The Raymond and Beverly Sackler Faculty of Exact Sciences

The Blavatnik School of Computer Science

Finding Inter-textual Relations in Historical Texts

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science

by

Daniel Labenski

The research for this thesis has been carried out at Tel Aviv University under the supervision of Professor Nachum Dershowitz

August 2016

Abstract

Historical texts have been undergoing large-scale digitization in recent years, which provides scholarly and nonprofessional communities with simple and fast access to cultural heritage documents. It also offers scholars new opportunities for exploring these resources and using text mining techniques to find hidden patterns that otherwise would be like finding a needle in a haystack.

In this thesis, we deal with two text mining tasks that can help with finding related texts within large corpora. The first is finding newspaper articles related by topic (covering the same event), and the second is finding text fragments that are related by having shared compositional origins (reused texts). For article-topic detection, we used a common vector space model to represent the articles. Despite the very noisy text (a result of poor optical character recognition), there was a high rate of success in finding the relevant similar article. We then obtained a 3% improvement by adding words that have similar distributional word representations.

For the text-reuse problem, we adapted an algorithm designed for finding all approximate matches in DNA subsequences. We modified the method for natural language by changing it to work with syllables as the basic building block, rather than individual characters. This improved both runtime and quality of results very significantly. We also suggest a new ranking method for match candidates that scores syllable matches and mismatches based on their importance. This improved the relevancy of the highest ranked matches.

Table of Contents

Acknowledgments

This research was supported in part by Grant #I-145-101.3-2013 from the German-Israeli Foundation for Scientific Research and Development (GIF), Grant #01019841 from the Deutsch-Israelische Projektkooperation (DIP), Grant 1330/14 of the Israel Science Foundation (ISF), and a grant from the Blavatnik Family Fund.

My special and warm thanks go to Professor Nachum Dershowitz for his continued guidance during the various stages of this project and his always forthcoming support, help and advice in all the stages of this research.

I would like to thank Professor Yaron Tsur, the Jewish Historical Press Project and the National Library for giving me access to the JPRESS archive data and giving me the support needed.

I would like to thank Dr. Orna Almogi for giving me the access to the Indo-Tibetan corpora and teaching me the principles of the Tibetan language and culture.

I would also like to thank Lena Dankin, Adi Silberpfennig and Elad Shaked for their contribution to our joint work and all their help along the way.

Finally, I would like to express my love and appreciation to my spouse Daria for her patience, her encouragement and for supporting me in the tougher moments.

1 Introduction

In recent years, many historical books, manuscripts and physical media have been transformed into digital representations, opening doors to new opportunities, like distant reading and gaining insights from the data.[1] These possibilities were not available by manual and traditional techniques, however recently developed computational tools enable the researchers to explore the data more extensively than before. In this thesis, we present two textual mining tasks that we worked upon.

The first task we tackled is topic detection and alignment of news articles in historical newspaper archive that were digitized using (noisy) Optical Character Recognition (OCR).

Topic Categorization and Topic Detection are two related but not similar tasks. Topic Categorization goal is to classify a document into one of predefined general topics , whereas in Topic Detection we wish to identify a finer relation, that is, articles that discuss the same event.

Finding such related articles in historical newspaper archives can help scholars to easily explore historical stories from a higher perspective, can help visitors of the archives' websites by having fast navigation – like in modern news websites – and can help compare coverage between different publications.

We started by approaching the problem with a traditional method and enhanced it by using distributional representation of words, also called "word embedding", which has gained popularity in the last few years. Many studies from the last few years report that word embedding encode semantic and syntactic features of the word. We use the fact that similar words are likely to have similar vectors in order to augment the documents' representation with similar words. This helps solve the sparsity of document vector problem, due to language variation, morphology and the OCR noise. Using such an augmentation proved to be helpful in finding more related articles and approved to be even better than language-specific standardization methods.

The second task we present in this work is the detection historical text reuse. More specifically, our goal is to locate parallel passages within Tibetan texts (regardless of any attribution). Our main goal is detecting inexact quotations within two corpora (exact matches are rather easy to identify

[1] `http://www.nytimes.com/2011/06/26/books/review/the-mechanic-muse-what-is-distant-reading.html`.

using standard plagiarism detection methods). Also, we do not aim to detect text regions with topic and semantic similarity (as these can be identified using standard bag-of-words and topic-modeling vectors).

In the Tibetan-Buddhist corpora, as in most historical texts, shared passages are not necessarily reproduced verbatim, but are often paraphrased or shortened, or both. In addition, orthographic differences or omission/addition of grammatical particles (both are very common in Tibetan) are of no great significance for this particular task.

We adapt a method designed for approximate matching DNA subsequences to our problem and post-process its results to provide scholars with optimal passage matches, ranked by apparent importance. The initial results seem to have very good precision in finding relevant parallel passages.

By finding cited or borrowed passages within the corpora of Buddhist Indo-Tibetan (i.e. translated) and Tibetan (i.e. autochthonous) literature, the following research questions can be better addressed:

1. Determining the history of composition of individual texts.

2. Determining relative chronology of groups of texts.

3. Determining the intellectual scholarly milieu in which the texts emerged.

4. Determining the intellectual history behind the texts (viz. terminology and concepts).

After identifying parallel passages, one can assess the frequencies of letter/syllable/word replacements in the aligned passages of selected texts or text groups. This can serve to help answer research questions:

1. Determining editorial policies and processes, such as standardization of orthography.

2. Standardization of employment of grammatical particles (i.e. according to the so-called *sandhi* rules).

3. Identifying processes of "revisions" of translated texts.

This thesis is organized as follows: Chapter 2 describes the historical datasets we use in this paper and previous work done in the domain of the problems. Chapter 3 describes a comparison of the methods we developed for finding related newspaper articles. In Chapter 4, we present our most recent efforts for finding inexact quotation in the Tibetan corpus. We summarize our work and thoughts in the discussion in Chapter 5.

2 Background

2.1 Historical Data

We worked with two historical datasets that were recently digitized for research use by scholars as well as the general public. The datasets are very large and would require years of human effort to exploit it for a goal like intertextual quotes, parallel passages, related articles and topic alignment. This is where computational tools come in handy; with them, one can scan the whole dataset in a matter of minutes and give scholars a list of findings ordered by their relevance.

2.1.1 The Jewish Historical Press Project

The Jewish Historical Press Project website [2] is first and foremost a unique source of information and satabase on the history, life and culture of world Jewry in the modern era. The project brings the digitization revolution to these fields and offers the possibility to perform a full-text search through over 115 publications, in 10 languages from a wide variety of Jewish communities. The Historical Jewish Press website has taken upon itself the task of uploading to the Internet Jewish newspapers published around the world from the eighteenth century onwards. The website hopes to digitize and give access online to the majority of Jewish newspapers and journals published in the far and the near past, including extremely rare newspapers, to which access has been practically impossible.

The project is a joint venture of the National Library of Israel and Tel Aviv University (under the academic leadership of Professor Yaron Tsur, who founded the website in 2004). Not only did it transform the way scholars conduct their research, but also – and just as importantly – it has brought this rich source to the wide public. The project's users today range from independent scholars, to architects, journalists, lawyers, genealogists, high-school students and many more. Due to the fact that all of the material (over 1.5 million pages) is available to the public free of charge, the project is a great example showing the merits of the democratization of knowledge. In the near future, the number of publications and pages on the site is planned to grow dramatically, and will continue into new directions and less represented (as of today) diasporas: South America, Asia, Australia and South Africa.

The ephemeral nature of newspapers and other periodicals, and the fact that they were written, edited and published by the people of the local community and intended to the local readership, make this type of material ever so intriguing, and the new technologies enable us to try and catch the zeitgeist, unmediated.

The newspapers were digitized in a process including scanning of the newspapers, transforming the input images to text using OCR, and segmenting each page to articles. Even though these technologies are mature and used heavily for many applications, they are not foolproof. OCR and segmentation identification usually does not get to 100% accuracy, and is highly affected by the quality of the material.

The Historical Jewish Press website works with some very old newspapers, so they had to overcome certain phenomena that reduce the OCR and segmentation success chances: [1]

> These phenomena include inferior quality of printing (which characterizes early publications), yellowing paper, marks or damage in the original printing, unique fonts, torn pages, scribbled-on pages, and even pages damaged by rodents.

The Technological limitations and the raw materials poor quality lead to word identification problems where a word was not identified or was not identified correctly (characters were replaced or added) or words that were identified from noise in the image but do not really exist. Another problem that occurs is wrong segmentation when a page is not well segmented and the output article contains more than one real article or one real article that is segmented into multiple articles.

We will show that even with the existence of word identification problems and noisy output texts we can get very good article alignment and topic categorization. Regarding the second problem, we rather ignore and regard to an alignment as a correct one if it is aligned to one of the articles contained in the output article.

2.1.2 Tibetan-Buddhist Canon

The Tibetan language belongs to the Tibeto-Burman branch of the Sino-Tibetan family. It is monosyllabic (morphemes are single syllables) for which computerized language tools are largely lacking.

The texts were inputted manually by the Asian Classics Input Project [1]. The texts are transliterated to roman letters using Wylie transliteration scheme [30].[2] You can see the transliteration scheme in Appendix A.

---

[1] From the JPress website: `http://web.nli.org.il/sites/JPress/English/about/Pages/about_technology.aspx`.

[2] ACIP inputted the texts using a slightly modified form of Wylie, we have done some adjustments, replacing the characters in questions to comply with the Wylie transliteration schemes.

The Tibetan language has a plethora of (usually) monosyllabic grammatical particles, which are often omitted. Occasionally, the same syllable can be written using one of several orthographic variations, for example, sogs and stsogs. In the case of verbs, the syllable has various inflectional forms that are often homophones, a fact that can result in variants in the reading due to scribal errors or lack of standardization. For these reasons we need a system that can find approximate string matching and not only exact string matching.

The Tibetan-Buddhist canon consists of two parts:

The *Kangyur* has around 108 volumes containing what is believed by tradition to be the Word of the Buddha, which means that they are anonymous, no author named. If the scholars can locate the shared texts, this would facilitate determining the history of their composition: the relative chronology of the texts (who copied from whom and when), in which intellectual milieu they were composed, which in turn will facilitate the study of history of ideas of Buddhism. These texts were mostly translated directly from the Sanskrit original (with some from other languages and others indirectly via Chinese).

The *Tengyur* has about 210 volumes consisting of canonical commentaries, treatises, and various kinds of manuals that were likewise mostly translated from Sanskrit (with some from other languages and a few originally written in Tibetan). Although here normally the author is named, very often the author cites other texts or "borrows" from them, and finding these shared passages would likewise help the study of their composition in a similar manner, and help see what texts were available to the authors, and so on.

2.2 Related Work

2.2.1 Topic Detection and Tracking (TDT)

The article alignment task that we presented is very close to the research field of Topic Detection and Tracking. The TDT research program started in 1997 and gathered a research community in order to find core technologies for organizing streams of news articles by topic [4].

The TDT program defined a *topic* as "a set of news stories that are strongly related by some seminal real world event", a *story* as "a topically cohesive segment of news that includes two or more declarative independent clauses about a single event.", and an *event* as "something that happens at a specific time and location" [19].

The TDT community selected five standardized tasks that were essential for the success of a practical real time TDT system:

1. Story Segmentation Task – segmenting a stream of text into individual stories by detecting topical changes.

2. Topic Tracking Task – given input stories the system needs to "track" these stories by alerting when a new story of the same topic has been encountered.

3. Topic Detection Task – cluster news stories by topic.

4. First Story Detection Task – decide if a story in a stream of news stories is the first one on a new topic.

5. Link Detection Task – decide if two given news stories have the same topic

It can be easily seen that the tasks share some similarities and have similar challenges. For instance, a perfect solution to the Link Detection problem would solve the other tasks.

The tasks had a cooperative evaluation program with yearly competitions that were sponsored by DARPA.

A difference between the aligning articles in historical newspapers and the TDT tasks that we look for stories in a retrospective way on a static database (Retrospective Event Detection (RED)) where the TDT wish "to identify new events as they occur, based on an on-line stream of stories." (New Event Detection (NED)). A Retrospective can use for the topic detection the hierarchical agglomerative clustering (HAC) which clusters the corpus in a bottom up by connecting the most similar clusters (every story starts as a cluster). But this clustering method is quadratic in the size of the corpus and is not feasible for an online event detection that need to use a single pass (incremental) clustering which is similar to the HAC method but it builds the clusters by the

order of the temporal ordering of the stories. For example a common way attending the detection problem was by representing stories with a set of features and assigning every new story to the cluster of the most similar story from the past (one nearest neighbor). Yang and Pierce [32] show that the retrospective event detection gets much better results than the online results.

Allan and Harding[5, 6] used a story representation of tf-idf weights of the 1000 most weighted terms in the document, computed cosine similarity of the story to every previous story and assign the story to the nearest neighbor cluster if the similarity was above a threshold otherwise it creates a new cluster containing the story. In this paper we use that simple but powerful method and it indeed got very good results.

Other document representation and similarity measures were developed and tested; especially worth mentioning [31, 27, 28] that used language models for the clusters in order to get probability that a new story is related to a cluster and used it as a similarity measure.

Kumaran and Allan[16] had a supervised system for the problem of First Story Detection. The system used a Support Vector Machine (SVM) classifier and the features were cosine similarity scores of three vector comparisons: named entities, topic terms and full text of the document. The addition of the named entities vectors helps finding articles that do not only have the same general topic but also share one or few named entities like organizations, locations, persons and time which increase the confidence of the match significantly.

C.C. Chen and Y.T. Chen [11] suggested adding an aging theory to model the life cycle of related sequential events. They added to regular single pass clustering an energy component that predicts the state in the life cycle of the story and removes clusters that their energy goes under a threshold. The energy of the cluster increases when new stories join the cluster but then it also has a time decay which means that a long time after no new stories come, the energy of this cluster will fade and it will be removed from the cluster candidates. They reported that when the name entity extraction did not work well on the output of automatic speech recognition (ASR) it degraded the accuracy of the system significantly. For similar noisy text reasons we could not use automatic named entity extractor.

New Event Detection has recently regained its popularity with the emerge of social media. The high volumes of data do not make the traditional way of comparing the similarity of a new document to all previous documents or clusters so Petrović and Osborne [21] suggested using locality-sensitive hashing (LSH), an approximate nearest neighbor algorithm, to get a constant number of comparisons instead of a linear quantity, while still achieving comparable results.

Petrović and Osborne [22] used lexical paraphrases from three sources of paraphrases including WordNet and to expand the document vector and overcome lexical variation in documents with different terms that discuss the same topic.

Moran and McCreadie [18] showed how to use word embedding for expanding tweets with semantically related phrases to improve the FSD accuracy. They considered word vectors with cosine similarity higher than a threshold as a lexical paraphrase. This method does not depend on

complex linguistic resource like WordNet and still got better results. This resource independence allows to run it on low resources languages. In this paper we used similar approach to augment the document bag of words.

Avraham and Goldberg [7] reported when using word embedding in hebrew, which is a morphological rich language and very inflectional, they discovered that the hebrew word vectors bear a mix of semantic and morphological properties and that many word vectors similarity are of other inflections of the target word base form and rather real semantic similarities. In this paper we leveraged the morphological properties of the word embedding to expand the terms with a terms of the same word base.

2.2.2 Noisy Document Topic Categorization

Agarwal et al. [3] experimented with texts from different resource types in order to understand how different level of noise in the text would hurt the text classification success. For that purpose they developed a spelling error simulator and created synthetic texts with few levels of noise. They reported that in up to 40% word error rate, the accuracy of the text classification was not affected significantly. Vinciarelly reported similar results in text clustering and categorization (classification) tasks [12, 29]. We were encouraged from these results to work on topic detection in existence of OCR noisy text.

2.2.3 Historical Text Reuse

Prasad and Rao [23] looked for citations in Sanskrit texts. They compared two methods: Exact string matching and approximate string matching. In the exact string matching, they reduced the number of sentence comparisons by sorting all the sentences in the source corpus and then running search again the ordered sentence list. For the approximate string matching they used a variant of the Smith-Waterman algorithm. This algorithm is used bioinformatics for local alignment of biological sequences. This Smith-Waterman is using dynamic programming to find the subsequences with the highest similarity score. Prasad and Rao use a similarity cutoff (a threshold) to decide if the two sentences are related. They reported that the results of the approximate matching are superior and returned many citations that were not recognized by the exact matches, but at a much higher computational cost. In this paper, we use the Smith-Waterman algorithm for a postprocessing step to find optimal boundaries of the matches attained in the previous step and use its score to rank the matches. We also showed running this algorithm in a word level and using word importance measures for the gap, mismatch and match scoring scheme (instead of using fixed weights) can improve the results.

Horton and Olsen [20] implemented similar passages and near quotations identification in a document against a preprocessed corpus. They extracted all the shingles (trigrams of words) from every document and indexed the shingles as the keys while all the locations of a shingle in all the documents stored as value. Later given a query document it retrieves from the index all the shingle

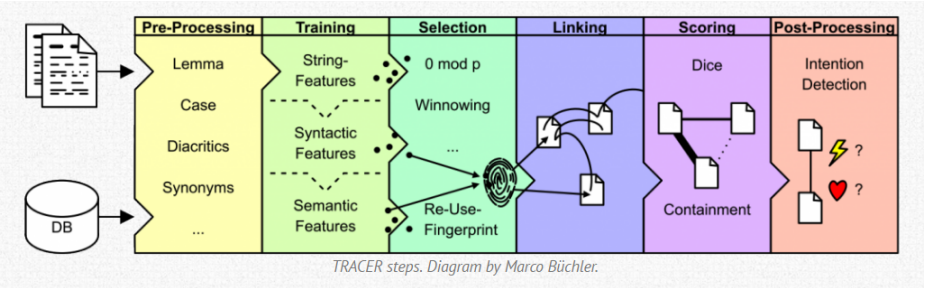


Figure 2.1: Architecture of the TRACER algorithm

matches and connects the matches if they fall in a gap window predefined. Kolak and Schilit [15] had a similar solution but only for exact quotations by grouping together only overlapping shingles.

Büchler et al. [9, 10] developed a general text re-use called TRACER with 7-level architecture . Each step is configurable and can be optimized to specific text re-use task and corpora. The steps are preprocessing, featuring, selection, scoring and post-processing as illustrated in Figure 2.1. This approach is called Feature-Based linking where only text reuse units with shared features are compared and it has squared complexity in the number of extracted features as appose to Brute-Force linking which compares all text re-use unit couples and has squared complexity in the number of text re-use units. First step is segmentation, the process of dividing the corpus into text re-use units. The segments can be overlapping text fergments or disjoint segments like documents, pages, paragraphs, lines or sentences. The preprocessing step can include cleaning of the text, case transformation, and to create equivalence classes by using stemming, lemmatization, synonyms and finding close words with low edit distance. The featuring step decides how the text re-use unit is represented: word types, character n-grams or word shingling are possible features that could be compared. The selection step removes features that are less important. A simple and common selection is min and max pruning (removing the extreme high and low frequency features). The linking step is usually the computationally expensive part where features of the different text re-use units are matched, usually by hashing. The scoring step is for evaluation and scoring of the overlap in the linked re-use units, and post-processing could be used to remove noisy links.

Shmidman and Koppel [25] proposed a very efficient method for locating parallel passages in large Hebrew and Aramaic corpus. They represent every word with the two least frequent letters and for every document keep all four skip-grams. Because of the lean representation (8 character string with alphabet of 22 letters leads to 22^8), the skip-gram can be uniquely represented with a single 64-bit integer, or even with 32-bit integer when keeping a separate index for every first word in the skip-gram and lest skip gram is 6 characters long. After indexing the corpus in a single pass, they iterate over the indexes to merge the skip-gram matches into longer parallel paragraph.

3 Finding Related Articles Despite Noisy OCR

The main problem we try to solve in this chapter is the following: given an article in a (historical) newspaper, find more articles relating to the same event.

The way we do this is by representing the news articles as vectors and finding articles that have the closest vector to the vector of the query article. Different representations assist in different tasks. In this work, we tried to find the representation that is optimal for this task.

3.1 Algorithm Outline

1. Text preprocessing and cleaning.
2. Representing articles as article vectors.
3. Finding nearest neighbors for the query articles vectors.

3.1.1 Text preprocessing and cleaning

The OCR outputs many non-alphanumeric characters because of the noise in the image that it sometimes recognizes as characters. The noisy characters when appearing in words have only a small probability to appear again in the same word exactly in the same location and it causes the article vectors to become much more sparse and harder to compare. We remove all non-alphanumeric characters. The removal of punctuation does not affect the semantics of the article. In Hebrew, the only words affected by the removal are acronyms that have *gershaim* (double quotation mark used to indicate the abbreviation), but it will map the same acronyms to the same words all over the corpus. Another problem that is common to OCR systems is poor word segmentation. Some words are concatenated, while sometimes the OCR outputs single letters as words (could also occur because of noise in the image). We remove all single-letter words that anyway do not have a meaning in Hebrew.

3.1.2 Representing articles as article vectors

We need a way to represent documents as vectors so we can calculate similarity measures between the documents. We used an algebraic model, in which every document is represented by a term vector. (This model is also called the "Vector Space Model" or VSM.) Terms can be single words, keywords or phrases, and the vector could also have as attributes word order and part-of-speech tags. Terms may be weighted according to their importance in the document.

3.1.2.1 Set of Words

This is the simplest baseline. Every document is represented by an unordered collection of the distinct words in it, and so the dimension of each vector is the same as the size of the vocabulary (number of distinct words occurring in the corpus). In this weighting scheme, all the terms that appear in a document get the same weight disregarding how many times they appeared in the document and ignoring their frequency in the corpus.

3.1.2.2 tf-idf

[24] This well-known representation is a standard in information retrieval, and while it is simple, it gives good and competitive results. It reflects the word importance in the document that is in a collection of documents (corpus). The weight $w_{i,j}$ of the term t_i in document d_j is proportional to number of occurrences of the term ti in document d_j , but is inversely proportional to the number of documents the term appeared in the entire corpus.

$$tf(t_i, d_j) = \sqrt{freq(t_i, d_j)}$$

$$idf(t_i) = 1 + \log\left(\frac{numDocuments}{docFreq(t_i) + 1}\right)$$

$$\textit{tf-idf}(t_i, d_j) = tf(t_i, d_j) \times idf(t_i)$$

This lets us know which words are more general because they are common in the corpus, and which words and more unique and maybe relate more to specific documents. This approach can help us to find sets of words that are discriminative for documents in the collection.

3.1.2.3 tf-idf augmented with Hebrew stems

Hebrew is a morphological language, and has few prefixes and suffixes attached to the content word without changing much the semantic of the word. The affixes lead to sparser term vectors and lower the chances of documents with similar topic to intersect on these terms. That is why

it is common to remove the affixes and add the morphological stems. The original word is kept. Adding the word without the prefixes and if possible its lemma can increase its generalization and the hit ratio when comparing with similar documents. We used an Hebrew analyzer that expands each term to all recognized forms found in the hspell dictionary [13]. In case a word wasn't recognized, it will try to tolerate spelling errors or omitted *matres lectionis* (*yod/vav*).

3.1.2.4 tf-idf augmented with word2vec Nearest Neighbors

"You shall know a word by the company it keeps" (J. R. Firth, 1957). Based on this maxim, many distributional word representations algorithms have been developed in the past years. These algorithms utilize the contexts of words and their co-occurrence distribution to represent words. Their output is a mapping from words and phrases to real numbers vectors. The method are also called word embedding algorithms.

In 2013, Tomas Mikolov et al. presented a word embedding tool that uses shallow neural networks called word2vec [17]. word2vec has gained popularity due to its efficiency and its success in representing words as vectors, twhich has been used in many NLP tasks to obtain state-of-the-art results.

As in Mikolov's paper, one is using the fact that words with similar semantics but with different syntactic roles have similar vectors. (In their paper they are looking to find related words by type automatically.)

In our case, we leverage two attributes of word2vec:

1. Words with different affixes and shared lemma will have similar vectors.

2. Words with OCR recognition errors will have similar vectors to the original words (as actually written in the newspaper).

The first attribute help us to achieve generalization similar to that of the morphological analyzer but without any specific language logic needed. The second attribute can help reduce the noise in the OCR output, but – unlike OCR post-processing for correcting OCR errors – here we do not need to replace the error with the exact word that appeared in text but rather with a good representative that appears in other documents.

Our method of word2vec neighbors term expansion:

1. Run the word2vec on the OCR'ed newspaper corpus (in a single language).[1]

2. For every word in the article (after cleaning):

[1] `https://code.google.com/archive/p/word2vec/`

(a) Find the word2vec K nearest neighbors of the word (we used $K = 20$)

(b) Filter the neighbors with edit distance $< d$ (we used $d = 3$).

(c) Add to the article's bag of words the most frequent neighbor if its frequency is bigger than the original word's frequency.

We trained the word2vec algorithm over the all of the articles of the OCR'ed *Davar* newspaper during 1981, which contains over 14 million words. We used word2vec's defualt configuration: continuous bag of word (CBOW) architecture with dimensionality of the word vectors set to 200, context window size set to 5 and hierarchical softmax training algorithm.

We wanted to augment the documents' vectors only with syntactic similar words and not with semantic lexical paraphrases so we augment with words that are close to what appeared in the actual text (this is usually better for Topic Detection). In order to eliminate words we believe do not have the same word base from the word's word2vec neighbors candidates, we removed neighbors that have an big edit distance. We used Levenshtein distance which is defined by the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. Our tests showed that 3 is a good threshold even though there still are a few cases where semantic neighbors (that are not from the same word root) pass the filter.

In Figure 3.1, we show few examples of comparison between the input words of the two methods (word2vec nearest neighbor and Hebrew analyzer) that were augmented to the output words (in the left column). It is easy to see that the word2vec is augmenting many words with ocr noise, and also words that have the same word base where the analyzer is only good for the syntactic augmentation. Another advantage that the word2vec has over the Hebrew analyzer is in augmenting named entities that are frequent in the corpus which the analyzer does not recognize.

3.1.2.5 Word Vector Averaging

This method is a common baseline in sentence and document vector representation. It gives the centroid of the document's distributional word vectors, and so it is expected to represent the topic of the document. In this task, we find that using tf-idf weights for the weighted arithmetic mean of the word vector performs better than uniform weighting, and therefore we only report the results of methods with tf-idf weighting.

3.1.3 Finding Nearest Neighbors for query articles

There are several similarity measures or distance measures possible to find the nearest neighbors among them are cosine similarity, Jaccard similarity, Hellinger similarity, KL divergence, euclidean distance and Manhattan distance. We chose to work with cosine similarity as it was reported as the most successful in many previous topic detection task experiments and also it is very simple and fast to calculate using the dot product of the normalized vectors. It is also easy to

Table 3.1: Method performance in k-best "same event" related articles output

Method	precision@1	precision@3	precision@5	precision@8	recall@8
Set of Words	0.77	0.66	0.58	0.48	0.54
tf-idf	0.87	0.77	0.71	0.62	0.70
Hebrew stem	0.89	0.78	0.71	**0.65**	**0.73**
word2vec nearest neighbor	**0.90**	**0.80**	**0.72**	0.64	0.72
word2vec words vectors centroid	0.73	0.62	0.52	0.43	0.48
Gold Standard	0.96	0.89	0.82	0.71	0.79

prove that when the vectors are normalized, the problems of finding the most similar vector with cosine similarity or euclidean distance are equivalent.[2]

$$\text{cosSimilarity}(V_x, V_y) = \frac{V_x \cdot V_y}{\|V_x\|\|V_y\|}$$

3.2 Evaluation

We evaluate our results by finding articles of the same continuous event from different editions of the same publication.

For this task, we used articles from one month of the *Davar* newspaper. By finding articles that cover the same event from different dates, we can get a quick understanding of an ongoing story.

We chose 111 query articles, some reporting on major continuous events while some reporting on anecdotal events that had only a few or no related articles. For each query article, we searched for the top 8 nearest neighbors using each representation and labeled the returning results as one of: same event match, same genre, or not related at all.

As you can see in Table 3.1, the basic baseline of tf-idf vectors got about 90% of the possible related articles in the gold standard. The vectors augmented with hebrew stems improved the results by 2%, while the word2vec augmentation added 3% to the tf-idf baseline, beating the Hebrew analyzer. The word2vec vectors centroid did not get as good results but got more articles that are from the same genre, this is probably because for topic detection it is important to have term matches and when using more general word classes (and in this case words with similar vectors) we get more general topic vectors and less event specific vectors.

[2] `http://stats.stackexchange.com/questions/146221/is-cosine-similarity-identical-to-l2-normalized-euclidean-distance`

Output Word	Hebrew analyzer expansions	Word2Vec neighbors expansions
פורסם	[שפורסם, ופורסמה, ופורסמו, המפורסם, יפורסמו, פורסמו, פורסמה, שפורסמו, ופירסם, פירסם, שפירסמה, שפירסמו, פירסמו, פירסמה, שיפורסם, המפורסמת, שפירסם, יפורסם, ומפורסמים, מפורסמים, שיפורסמו]	שפורסם, פורסם, פורטם, פורפם, פרסם, פורסמו, פורסמה, פירפם, פירסם, פירסם, יפורסם
עיראק	העיראקי, לעיראקים, מעיראק, עיראקיים, ועיראק, שעיראק, בעיראק, עיראקית, העיראקים, העיראקית, העיראקיות, העיראקיים, לעיראק, והעיראקים, עראק, עיראקי	העיראקי, עידאק, עייאק, עיואק, ועיראק, שעיראק, עיראקית, לעיראק, ראק, עראק, עיראקי, חעיראקי
פיליוני	פילון, פלוני	מיליוני
רבין	ורבין, שרבין, לרבין	ורבין, ורביו, זורבין, רביו, רביז, רגין, רכיו, רכיז, רכין, רנין, רבמ, רבץ, לרבין
חיפה	ובחיפה, שיחפה, לחפות, מחיפה, בחיפה, סחפו, וחיפו, וחיפה, החיפאית, החיפאים, החיפאי, בחפות, כחיפה, חיפת, חיפי, חיפו, חפה, חפי, חפו, תחפו, חפינה, יחפה, חפותה, מלחפות, לחיפה	וויפה, וזיפה, היפח, היפה, היפר, היפת, היסה, חיסר, חיפת, חיפר, חיפח, חיפו, חיפד, חיסח, חיסה, חיטה, חפה, חיפוז, חיפזז, זויפה, תיפה, תיפח
רייגן		דייג, מיגן, רייגל, רייגו, רייגז, רייגי, ברייגן, ריגן, רייג, ריען, רימן, וייגן, ויינן, שרייגן, דייגן, דייגו, דיינן, ריינן, ריינז, ייגן, תיגן, רגן, רי, לרייגן, ייגן
שמידט		שפידס, שפידפ, שפידט, שפיוט, שפירט, שמידט, שמירט, שמידם, שמידמ, שמידס, שמידפ, שמדט, שטידט, לשמידט

Figure 3.1: Example of words that were augmented using Hebrew analyzer and word2vec NN methods

4 Intertextuality in Tibetan Texts

In this chapter we present the algorithm steps for finding parallel passages in the Tibetan corpus.

4.1 All Paths Below Threshold

The *All Paths Below Threshold (APBT)* algorithm was developed by Barsky et al. [8] for finding similar biological subsequences. This algorithm looks for an "all against all approximate substring matches" at the character level. Klein et al.[14] adapted this algorithm for the Tibetan corpus. They added preprocessing step for cleaning the text from non-letters charters and a postprocessing step to merge the matches and remove non-aligned text by using local alignment. They added support for parallelism to boost the long calculation on alignment of long sequences.

The definition of the problem that APBT solves is:

ALL ERROR-BOUNDED APPROXIMATE MATCHES

Input: Strings s and t over alphabet Σ with corresponding lengths M and N, and positive numbers K and S.

Output: All error-bounded approximate maximal matches $(s[i,j],t[k,l])$ such that

1. $EditDistance(s[i,j],t[k,l]) \leqslant K$
2. The lengths of the matched substrings $s[i,j]$ and $t[k,l]$ must be at least S

APBT solves the problem by recasting it to the problem of finding the cheapest (maximal) path in a matching graph.

APBT represents the problem using a matching matrix $M_{s,t}$ of s and t. The matrix has all the matches of characters in s and characters in t such that $M_{s,t}[i,j] = 1$ if $s[i] = t[j]$ otherwise it is zero.

Based on the matching matrix $M_{s,t}$ they define an induced weighted directed graph $G_{M_{s,t}}$ such that every cell with the value 1 is a vertex and there is an edge from $v_{i,j}$ to $v_{k,l}$ iff $i < k$ and $j < l$.

	t	a	a	c	b	a	c	a	c	a	b
s		0	1	2	3	4	5	6	7	8	9
a	0	1	1	0	0	1	0	1	0	1	0
b	1	0	0	0	1	0	0	0	0	0	1
a	2	1	1	0	0	1	0	1	0	1	0
b	3	0	0	0	1	0	0	0	0	0	1
a	4	1	1	0	0	1	0	1	0	1	0
c	5	0	0	1	0	0	1	0	1	0	0
b	6	0	0	0	1	0	0	0	0	0	1
b	7	0	0	0	1	0	0	0	0	0	1
b	8	0	0	0	1	0	0	0	0	0	1
c	9	0	0	1	0	0	1	0	1	0	0

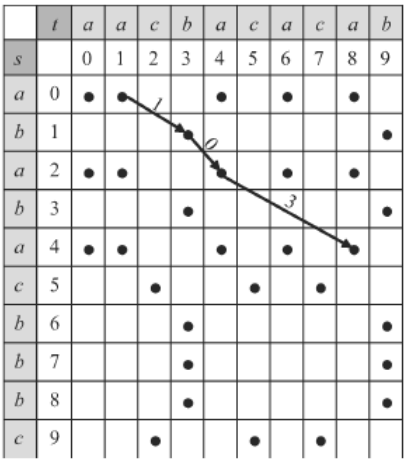


Figure 4.1: Example of matching matrix and its induced graph

The weight of the edge (or cost of the edge) is the number of characters being skipped (and therefore increasing the edit distance) and equals to $max(k-i, l-j)-1$.

A path π starting at $v_{i,j}$ and ending at $v_{k,l}$ will represent the substrings $s[i,k]$ and $t[j,l]$. The match length of π equal to $min(k-i+1, l-j+1)$ and the error number of π (the edit distance) is the sum of all costs of all edges in π. You can see a simple example of this problem representation in Figure 4.1

APBT has time complexity $O(MNK^3)$ and memory bounded by $O(N \cdot (2K+1) \cdot S_{max})$. The algorithm avoids explicitly building the full matching matrix (with quadratic time and space) and instead generates lean representation in linear time and space of $O(N \cdot |\Sigma| + M)$. Regarding the graph $G_{M_{s,t}}$ the algorithm never explicitly constructs it but rather traverses it by constructing only the needed paths. The algorithm and has few optimizations that discard many others paths that are proven to be inferior or not valid (violating the thresholds). The optimizations reduce the search-space and bound the runtime.

We modified the method to work with syllables as the "alphabet" of the string, rather than the individual character. We did it by giving every syllable in the vocabulary of the corpora a unique numerical id, and represented the texts as sequences of these ids.

The quality of the results was better because, with character-wise alignment, syllables can share many letters but have no semantic similarity. In contrast to bioinformatics and the DNA sequences the algorithm was applied on, natural language string is split to meaningful words and in Tibetan also to syllables. When running the APBT code directly on the character level strings there are character alignments that we would like to avoid: when aligning characters from one word to characters in more than one word in the other text. Performing the APBT algorithm on word level instead of character level removes this kind of false positives.

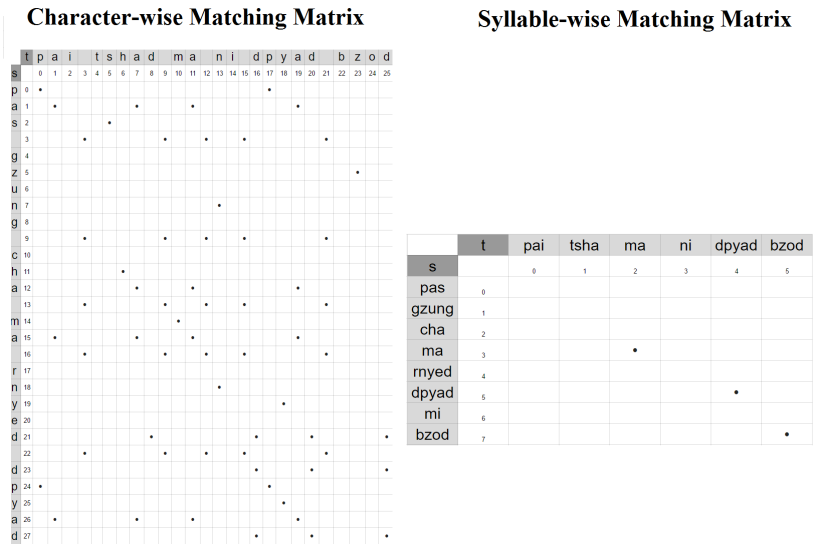


Figure 4.2: Example of character-wise and syllable-wise matching matrices

Another positive side affect is that it reduces the run-time by 99 percents. The APBT algorithm is faster when the alphabet Σ is larger and sparser so it reduces the search space, as illustrated in Figure 4.2. Also, since on the average a syllable has 4 or 5 characters (and the run-time is squared in the string input size) the speedup was in magnitudes.

The major downside of this approach is that is assumes that the matches have many shared syllables. On the last step we have local alignment and check whether the replaced syllables (mismatches) have small edit distance and high similarity and make this into account for setting the boundaries and the ranking of the matches.

4.2 Merge

The APBT algorithm can output overlapping or adjacent passages, which can be merged into a longer passages. The APBT has thresholds that allow the algorithm to reduce the search space to bounds the performance of the run-time and memory, like maximum match size S_{max} and maximum edit distance K. These threshold lead to many overlapping or close spans of matches. We are interested in finding parallel passages rather than many small fragments. We also would have prefer to have a proportional error threshold that allows longer matches with more errors rather than fixed. (The thresholds are fixed and can not be changed during the run of the algorithm because they are an inherent part of the optimizations that bound the runtime.)

One way to get by this barrier is by outputting the segments in the first step and recursively merge overlapping matches and segments that are within a distance which is proportional to the matches lengths together to longer matches.

4.3 Local Alignment and Ranking

A followup task is to rank the parallel passages. There are two main parameters that are relevant: the length of the shared text and an estimate of significance. This step is crucial since the APBT algorithm outputs thousands of matches and we aim to allow the scholars to work only on the most interesting and nontrivial matches, moving the more noisy results to the bottom of the list. In our current work, we are incorporating a new ranking mechanism that counts not only match length but also the importance of the content of the match through a smarter local alignment that scores matches and mismatches of different syllables with using significance differentially. We run this algorithm with syllable as a character in the alphabet (sequence of syllables).

We locally align sequences using the dynamic programming (Smith-Waterman) algorithm [26], which scores each match, gap and replacement. Consider two sequences $A = a_1a_2a_3\ldots a_m$ and $B = b_1b_2b_3\ldots b_n$

$$
\begin{aligned}
&H(i,0) = H(0,j) = 0, 0 \leq i \leq m, 0 \leq j \leq n \\
&H(i,j) = max \begin{cases} 0 & \\ H(i-1,j-1) + sim(a_i,b_j) & \text{Match or Mismatch} \\ H(i-1,j) + gap(a_i) & \text{Deletion} \\ H(i,j-1) + gap(b_j) & \text{Insertion} \end{cases} \\
&where 1 \leq i \leq m, 1 \leq j \leq n
\end{aligned} \tag{4.1}
$$

After filling the matrix, we locate the maximum score in the similarity matrix, traverse backward in the matrix (choosing the highest neighbors) until reaching a cell with zero. The path from the highest score cell to the zeroed cell gives us the maximum substring match.

In the default setting, the match, gap and mismatch have fixed scores (configurable). For example:

Scoring Scheme	
Match Award	+1
Mismatch Penalty	-1
Gap Penalty	-1

In our "smart" weighting, we extract the scores for each of the three options using idf counts from the entire Tibetan canon. These scores enable us to score sequences not only based on their length but also by the importance of their matched terms, which affects both the ranking and also the boundaries of the matched passages. To handle the orthographical differences between sources, we consider as a match also two syllables with a small edit distance (using edit distance). The scoring scheme can be seen in Equation 4.2.

Table 4.1: Test Set Details

Text	number of chars	number of syllables	number of unique syllables
Phywa	38191	8797	414
Khlonh	44319	10310	496

	Average match size	Minimum match size	Maximum Match size	SD of match size
Labeled Matches	84.6	4	311	90

Table 4.2: Method performance in k-best matched passages

Method	p@5	p@10	p@15	p@20	p@25
Character based	1.0	0.7	0.466	0.35	0.28
Syllable based	1.0	1.0	0.933	0.8	0.76
Syllable based + idf based scoring	1.0	1.0	**1.0**	**0.9**	**0.8**

$$sim(a_i, b_j) = \begin{cases} +idf(a_i) & a_i = b_j \\ +\max(idf(a_i), idf(b_j)) \left(\frac{LCS(a_i,b_j)}{max(\|a_i\|,\|b_j\|)}\right) & a_i \neq b_j \wedge editDist(a_i, b_j) \leq \min(\|a_i\|, \|b_j\|) \times 0.5 \\ -\max(idf(a_i), idf(b_j)) & a_i \neq b_j \wedge editDist(a_i, b_j) > \min(\|a_i\|, \|b_j\|) \times 0.5 \end{cases} \quad (4.2)$$

$$gap(x) = -idf(x)$$

4.4 Method Comparison

The results are reported on a test set that was manually annotated by Tibetan scholars. This data set was extracted from two sources, one is known to "borrow" from the other one. Since the exact boundaries of the matched passages are less important to us, we consider each pair that overlaps a labeled match as a correct match.

There were a total of 24 labeled matches. There is a big variation in the labeled matches lengths, As you can see in Table 4.1.

As seen in Table 4.2, the syllable-wise APBT was superior to the character-wise APBT. We expected all the algorithms to return first the long matches and they all did but after the top 5 matches the matches are shorter and have closer lengths so the character-wise that have more noise has less precision and we can see that the syllable-wise APBT that ranked the results with the idf based scoring improved the syllable-wise baseline.

5 Conclusions

In this work, we have explored different ways of enriching historical datasets and have developed utilities to enable scholars to mine the huge mass of historical information that is now available in digital form.

First, we showed how historical newspaper articles can be connected together into a continuous story with high precision. Second, we presented a new way to identify parallel passages in ancient (Tibetan) texts.

In order to find related articles in OCR'ed Hebrew newspapers, we used a traditional and simple method that represents articles as tf-idf vectors and finds the related articles by using nearest neighbors search with cosine similarity measure. It proved to work well even though the input text was very noisy and the language is highly inflectional and morphological rich. We demonstrated that using prefix removal and augmenting the vector with lemma forms of the tokens improves the precision. More important, we showed how to achieve even better results without any language specific resources using word embedding similarities.

We suggest for future work to filter some of the neighboring candidates by first running a topic classifier and only comparing them to neighbors within the same broader topic. We plan to try our algorithm on other newspapers in more languages.

We think that the word-embedding vector similarity could be utilized also for query expansion in information retrieval and could be used for term standardization instead of the regular lemmatization process. In noisy OCR texts this method can be used for post processing error correction, and also to help in finding named entities that were not identified correctly by the OCR.

In the work on finding approximate quotations we borrowed an algorithm from bioinformatics that finds all approximate string matching of two strings and adapted it to natural language by running it on a string that its basic alphabet is words instead of letters. This adaptation improved both precision of sensible matches and reduced the runtime in magnitudes of order so now it is feasible to run it over a collection of documents. We also introduced a post-processing step that runs specialized local alignment in order to set best boundaries for the matched spans and output scores influenced by quantity and importance of the matched, replaced and misses words in order to rank the match. This step improved the results and we plan to keep investigating this direction to achieve a general approximate string matches ranker.

We will continue working on a generic method for finding parallel passages (of whatever length) in different texts and corpora. Approximate text matching can find all similar passages of one work in another or within a given large corpus. The results can also be used to uncover orthographic peculiarities and lexical variations. Incorporating morphological and lexical awareness (for specific languages of interest) to the approximate-search algorithm would also be invaluable. Another natural extension would be to use machine-translation tools to find parallels between texts written in different languages. For example, Buddhist scriptures are constructed largely of pericopes; thus, there is a high level of intertextuality between sources in a variety of languages (Sanskrit, Pali, Tibetan, Chinese, Khotanese).

We believe that a way to overcome the current severe limitations of handwriting character recognition is by matching partial and noisy recognition results to available texts, as suggested in [33]. Our approximate string-matching algorithms can be used to locate other occurrences of tentative and fragmentary readings within existing corpora. These need to be adapted to take the likelihood of alternate readings into account, as well as expected letter widths and spacing considerations for defective manuscripts. Such flexible algorithms will find multiple uses in manuscript studies.

References

[1] The Asian Classics Input Project website. URL http://www.asianclassics.org/.

[2] The Jewish Historical Press Project website. URL http://jpress.nli.org.il.

[3] S. Agarwal, S. Godbole, D. Punjani, and S. Roy. How much noise is too much: A study in automatic text classification. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 3–12, Oct 2007. doi: 10.1109/ICDM.2007.21.

[4] James Allan. *Introduction to Topic Detection and Tracking*, pages 1–16. Springer US, Boston, MA, 2002. ISBN 978-1-4615-0933-2. doi: 10.1007/978-1-4615-0933-2_1. URL http://dx.doi.org/10.1007/978-1-4615-0933-2_1.

[5] James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study – Final report. Technical report, 1998.

[6] James Allan, Stephen Harding, David Fisher, Alvaro Bolivar, Sergio Guzman-Lara, and Peter Amstutz. Taking topic detection from evaluation to practice. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 4 - Volume 04*, HICSS '05, pages 101.1–, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2268-8-4. doi: 10.1109/HICSS.2005.576. URL http://dx.doi.org/10.1109/HICSS.2005.576.

[7] Oded Avraham and Yoav Goldberg. Word embeddings in Hebrew: Initial results. 2015.

[8] Marina Barsky, Ulrike Stege, Alex Thomo, and Chris Upton. A graph approach to the threshold all-against-all substring matching problem. *J. Exp. Algorithmics*, 12:1.10:1–1.10:26, June 2008. ISSN 1084-6654. doi: 10.1145/1227161.1370601. URL http://doi.acm.org/10.1145/1227161.1370601.

[9] Marco Büchler. *Informationstechnische Aspekte des Historical Text Re-use*. PhD thesis, 2013.

[10] Marco Büchler, Philip R Burns, Martin Müller, Emily Franzini, and Greta Franzini. Towards a historical text re-use detection. In *Text Mining*, pages 221–238. Springer, 2014.

[11] Chien Chin Chen, Yao-Tsung Chen, and Meng Chang Chen. An aging theory for event life-cycle modeling. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 37(2):237–248, 2007.

[12] David Grangier and Alessandro Vinciarelli. Noisy text clustering. Idiap-RR Idiap-RR-31-2004, IDIAP, 2004.

[13] Nadav Har'El and Dan Kenigsberg. Hspell – the free Hebrew spell checker and morphological analyzer. In *Israeli Seminar on Computational Linguistics*, December 2004.

[14] B. Klein, N. Dershowitz, L. Wolf, O. Almogi, and D. Wangchuk. Finding inexact quotations within a Tibetan Buddhist corpus. In *Digital Humanities (DH)*, 2014.

[15] Okan Kolak and Bill N. Schilit. Generating links by mining quotations. In *Proceedings of the Ninteenth ACM conference on Hypertext and hypermedia*, pages 117–126. ACM, 2008.

[16] Giridhar Kumaran and James Allan. Using names and topics for new event detection. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 121–128, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220591. URL `http://dx.doi.org/10.3115/1220575.1220591`.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[18] Sean Moran, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Enhancing first story detection using word embeddings. 2016.

[19] Ramesh Nallapati, Ao Feng, Fuchun Peng, and James Allan. Event threading within news topics. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 446–453, New York, NY, USA, 2004. ACM. ISBN 1-58113-874-1. doi: 10.1145/1031171.1031258. URL `http://doi.acm.org/10.1145/1031171.1031258`.

[20] Mark Olsen, Russell Horton, and Glenn Roe. Something borrowed: sequence alignment and the identification of similar passages in large text collections. *Digital Studies/Le champ numérique*, 2(1), 2011.

[21] Saša Petrović, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to Twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 181–189. Association for Computational Linguistics, 2010.

[22] Saša Petrović, Miles Osborne, and Victor Lavrenko. Using paraphrases for improving first story detection in news and twitter. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 338–346, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-20-6. URL `http://dl.acm.org/citation.cfm?id=2382029.2382072`.

[23] Abhinandan S Prasad and Shrisha Rao. Citation matching in Sanskrit corpora using local alignment. In *Sanskrit Computational Linguistics*, pages 124–136. Springer, 2010.

[24] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, August 1988. ISSN 0306-4573. doi: 10.1016/0306-4573(88)90021-0. URL `http://dx.doi.org/10.1016/0306-4573(88)90021-0`.

[25] Avi Shmidman, Moshe Koppel, and Ely Porat. Identification of parallel passages across a large Hebrew/Aramaic corpus. *CoRR*, abs/1602.08715, 2016. URL `http://arxiv.org/abs/1602.08715`.

[26] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981. ISSN 0022-2836. doi: 10.1016/0022-2836(81)90087-5.

[27] Martijn Spitters and Wessel Kraaij. TNO at TDT2001: Language model-based topic detection. 2001.

[28] Martijn Spitters and Wessel Kraaij. Using language models for tracking events of interest over time. In *Proceedings of LMIR 2001*. Citeseer, 2001.

[29] Alessandro Vinciarelli. Noisy text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1882–1895, 2005.

[30] Turrell Wylie. A standard system of Tibetan transcription. *Harvard Journal of Asiatic Studies*, 22:261–267, 1959. ISSN 00730548, 19446454. URL `http://www.jstor.org/stable/2718544`.

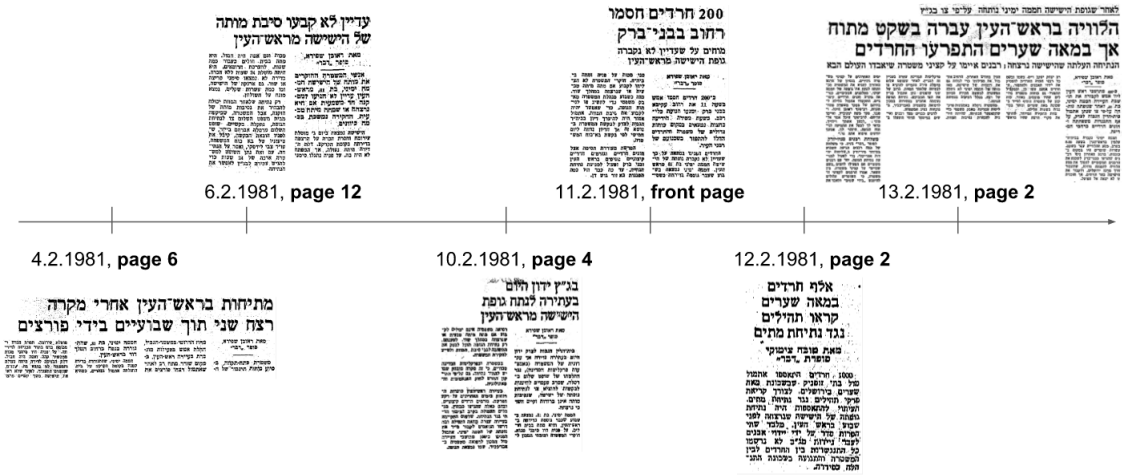
[31] JP Yamron, S Knecht, and P Van Mulbregt. Dragon's tracking and detection systems for the TDT2000 evaluation. In *Proceedings of Topic Detection and Tracking Workshop*, pages 75–80. Citeseer, 2000.

[32] Yiming Yang, Tom Pierce, and Jaime Carbonell. A study of retrospective and on-line event detection. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 28–36, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5. doi: 10.1145/290941.290953. URL `http://doi.acm.org/10.1145/290941.290953`.

[33] Alexander Zhicharevich and Nachum Dershowitz. Matching poorly recognized texts to a corpus (abstract). In *Israeli Seminar on Computational Linguistics (ISCOL)*, Bar-Ilan University, Ramat Gan, Israel, 2011.

A Appendix

Story Detection Example



Across Publishers Example

Maariv

Davar

Article Continuation Example

המשא־ומתן בשאלות העלאת השכר
ותוספת היוקר – יסוכם בשבוע הבא

העלאת השכר

Word Identification Errors Example

גל התייקרויות נוסף צפוי בימים הקרובים

תוך שבוע יתייקרו מאות מוצרי מזון

Article Segmentation Error Example

גל התייקרויות נוסף
צפוי בימים הקרובים
תוך שבוע יתייקרו מאות מוצרי מזון

ג :לי הת י ק ח _" ו_מ נוסף : ו צפ י ב»מים מ ק ר ו ב י
ם '

תור שבוע יתייקרו מאות מוצרי מזון י י _נאתפופדוחבלגלי .: ;:_' , . ' .'*_ ליבי'/ י
. "_ ; '

גל.התייקרויות _זמי"צפזי _גימיפ ' ,ה5ןדוביג3_, י • ntwnns_ מייקור _חפולד,החשמל •_ןה• צד _פ
. יצרני י : המזון;הכחיימ אמש,כל ייאלצו לרעלות _^_תלמחירי _מוצריחפ aM-aa_ אחה כעקבות י
חחתייק• רויות_»ל,מרכיגי,הייצור _משוניפ.כמו?כן יקרי כחש• ןן את _חשלזפ מוספת ה• יזקר .
ההערכה חחדשה היא. . ? _ןשתון ?שבמ יחד ' ;» _נימליפ ־ " «vxs ממנורת אינדונזית
.._פפפימאמ'.!תגלו »ל אי בפרחק .' 160' קיפ פן האזור »י_1_פ _יאווזז שבו._פבעת »ל_כ־»<>
גוסעיהי ייקרו.«א1_» פוצרי פזיז.בעיקר אלה שבהם מרביג ,הקמח.' הוא גבוה.'ההתייקרות•
של._אפשיפחיי־ בת את _הינדגיפ.לגוזק מחדש את _פחירוגיהפ .לחודש• סגרזאר.• אשד _רק_»
נ0ז,ביופז'. •אמש• saw :זמש.?ובעלי מאפיות 'יעדיפו לאפות לחפ מפד ?י ? " גיפ יקריפ ל • על •
לתפ אחיד,כפי שקיי!_לסגי ?נטה שגי • _* _נויין,V• דק _באפנעות' _סיקוח יעיל •אפשר ? יהיה ־
להבפיוו הספקתי בהיקף ה* דרוש של לחפ.אחיד.טרפ ביוד. אפ פשרד' _התמית >iv_ _להתארגל
לביצוע פיקות זח. • התארגנות ' ?הוזלת " .. ג י.. ושוצריפ . י ... יבואני .פוני?חשמל ומג קיר• פו
בברכה ־את _נעדי האוצר.אד פתי לפ ד יחלו ,פ הדגישה ' הפזזיריפ כי לצרכן טופברור , . ואפ , ,
ההוזלה תחול.גפ על _הפונריופ, הגפצאיפ במלאי.שלתפ..מניות אלד _ינפרמ .להתברר הייפ ,ומזו.
בעיה י נוספת היא י לגבי י לקי• חות..שנבר שילמו _פקדפות גדולות על חשבון.פקלסי _הטלור•
ז>ד,.._יהפקרריפ.._שרכשו .וטופ סופקו להט.לא י ברור.אם לקוחות אלה יהיו זכאים ־ _להוזלות.
היבואנים יתבעו מהאוצר,שגם _לקיהזת אלה ייהנו _מההוזלה.. :מלשבת _saos• בת"א _גמסי'
אפש,ני הלשנה'פוכגד,לסייע לאוצר בביצוע מדיניות ההוזלות _ולהבפיוז.שהיבואנים אפנם יוזילו את
פחירי הפוצריפ לני'נגיפ.

Wylie Transliteration of the Tibetan Alphabet

ཨ་	ཨི་	ཨུ་	ཨེ་	ཨོ་
a	i	u	e	o

ཀ་	ཁ་	ག་	ང་
ka	kha	ga	nga
ཅ་	ཆ་	ཇ་	ཉ་
ca	cha	ja	nya
ཏ་	ཐ་	ད་	ན་
ta	tha	da	na
པ་	ཕ་	བ་	མ་
pa	pha	ba	ma
ཙ་	ཚ་	ཛ་	ཝ་
tsa	tsha	dza	wa
ཞ་	ཟ་	འ་	ཡ་
zha	za	’a	ya
ར་	ལ་	ཤ་	ས་
ra	la	sha	sa
	ཧ་	ཨ་	
	ha	a	

Alignment example

List of Figures

List of Tables

הפקולטה למדעים מדויקים ע"ש ריימונד ובברלי סקלר

בית הספר בלווטניק למדעי המחשב

מציאת קשרים בין טקסטים היסטוריים

חיבור זה מוגש בתור עבודת מחקר לקראת תואר מוסמך במדעי המחשב על-ידי

דניאל לבנסקי

העבודה נכתבה בבית הספר למדעי המחשב

בהנחיית פרופ' נחום דרשוביץ

אוגוסט 2016

אב התשע"ו

תקציר

בתיזה זו, אנו מתעסקים עם שתי משימות מיצוי טקסט שיכולות לעזור במציאת קטעי טקסט קשורים בתוך מאגרי טקסט גדולים. המשימה הראשונה היא מציאת מאמרי עתונות הקשורים לאותו נושא (מתארים את אותו אירוע), והשנייה היא מציאת קטעי טקסט הקשורים בכך שיש להם מקורות משותפים (שימוש חוזר בטקסט).

עבור זיהוי נושא המאמר השתמשנו במודל וקטורי מרחבי המייצג את המאמרים. למרות טקסט רועש (כתוצאה מזיהוי תווים אופטי בעל שגיאות מרובות) הייתה הצלחה במציאת מאמרים דומים. השגנו שיפור בתוצאות של כ3% על ידי כך שהוספנו לייצוג של הכתבה מילים המופיעות בהקשרים דומים לאלו של המילים שזוהו בכתבה.

בבעיית שימוש החוזר בטקסט, השאלנו אלגוריתם למציאת כל ההתאמות המקורבות ברצפי DNA. התאמנו את השיטה כך שתתאים יותר לשפה טבעית על ידי כך ששינינו את יחידת הבסיס במחרוזת להברה במקום אות בודדת. שינוי זה שיפר גם את זמן הריצה וגם את איכות התוצאות בצורה משמעותית. בנוסף הצענו שיטה חדשה לדירוג מועמדים להתאמה שמשקללת את ציוני התאמת או חוסר התאמת ההברות על סמך החשיבות שלהן. שיטה זאת שיפרה את רלוונטיות של התוצאות המדורגות ראשונות.