Noname manuscript No.
(will be inserted by the editor)

Usage of documents embedding for Person Name Disambiguation on the Web

Christophe Lebrun · Kieran Schubert · Biagio Zaffora · Dmitrii Dobrovolskii · Arnaud Gaudinat

Received: March 16, 2022/ Accepted:

Abstract This article introduces a new approach to improve person names' disambiguation on Web based search using document embedding. This new approach drastically simplifies the pre-processing stage, while also outperforming state-of-the-art methods in a monolingual scenario. The addition of a summarization step, even if it comes with a drawback of increasing the computational cost, was able to significantly improve the performance of the whole pipeline. The methods were evaluated on the subset of English pages of the MC4WePS corpus with different metrics: BCubed precision, BCubed recall and F0.5-score. Therefore, the F0.5-score was improved by 5 percent compared to state-of-the-

C. Lebrun
HES-SO/HEG Genève
Rue de la Tambourine 17, 1227, Carouge, Switzerland
ORCID : 0000-0002-8846-6328
E-mail: christophe.lebrun@hesge.ch

K. Schubert
HES-SO/HEG Genève
Rue de la Tambourine 17, 1227, Carouge, Switzerland
ORCID : 0000-0002-0373-5577
E-mail: kieran.schubert@hesge.ch

B. Zaffora
University of Geneva
Research Center for Statistics, GSEM, University of Geneva, 1211, Geneva, Switzerland
ORCID : 0000-0001-7169-823X
E-mail: biagio.zaffora@edu.unige.ch

D. Dobrovolskii
B2B Research Inc.
E-mail: dobrovolski@b2bresearch.com

A. Gaudinat
HES-SO/HEG Genève
Rue de la Tambourine 17, 1227, Carouge, Switzerland
ORCID : 0000-0003-3807-2256
Tel.: +41 22 388 19 06
E-mail: arnaud.gaudinat@hesge.ch

art on the English pages of the MC4WePS corpus. This allowed to take into account a realistic proportion of social media pages in the mix of search engine results.

Keywords Person Name Disambiguation on the Web (PNDW) · Web People Search (WePS) · Clustering · Documents embedding · Summarization · Machine Learning

1 Introduction

In this article we introduce a new strategy to cluster Web pages, referring to the same individual, which are retrieved via a query in a Web search engine using a person's name as an input. The proposed strategy makes use of text summarization and document embedding to improve the performance of state-of-the-art pipelines to group Web pages, which are based on clustering. Clustering individuals on the Web is still a challenging task where pages of same individuals are heterogeneous and name of individual could refer to different individuals (i.e so called homonym, like John Smith). This task is also the second step of a more general task which is called search expertise or expertise retrieval where the first step is to find a list of people with different attributes (blockchain developer in Geneva) and the second is to search individually each people. An overview of the expertise search task and their challenges is fully covered on Balog et. al 2012 [27]. For a comprehensive discussion about the most recent advances in this field see for instance Delgado et al. [1].

Our methodology addresses the so-called Person's Name Disambiguation on the Web (PNDW) challenge, proposed by the Web People Search (WePS) campaigns[1] conducted respectively as a Semeval 1 task in 2007 [2] (WePS-1), as a workshop of the WWW 2009 Conference [3] (WePS-2) and as a workshop of CLEF 2010 Conference [4] (WePS-3). At a high level, the aim of a PNDW task is to correctly cluster Web pages referring to the same referent or entity without knowing the final number of clusters. This implies that we do not know how many individuals with the same name appear as a result of the Web query made using that individual's name.

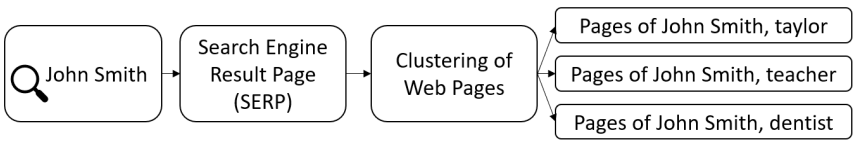

Fig. 1 The People Name Disambiguation on the Web clusters the web pages returned by a Search Engine by the individuals they are referring to.

Once an individual name query is made on a Web search engine and a list of Web pages is returned, a classical PNDW system proceeds in four steps [1]:

1. Pre-processing: several pre-processing steps can be applied to extract plain texts and to remove noisy information including result page fetching, html parsing and cleaning (e.g removing of ads and menus), text extraction, stop words and rare terms removal, stemming and lemmatization.

[1] `http://nlp.uned.es/weps/index.php`, accessed on August 19, 2021.

2. Feature selection and feature engineering: to reduce the computational cost of the process, it is common to extract relevant linguistic features like nouns and verbs as well as non-linguistic features like words or n-grams.
3. Mathematical representation: the Web pages are converted into a vector representation, to allow for the subsequent application of machine learning algorithms.
4. Clustering: a clustering algorithm [6] is applied to the mathematical representation built in the previous step.

In order to achieve a higher clustering performance, our methodology introduces new feature engineering approaches, such as text summarization and document embedding after the pre-processing step. The idea is to reduce the complexity of the original text while retaining its salient concepts via summarization and, at the same time, use document embeddings (doc-2-vec) to get an accurate mathematical representation which can help increase the performance of the clustering algorithm.

After the introduction, we discuss the steps of the new approach in Sec. 2 to handle the task with a focus on text summarization and document embedding. In Sec. 3, we present the methodology of evaluation. We then discuss the major results of our study in Sec. 4 and we summarize our findings and discuss a few directions of improvement in the last section (Sec. 5).

2 Approach

In this section we discuss the new methodology proposed to improve clustering tasks applied to Person's Name Disambiguation on the Web. Fig. 2 shows the main steps of the process which starts with the collection of data (Web pages) associated with a person's name.

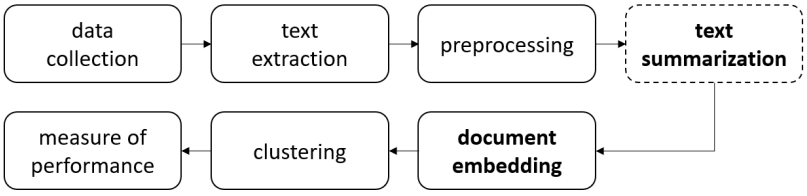

Fig. 2 A novel process for Person's Name Disambiguation on the Web using text summarization, document embedding and clustering. The text summarization step is in a dashed box as it is not compulsory.

After the data collection step, we extract the text contained on the Web pages. The collected text can be pre-processed using classical operations such as stemming and lemmatization, summarized with a dedicated algorithm or embedded directly. A combination of pre-processing actions, text summarization and documents embedding can be used in order to increase the performance of the process for a given corpus. We finally apply classical clustering methods to associate Web pages to the appropriate referent and we measure the performance of the entire process using multiple metrics such as BCubed precision and recall [7].

The following sections describe in detail the steps of the aforementioned process, focusing on the novelties this approach introduces.

2.1 Data collection, text extraction and pre-processing

The first step of the process consists of collecting the Web pages associated with an individual's name and returned by a query of a Web search engine. We scrap the search engine result page, to extract the URLs of the Web pages. Then, we fetch the Web pages and extract the text using dedicated tools such as BoilerPipe[2] and Apache Tika[3]. Since the benchmark used for the evaluation is providing the extracted texts, we skipped this step in this study.

Once the data is collected, we need to decide how to use the information contained in this collection of pages. In Natural Language Processing (NLP) there exists multiple strategies to extract and transform the information contained in a document in order to process it automatically. For instance, we can use techniques such as part-of-speech tagging to extract linguistic features like nouns, verbs, adjectives and adverbs. We can also use other techniques, such as tokenizers, to extract non-linguistic features like words and n-grams. Other features including links, snippets and tags from a Web page can also be retrieved using an html parser.

In our study we focus on the raw text extracted from Web pages and all features used in subsequent steps of the process are created uniquely from this text. Other features, like the ones generated by Web parsers, are not used.

By limiting the pre-processing phase to a very few steps (removal of extra white spaces and removal of the individuals' names that we are attempting to disambiguate), our new methodology is able to sensibly reduce the complexity of the overall process. In fact, the text that is gathered in the text extraction step is immediately fed into the text summarization step which is described next.

[2] `https://github.com/kohlschutter/boilerpipe`, accessed on June 24, 2021.

[3] `https://tika.apache.org/`, accessed on June 24, 2021.

2.2 Text summarization

Text summarization refers to the process of shortening a text to extract its sensible information while preserving the original meaning. There are two main approaches to text summarization: extractive summarization and abstractive summarization. On the one hand, extractive summarization is the task of selecting a sub-sample of items such as words or sentences from the original text, usually by scoring sentences and selecting the ones with the highest score, and concatenating them to generate a summary. On the other hand, abstractive summarization captures salient concepts in the text and generates a summary containing novel sentences, which implies some kind of natural language understanding and natural language generation capabilities in the summarization model. We focus on extractive summarization, since the main goal of the present study is to evaluate the clustering performance on Web pages relating to different individuals and not to generate novel summaries.

In the case of the WEPS task, Web pages referring to individuals can contain irrelevant text or unnecessary HTML boilerplate, which can be referred to as textual noise and don't add value to the identification of persons. Moreover, some Web pages in the corpus contain millions of tokens and are difficult to process. Although a text summarization step is not compulsory before the document embedding step, dealing with noisy and long documents can hinder performance on following tasks. As such, a text summarization algorithm can be beneficial in several ways. It can help to reduce the computational cost associated with embedding a long document by reducing its overall length, in certain cases allowing to circumvent the size limit of the sentences in the embedding step (see next section), such as when using the BERT algorithm [8]. Furthermore, summarization can lead to an increase in the overall performance of clustering by eliminating noise in the text and extracting only the relevant content. In other words, summarization allows to distil the essence of a text while standardizing it's length for further processing.

Over the years multiple algorithms have been presented to summarize texts and documents. In this study, we compare 6 popular extractive summarization algorithms, namely the Luhn summarizer [9], Latent Semantic Analysis [10], LexRank [11], TextRank [12], Edmundson's key-word extraction [13] and BERT-based extractive summarization [14]. The goal of the comparison is to identify the summarization algorithm that improves selected clustering metrics the most (see Sec. 3.1), thus evaluating the summarization algorithm's performance in a downstream task and not with gold standard summaries [17]. In this context, we expect that better summarization will lead to superior document embeddings at a reduced computational cost, which in turn will allow the prediction of optimal clusters associating Web pages to individuals. The text, once summarized, is then ready for the embedding step.

2.3 Document embedding

Aside the use of summarization, another novelty of our approach is the use of document embedding. It replaces more traditional approaches like TF-IDF (a frequency weighted bag-of-words representation of texts), that are widely used in the PNDW task.

Document embeddings are able to capture the semantics of a text and map each document to a vector space where semantically close meanings are also spatially close. For example, two paragraphs with close meanings will be close in the vector space used for their representation. Document embeddings allow to compute semantic similarity between two texts by computing their distance in the vector space, using for example the "cosine distance", the cosine of the angle θ between two vectors A and B:

$$cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}. \quad (1)$$

Among the available embedding techniques, Doc2Vec is an unsupervised learning algorithm proposed by Le and Mikolov [18] that learns vector representations for sequences of words of different lengths such as sentences and documents. The vector representations are learned to predict the words sampled from the paragraph. This method can be seen as an extension of the Word2Vec algorithm [19] to multi-token pieces of texts.

The document embeddings output by the Doc2Vec algorithm are dense, low dimensional and fixed dimensional vectors and are therefore suited to be used as an input to machine learning algorithms, such as clustering algorithms. Document embedding is nowadays widely used and achieves state-of-the-art performance in many different natural language processing (NLP) tasks, such as the Forum Question Duplication task and Semantic Textual Similarity (STS) task [28].

Previous work by Duque et al. [20] focuses on a naive sentence embedding approach by averaging word embeddings to compute a vector representation of the document. But Reimers et al. [21] have shown that averaging word embeddings produces poor results for NLP tasks. As such, we decided to use the Sentence-BERT (sBERT) model, which is one implementation of the Doc2Vec algorithm. More precisely, Sentence-BERT [21] is a modification of the pretrained BERT network [8] [14] built upon siamese and triplet network structures to create vector representations of sentences. For our study, we used the out-of-the-box sBERT model pretrained on the English Wikipedia corpus (2.5B words) and BooksCorpus (800M words) datasets [8].

2.4 Clustering

When using clustering in the context of PNDW, the goal is to group together items (for instance Web pages) based on a similarity measure, meaning that we aim at putting within the same cluster items that are most similar to each

other. Some techniques, including the so-called K-means algorithm, require the number of clusters to be defined at the beginning of the process. For PNDW tasks however, the exact number of clusters is almost always unknown and K-means or other similar algorithms are not appropriate. For this reason, the hierarchical agglomerative clustering (HAC) has acquired prominence over the years in the PNDW field [22]. We propose to use the HAC technique directly on the embedded documents.

The idea of hierarchical clustering is to define a measure of dissimilarity between disjoint groups of observations, based on the pairwise dissimilarity of the observations in each group. This method generates a hierarchical representation in which the clusters at each level are created by merging clusters at the next lower level. At the lowest level each cluster is a single observation. At the highest level all the observations belong to only one cluster. In HAC, the algorithm starts at the lowest level and, at each recursion, the two most similar clusters (or the two least dissimilar clusters) are grouped together using a bottom-up approach. It is up to the user to decide what is the most appropriate level of aggregation [6], using the HAC threshold hyper-parameter.

In order to define a dissimilarity measure, let us indicate with G and H two groups of observations. The dissimilarity $d(G, H)$ between the groups G and H is computed from the pairwise dissimilarities $d_{i,i'}$ where $i \in G$ and $i' \in H$. A first dissimilarity measure is called single linkage (SL) and quantifies the dissimilarity between G and H as the lowest dissimilarity between pairs of observations in the two groups:

$$d_{SL}(G, H) = \min_{\substack{i \in G \\ i' \in H}} d_{ii'}. \tag{2}$$

Complete linkage (CL) defines intergroup dissimilarity as the dissimilarity between the furthest pair:

$$d_{CL}(G, H) = \max_{\substack{i \in G \\ i' \in H}} d_{ii'}. \tag{3}$$

Another common dissimilarity measure is the group average (GA) measure which is defined as the average dissimilarity between all observations in the groups:

$$d_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'}. \tag{4}$$

where N_G and N_H are the number of observations in the groups G and H respectively.

Other methods for clustering have been proposed (see for instance the use of binary classification with custom dissimilarity measures in Delgado et al. [22]) as well as other dissimilarity measures (see reference [6] for a detailed treatment of clustering) but are not considered in the present study. Following Delgado et al. [1] we decided to use single linkage as a dissimilarity measure.

The implementation of HAC requires a threshold allowing to select the granularity of the clustering that determines the number of final clusters. This threshold is a hyper-parameter that can be chosen in an optimal way

by minimizing an error measure in a test set using classical machine learning procedures. For our study we used the MC4WePS corpus [23] (discussed in the next section) where the training set is comprised of Web pages associated with 65 individuals, and the test set is comprised of Web pages associated with 35 individuals. The hyper-parameter selection is discussed in greater detail in Sec. 4.

3 Evaluation

3.1 Benchmarks

To test the performance of a clustering algorithm applied to the PNDW scenario, we need standards in which Web pages are correctly labeled, meaning that each of them is correctly associated with a referent individual. However, different people may have the same name and the task can quickly become complex.

As already discussed, multiple campaigns have been organized with the aim of testing clustering capabilities for a person's name disambiguation. In the context of the reference campaigns WePS-1 [2], WePS-2 [3] and WePS-3 [4], gold standard datasets were provided to the participants and are often used to test the performance of new clustering strategies.

Another set of benchmark data, referred to as UvA [24], was compiled by researchers at the University of Amsterdam and is focused mainly on social media pages. As a consequence, social media pages are over-represented in this data set.

In 2017 Montalvo et al. [23] introduced a new benchmark called Multilingual Corpus for Web People Search (MC4WePS) that addresses the limitations of the previous corpora, including the lack of social media pages in the WePS campaigns corpora as well as the over-representation of social media pages in the UvA data set. The introduction of this new gold standard allowed also to account for multilingualism, which was missing in all previous data references.

In the present article, we adapted the MC4WePS gold standard data to filter out pages which are not in English using Apache Tika[4]. The reason is that even in a monolingual scenario, the PNDW task is still an open research question with state-of-the-art results being still too low for robust applications. This adaptation of the MC4WePS corpus allowed us to evaluate results in a monolingual scenario, while including social media Web pages. The corpus includes Web pages related to 100 individuals of which 34 have an English name. Overall, 62% of the Web pages retrieved are in English and it is this subset that we measure the performance of our methodology on according to the metrics that we describe next.

[4] `https://tika.apache.org/`, accessed on June 24, 2021.

3.2 Metrics

Once a clustering algorithm is selected, we can apply it to the embedded documents according to the flow in Fig. 2 and are ready to evaluate its performance. According to Amigó et al. [7], the most fit-to-the-purpose metrics to evaluate the performance of clustering algorithms are the so-called BCubed precision and recall, first introduced by Bagga and Baldwin [25].

Cubed metrics decompose the evaluation process estimating the precision and recall associated to each item. The item precision represents how many items in the same cluster belong to its category. The recall associated to one item represents how many items from its category appear in its cluster. The overall BCubed precision is the averaged precision of all items in the distribution. Since the average is calculated over items, it is not necessary to apply any weighting according to the size of clusters or categories. The BCubed recall is defined in an analogous way, by simply replacing "precision" with "recall" [7].

In addition to the previous definitions of precision and recall, those metrics are often combined in what is called an Fβ metric allowing for weighing more on either metrics. Mathematically, the Fβ-measure is provided by [26]:

$$F\beta\text{-measure} = \frac{(1+\beta^2)\cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}. \tag{5}$$

The β parameter can be interpreted as a factor such that recall is considered β times as important as precision. In this article, we compare the performance of the process using all the metrics introduced here and setting β to 0.5.

4 Analysis of results and discussion

4.1 Results with documents embedding (without summarization)

4.1.1 HAC threshold selection

Aside the weights of the neural network of the document embedding model, the only hyper-parameter of our approach is the *distance threshold* of the *hierarchical agglomerative clustering* (HAC) algorithm, which is the linkage distance threshold above which clusters will not be merged, as presented in Sec. 2.4.

To select the value of this threshold, we optimised the F0.5 score on the training set of the MC4WePS benchmark (restricted to English pages), using the *grid search* method. On this data-set, we observed that optimal interval for the threshold was [0.23 ; 0.25] (Fig. 3). The smooth shape of the curve is giving confidence in the stability and robustness of the method.

We then proceeded to the final evaluation of our approach on the testing set of the MC4WePS benchmark (restricted to English pages). For the value of the threshold, we selected 0.24, the central value of the optimal interval. We measured a F0.5 score of 0.74 on this testing set. This score, even better than

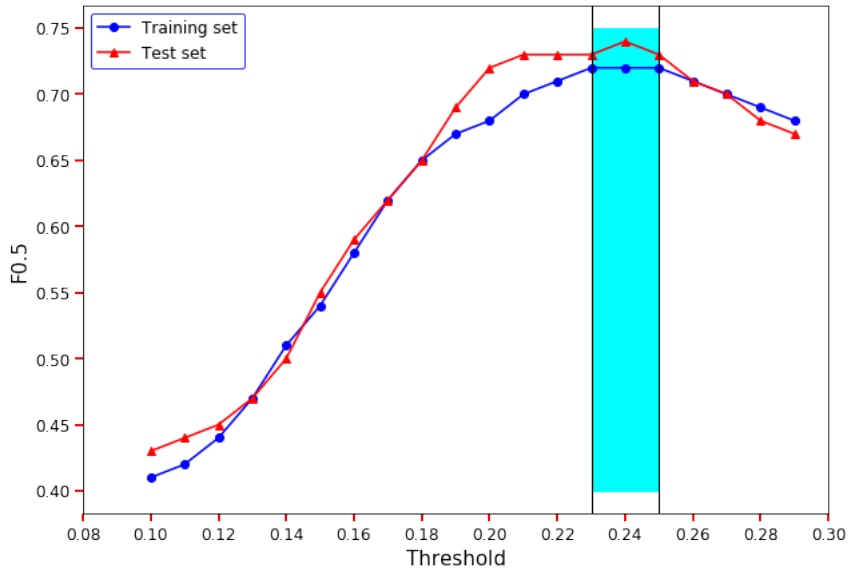

Fig. 3 Threshold selection for the HAC algorithm using training and test data from the MC4WePS data set. The shaded area indicates the selection region for the threshold.

the one obtained on the training set, confirms that there is no over-fitting, but can also indicate that the training set is not totally statistically representative of the testing set, which seems to be a little less challenging for the PNDW task.

4.1.2 Comparison to state of the art method (ATC2018)

Table 1 Comparison of results between models

System	BP	BR	F0.5
HAC_0.24	0.71	0.84	0.74
ATC2018	0.73	0.78	0.72
AllInOne	0.52	1.0	0.61
OneInOne	1.0	0.32	0.42

The new method represents a slight improvement in F0.5 score compared the *Adaptative Threshold Clustering* (ATC) method of 2018, improving its value by 2 percents (0.74 vs 0.72). In detail, the new approach improves the b-cubed recall by 6 percents (0.84 vs 0.78), at the cost of a slight decrease of the b-cubed precision by 2 percents (0.71 vs 0.73).

Table 2 Comparison of results between optimal summarization models using HAC (0.24)

	BP	**BR**	**F0.5**	**Sentences**	**Text Truncation**
Keyword	0.77	0.8	0.77	29	Yes
Luhn	0.74	0.79	0.75	2	Yes
LSA	0.73	0.82	0.75	9	Yes
LexRank	0.76	0.78	0.75	6	Yes
TextRank	0.73	0.8	0.75	2	Yes
BERT	0.78	0.75	0.75	4	No
HAC_0.24	0.71	0.84	0.74	-	Yes

4.2 Results with summarization and documents embedding

The following results show the influence of the summarization algorithms presented in Sec. 2.2 on the clustering of Web pages measured by b-cubed precision, b-cubed recall and F0.5 score in the MC4WePS dataset. Furthermore, we investigate how the best summarization methods scale with document size and implement two variants to speed up computations, a greedy summarization algorithm and its threaded version.

First, we optimize the summarization hyper-parameter through *grid search* (number of sentences to include in a summary or number of keywords in the case of Edmunson's keyword extractor) on the test set for each summarization model based on F0.5 score. We evaluate summarization on two versions of text with pre-processing to remove white-spaces : one with a character cutoff of 10'000 characters and another with full length text. We then compare the models with the optimal hyper-parameter and find that *Edmunson's Keyword Extractor* with 29 keywords and a character cutoff improves clustering performance the most, as reported in Table 2. We note that including a character cutoff generally improves results across summarization models compared to using full texts while also allowing faster processing. The addition of the best performing summarizer in the clustering pipeline increases F0.5 score by 4.1 percents compared to the baseline HAC model (0.77 vs 0.74). Furthermore, Figure 4 illustrates how the number of sentences or keywords (depending on the summarization algorithm) influence processing time. We see that hyper-parameter choice doesn't seem to influence processing time, except for the BERT extractive summarizer model. It is interesting to note that the simpler models outperform the much more recent BERT extractive summarizer on this task at a reduced computational cost. Our results indicate that most summarization models are able to improve clustering results on this data-set but are between 7 to 20 times slower on average than the HAC baseline model.

The inclusion of a summarization component to the pipeline has two main goals: remove textual noise to improve clustering and reduce the cost of text embedding by using shorter text summaries. Even though the first objective is reached, the cost of summarizing texts is quite high and doesn't compensate the speed gain at the embedding step. Indeed, all summarization techniques require significant additional resources, especially when dealing with documents above a million characters in length. Since most of the summarization

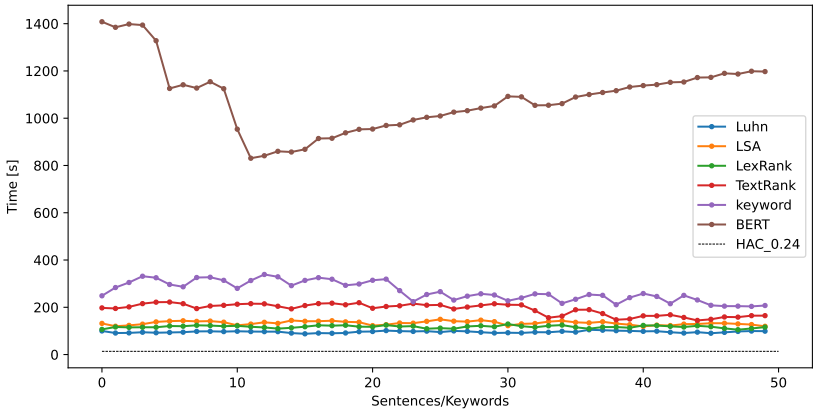

Fig. 4 Summarization timing with different hyperparameters on the truncated MC4WePS test data set.

techniques presented in this paper rely on matrix computations, the quadratic increase in processing time becomes quite problematic when dealing with long and numerous documents, which is quite frequent when processing real documents from the Web. Although most documents of the MC4WEPS corpus where below a million characters in length, a single long document can significantly hinder performance and it is as such of interest to obtain a fast and reliable textual processing pipeline.

We then investigate how to decrease the computational cost of including a summarizer in our pipeline by implementing a greedy version of the optimal *Edmunson Keyword Extractor* as well as a threaded version of this greedy algorithm. The greedy algorithm will truncate a text into chunks of a specified size, then summarize each chunk individually and concatenate them to produce a summary. If the resulting summary is still longer than a given threshold, the greedy summarization is repeated until the final summary length is below the threshold. Results indicate that the threaded implementation is on average slower than the greedy algorithm, and that the 47.4 percent increase in speed provided by the greedy algorithm over the base summarizer is accompanied by an important decrease in F0.5 score (0.63 vs 0.77).

4.3 Limitations and possible improvements

In our approach, we used a standard pre-trained sBERT model for documents embedding. It could be an improvement to fine-tune this model on a corpus of documents from the domain (content of web pages related to individuals). It could also be worthy to test some other document embedding models to see which ones performs best.

Another interesting improvement could be the use of pairwise binary classification (for example using a neural network) instead of traditional clustering with HAC.

It is worth mentioning the modified BERT architecture for extractive summarization developed in [15] and [16]. Although not included in this analysis, it would be interesting to evaluate its performance in further work.

5 Conclusions

We have used *documents embedding* of the textual content of web pages to get a suitable representation of these documents for further use of clustering algorithms. This approach has shown improved results, compared to state of the art algorithms for the PNDW task. A very convenient side-effect is the simplification of the pre-processing step. Indeed, the text extracted from the web pages require only very light processing: removal of extra spaces and a person's name to disambiguate.

Moreover, we have shown that the use of an additional summarization step, just before the document embedding step, improved significantly the results of the whole pipeline for the task. This improvement in performance came at the cost of increased computational cost. This approach of adding a summarization step could therefore be extended to all other offline tasks using documents embedding and where performance is more important than the computational cost.

If the PNDW task remains unsolved, as the achieved performance by automated methods is still too low to allow any industrial use, we think that a disruptive approach is required to reach drastic improvements. More specifically, we believe that replacing the clustering algorithm with a binary classification algorithm as explored by ATC 2018 could both simplify the process and improve greatly the performance.

Acknowledgements – HOSTKEY.COM (GPU dedicated server provider) for GPU computing power under the grant program for research and startups
– Innosuisse Program
– b2bresearch

Conflict of interest

The authors declare that they have no conflict of interest.

Appendix A: Summarization Algorithm Implementation

All summarization algorithms are implemented in Python and are obtained from open source packages. The Luhn, Edmunson, Latent Semantic Analysis,

LexRank and TextRank summarizers used in this study come from the `sumy`[5] Python package, and the BERT extractive summarizer is made available in [14]. The greedy summarization algorithm is based upon a threshold (T). If the number of characters in the input text (N) is below the threshold, the text is summarized normally. If the text is longer than the threshold, the additional text is recursively split and summarized. The final summaries are then concatenated and summarized one last time. The optimal threshold was selected through *grid search* as the one that sped up computations the most.

Algorithm 1 Greedy Summarization Algorithm

Require: $T > 0$
  $N \leftarrow len(text)$
  if $N < T$ **then**
    $summary \leftarrow Summarize(text)$
  else
    $summary \leftarrow Summarize(text[:T]) + GreedySummarize(text[T:], T)$
    $summary \leftarrow GreedySummarize(summary, T)$
  end if
    return *summary*

Appendix B: Technical Specifications

- OS: Ubuntu 20.04.2 LTS (Focal Fossa) - CPU: 2x 26-Core Intel Xeon Gold 6230R - GPU: 4x NVIDIA GV100GL [Tesla V100 SXM2 32GB] - RAM: 12x 64GiB DIMM DDR4 Synchronous Registered (Buffered) 3200 MHz [768GB] - Disk: 94TB PERC H840 Adp

References

1. A. D. Delgado, S. Montalvo, R. M. Unanue, and V. Fresno, "A survey of person name disambiguation on the web". IEEE Access, vol. 6, pp. 59496-59514 (2018).
2. J. Artiles, J. Gonzalo, and S. Sekine, "The SemEval-2007 WePS evaluation : establishing a benchmark for the Web people search task", Proceedings of the 4th International Workshop on Semantic Evaluation, pp. 64-69 (2007).
3. J. Artiles, J. Gonzalo, and S. Sekine, "WePS 2 evaluation campaign : overview of the Web people search clustering task", Proceedings of the 2nd Web People Search Evaluation Workshop (WePS), pp. 1-9 (200).
4. J. Artiles, A. Borthwick, J. Gonzalo, S. Sekine, and E. Amigo, "WePS 3 evaluation campaign : overview of the Web people search clustering and attribute extraction tasks", Proceedings of the 3rd Web People Search Evaluation Forum, pp. 1-9 (2010).
5. H.-H. Huang, and Y.-H. Kuo, "Cross-lingual document representation and semantic similarity measure : a fuzzy set and rough set based approach", IEEE Transaction Fuzzy Systems, vol. 18, n. 6, pp. 1098-1111 (2010)
6. T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning. Data Mining, Inference and Prediction. 2nd Edition", Springer, New York, pp. 501-528, (2009).

[5] https://github.com/miso-belica/sumy

7. E. Amigó, J. Gonzalo, J. Artiles, F. Verdejo, “A comparison of extrinsic clustering evaluation metrics based on formal constraints”. Information Retrieval, vol. 12, pp- 461–486 (2009).
8. J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, https://arxiv.org/abs/1810.04805 (2018).
9. H. P. Luhn, “The Automatic Creation of Literature Abstracts”. IBM Journal of Research and Development, vol. 2, no. 2, pp. 159-165 (1958).
10. J. Steinberger, K. Jezek, ”Using latent semantic analysis in text summarization and summary evaluation”. Proc. ISIM 4 (2004): 93-100.
11. G. Erkan, and D.R. Radev, “Lexrank: Graph-Based Lexical Centrality as Salience in Text Summarization”. Journal of Artificial Intelligence Research, vol. 22, pp. 457-479 (2004).
12. R. Mihalcea, and P. Tarau, “TextRank: Bringing Order into Texts”. Proceedings of EMNLP-04 and the 2004 Conference on Empirical Methods in Natural Language Processing (2004).
13. H.P. Edmundson, “New Methods in Automatic Extracting”. Journal of the ACM, vol. 16, no. 2, pp. 264-285 (1969).
14. D. Miller, “Leveraging BERT for Extractive Text Summarization on Lectures”, https://arxiv.org/abs/1906.04165 (2019).
15. Liu, Yang, and Mirella Lapata.“Text summarization with pretrained encoders.” arXiv preprint arXiv:1908.08345 (2019).
16. Liu, Yang.“Fine-tune BERT for extractive summarization.”, arXiv preprint arXiv:1903.10318 (2019).
17. Y. Dong,“A survey on neural network-based summarization methods.” arXiv preprint arXiv:1804.04589 (2018).
18. Q. Le, T. Mikolov, “Distributed Representations of Sentences and Documents”. Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32 (2014).
19. T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient Estimation of Word Representations in Vector Space”, https://arxiv.org/abs/1301.3781 (2013).
20. A. Duque, L. Araujo, and J. Martinez-Romo, “LSI-UNED at M-WePNaD: Embeddings for Person Name Disambiguation”. Proceedings of the Second Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2017).
21. N. Reimers, I.Gurevych, ”Sentence-bert: Sentence embeddings using siamese bert-networks.” arXiv preprint arXiv:1908.10084 (2019).
22. A. D. Delgado, R. Martinez, V. Fresno, S. Montalvo, “A Data Driven Approach for Person Name Disambiguation in Web Search Results”. Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 301–310, Dublin, Ireland, August 23-29 (2014).
23. S. Montalvo, R. Martínez, L. Campillos, A. D. Delgado, V. Fresno, and F. Verdejo, “MC4WEPS: a multilingual corpus for Web people search disambiguation”, Language Resources and Evaluation, vol. 51, pp. 805–832 (2017).
24. R. Berendsen, B. Kovachev, E.-P. Nastou, M. de Rijke, and W. Weerkamp, “Result disambiguation in Web people search”, Proceedings of the 34th European Conference in Information Retrieval (ECIR), pp. 146-157 (2012)
25. A. Bagga, B. Baldwin, “Entity-based cross-document coreferencing using the vector space model”. Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL), vol. 1, pp. 79–85 (1998).
26. C. J. Van Rijsbergen, “Information Retrieval (2nd ed.)”, Butterworth-Heinemann (1979).
27. Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, “Expertise Retrieval”, Foundations and Trends in Information Retrieval Volume 6 Issue 2 (2012).
28. Jey Han Lau,Timothy Baldwin, “An Empirical Evaluation of doc2vec”, Proceedings of the 1st Workshop on Representation Learning for NLP, pages 78–86 (2016).