus, we do not differentiate between CPU and GPU in terms of accuracy.

4.5 Results

In this section, we present the results gained using LESS in comparison to the BERT-based approaches and the previously described workflow.

In the left part of Figure 2, the runtimes of LESS, BERT Q&A, and BERT Vectors are displayed. The values are averaged over all ten evaluated SCD matrices and are shown on a square root scale. Comparing both BERT-based methods, BERT Vectors is always faster than BERT Q&A, especially when running on CPU. LESS is on off-the-shelf hardware as fast as BERT Vectors on the A 100 GPU and always faster then BERT Q&A. The different computational resources needed by BERT and LESS are good to grasp by comparing the runtimes of on the CPU. LESS utilizes only one core while BERT uses eight cores and still BERT is significantly slower. Thus, LESS needs less time on less cores to compute the labels. Typically distilled LLMs represent lean computing, but distilled LLMs still need to run on a GPU to be fast, while LESS remains lean.

In the right part of Figure 2, the accuracies over all ten evaluated SCD matrices are shown with boxplots. We divide between considering all SCDs or only SCDs with more than one referenced sentence. There are no big differences

[6] `https://radimrehurek.com/gensim/`
[7] `https://numpy.org/`
[8] `https://huggingface.co/docs/transformers/`

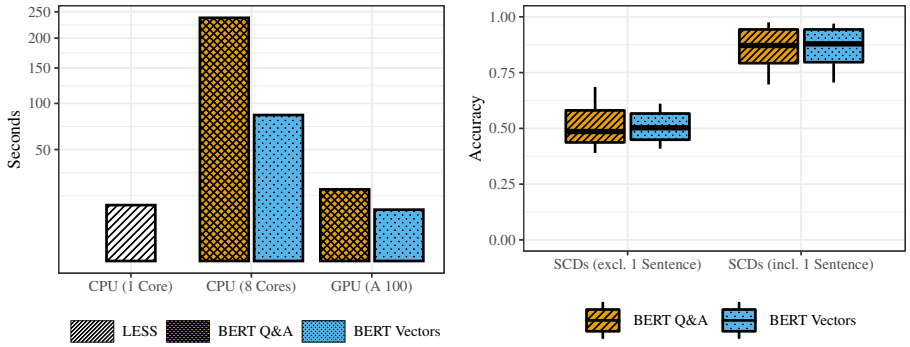

Fig. 2. Left: Runtime of LESS, BERT Q&A, and BERT Vectors with a square root scale. Right: Accuracy of BERT Q&A and BERT Vectors (compared to LESS respectively) divided by considering all SCDs or excluding those with one referenced sentence.

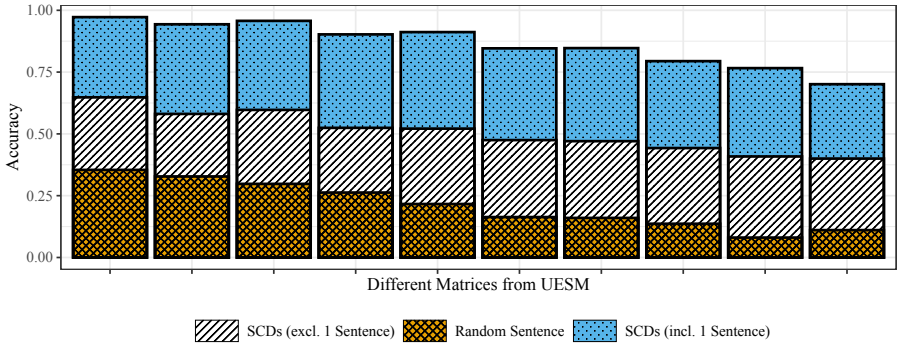

Fig. 3. Theoretical average accuracy of an approach that randomly selects a sentence as label compared to the accuracy of LESS and BERT.

between BERT Q&A and BERT Vectors. Considering all SCDs gives very good accuracies, which is particularly important as we are interested in labels for all SCDs. At first glance, the accuracies considering only SCDs with more than one referenced sentence do not look good but we have to look at the number of the referenced sentences of each SCD.

In order to better rate the accuracies, we calculate the accuracy for random sentence, which is the theoretical accuracy a random approach would result in. This random approach randomly chooses for each SCD which referenced sentence becomes the label. In Figure 3, the accuracy for random sentence is added as baseline to rate the accuracies of BERT and LESS. Besides random sentence, we have the accuracies for SCDs including single sentences and excluding single sentences already known from Figure 2. The three accuracies are displayed for

each of the ten SCD matrices and we do not differentiate between BERT Q&A and BERT Vectors as the values do not show a relevant difference.

For the leftmost bar in Figure 3, randomly selecting a sentence from the clustered sentences results in an accuracy of around 0.35. By excluding SCD with single sentences, LESS and BERT agree 68 % of the time and by including all SCDs nearly all the time (97 %).

Implicitly, the accuracy for random sentence shows the number of referenced sentences the SCDs have. The SCD matrices on the left side of Figure 3 have less referenced sentences per SCD as the ones on the right side. With an increasing number of sentences, there are more sentences to select from and the computation becomes more difficult, i.e., it is more *easy* to correctly select an item from a set of three than of ten items. This increasing difficulty to the right is also demonstrated by the fact that all accuracies become smaller to the right.

The accuracy for random sentence is always clearly the lowest, with both other accuracies following at some distance. Thus, BERT and LESS achieve a high level of accuracy. In summary, LESS computes good labels requiring less resources and time.

5 Conclusion

This paper presents LESS and LESS is more. LESS is an unsupervised lean computing approach to compute labels for SCDs. LESS works on any corpus and does not require training data. LESS only needs clusters of similar sentences, which are contained in SCDs and are estimated in an unsupervised way by UESM. Hence, together with UESM, LESS can generate SCDs with labels for any corpus to help information retrieval agents. We evaluate LESS against two approaches using an LLM, in this case BERT. The evaluation shows that LESS requires significantly less computational resources. Furthermore, LESS does not need any training data. Therefore, we evaluate LESS in a setting, where no training data is available. Hence, we can not fine-tune a BERT model for our needs and evaluate LESS against two approaches using already fine-tuned BERT models. The labels computed by the BERT-based methods significantly coincide with those of LESS. Summarized, LESS computes good labels needing less computational resources.

In this paper, we have proposed a possibility to compute SCDs with labels and their referenced sentences for any corpus without needing additional data. Next, we put efforts in using this computed SCDs to provide an information retrieval service for humans using SCD-based techniques like MPS²CD [10]. Future work will then focus on optimizing the computed SCDs and the labels by updating the matrix incrementally and efficiently based on feedback of the users.

Acknowledgment

The authors thank the AI Lab Lübeck for providing the hardware used in the evaluation.

References

1. Bender, M., Braun, T., Möller, R., Gehrke, M.: Unsupervised estimation of subjective content descriptions. Proceedings of the 17th IEEE International Conference on Semantic Computing (ICSC-23) (2023), https://doi.org/10.1109/ICSC56153.2023.00052
2. Bender, M., Kuhr, F., Braun, T.: To extend or not to extend? enriching a corpus with complementary and related documents. International Journal of Semantic Computing **16**(4), 521–545 (2022). https://doi.org/10.1142/S1793351X2240013X
3. Bhatia, S., Lau, J.H., Baldwin, T.: Automatic labelling of topics with neural embeddings. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. pp. 953–963. The COLING 2016 Organizing Committee, Osaka, Japan (Dec 2016), https://aclanthology.org/C16-1091
4. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020), https://arxiv.org/abs/2005.14165
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019), https://arxiv.org/abs/1810.04805
6. Elnaggar, A., Gebendorfer, C., Glaser, I., Matthes, F.: Multi-task deep learning for legal document translation, summarization and multi-label classification. In: Proceedings of the 2018 Artificial Intelligence and Cloud Computing Conference. p. 9–15. AICCC '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3299819.3299844
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press pp. 226–231 (1996)
8. Gehman, S., Gururangan, S., Sap, M., Choi, Y., Smith, N.A.: RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 3356–3369. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.findings-emnlp.301
9. Hindle, A., Ernst, N.A., Godfrey, M.W., Mylopoulos, J.: Automated topic naming. Empirical Software Engineering **18**(6), 1125–1155 (2013)
10. Kuhr, F., Braun, T., Bender, M., Möller, R.: To Extend or not to Extend? Context-specific Corpus Enrichment. Proceedings of AI 2019: Advances in Artificial Intelligence pp. 357–368 (2019), https://doi.org/10.1007/978-3-030-35288-2_29
11. Lau, J.H., Grieser, K., Newman, D., Baldwin, T.: Automatic labelling of topic models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pp. 1536–1545. Association for Computational Linguistics, Portland, Oregon, USA (Jun 2011), https://aclanthology.org/P11-1154
12. Lloyd, S.: Least squares quantization in pcm. IEEE Transactions on Information Theory **28**(2), 129–137 (1982). https://doi.org/10.1109/TIT.1982.1056489

13. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3645–3650. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-1355
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017), https://arxiv.org/abs/1706.03762
15. Vijayarani, S., Ilamathi, J., Nithya, S.: Preprocessing techniques for text mining - an overview. International Journal of Computer Science & Communication Networks **5**, 7–16 (2015)
16. Yang, Z., Zhu, C., Gmyr, R., Zeng, M., Huang, X., Darve, E.: TED: A pretrained unsupervised summarization model with theme modeling and denoising. In: Findings of the Association for Computational Linguistics: EMNLP 2020. pp. 1865–1874. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.findings-emnlp.168