SOFSAT: Towards a Set-like Operator based Framework for Semantic Analysis of Text

Shubhra Kanti Karmaker Santu[1], Chase Geigle[1], Duncan Ferguson[1], William Cope[1], Mary Kalantzis[1], Duane Searsmith[1], and Chengxiang Zhai[1]

University of Illinois Urbana-Champaign
{karmake2, geigle1, dcf, billcope, kalantzi, dsearsmi, czhai}@illinois.edu

ABSTRACT

As data reported by humans about our world, text data play a very important role in all data mining applications, yet how to develop a general text analysis system to support all text mining applications is a difficult challenge. In this position paper, we introduce SOFSAT, a new framework that can support set-like operators for semantic analysis of natural text data with variable text representations. It includes three basic set-like operators—TextIntersect, TextUnion, and TextDifference—that are analogous to the corresponding set operators intersection, union, and difference, respectively, which can be applied to any representation of text data, and different representations can be combined via transformation functions that map text to and from any representation. Just as the set operators can be flexibly combined iteratively to construct arbitrary subsets or supersets based on some given sets, we show that the corresponding text analysis operators can also be combined flexibly to support a wide range of analysis tasks that may require different workflows, thus enabling an application developer to "program" a text mining application by using SOFSAT as an application programming language for text analysis. We discuss instantiations and implementation strategies of the framework with some specific examples, present ideas about how the framework can be implemented by exploiting/extending existing techniques, and provide a roadmap for future research in this new direction.

Keywords

Text Mining, Semantic Analysis, Intelligent Text Analysis, Semantic Operator for Text

1. INTRODUCTION

Text data broadly include all kinds of data generated by humans in the form of natural language text, which can exist in the form of written text data or transcribed text data based on human speeches. Written text data include all kinds of information on the Web, such as web pages, news articles, product reviews, and social media, enterprise text data, emails, and scientific literature, while transcribed text data can be produced from many video data and speeches. Since text data can be regarded as data generated by human sensors describing the observed world, they can be naturally combined with data generated from all kinds of physical sensors to provide a more complete view of the observed world and enable more effective data mining via joint analysis of text and non-text data [43]. As humans are involved in virtually all "big data" applications, text data are generally available in all application domains, making them valuable for applications in all the domains.

The unique value of text data from the perspective of data mining can be reflected in the two important differences between human sensors and physical sensors: humans are subjective and far more intelligent than physical sensors. The inevitable subjectivity means that the text data reported by humans contain not only the observed (objective) information about the world, but also their subjective opinions, making text data an extremely useful source of data for discovering (and understanding) people's attitudes, opinions, and preferences, which is needed in optimizing all kinds of decisions related to people, ranging from making effective and acceptable public policies by governments, to providing personalized tutoring materials to students by teachers, and to effective advertising of products to people on the Internet by companies. Human intelligence enables humans to effectively process and digest what has been observed based on all kinds of background knowledge, and thus the text data reported by human sensors are not only generally meaningful, but also directly useful as knowledge; in this sense, even a small amount of text data can also be very useful if computers can understand the data accurately. For example, while analysis of data sent through a computer network may reveal an abnormal pattern that might suggest the possibility of a virus spreading on the network, a few explicit comments about the virus made by system administrators of the network may directly report a suspected virus or help confirm a virus.

Unfortunately, text data are expressed in natural languages which are invented for humans to use and thus not "computer-friendly," making it extremely challenging for computers to understand text data precisely. Indeed, despite great progress has been made in the natural language processing (NLP) field, computers are still far from being able to accurately understand unrestricted natural language; as such, how to analyze and mine big text data effectively and efficiently is a pressing difficult challenge. Thus, involving humans in a loop of interactive text mining is essential, but how can we develop a general system that can support users in potentially many different text mining applications? The answer to this question at least partially depends on

whether/how we can define a "text mining language" that can allow a user to flexibly specify potentially many different workflows as needed in different applications in a similar way to how users use an application language such as SQL for querying a database in many different ways. We address this question by drawing insights from set theory.

Set theory provides a theoretical foundation for constructing (arbitrary) new sets from given sets by applying different operators. For example, the intersection operation (denoted by $\cap$) takes two sets as input and returns the set of objects present in both sets, while the union operation (denoted by $\cup$) takes two sets and returns the set of all objects present in either (or both) of the two sets. The difference operator (denoted by $-$) takes two sets and returns the set of unique objects present in the first set that are not also present in the second set. Because these operators have compatible data types (i.e., we can apply any operator to the results generated from applying any other operators), we can flexibly combine them to define more complex operations on potentially a large number of sets. Thus even with just these three basic operators, we can already support potentially infinitely many different complex operations. In other words, we can "program" with these individual operators to support complex set construction tasks.

Can we define similar operators for semantic analysis of text data? That is, can we define a number of set-like operators that are "sufficient" for supporting potentially many different text analyses? If we can do that, we would be able to define a text analysis programming language based on a small number of semantic operators on text that users can then use to flexibly program potentially infinitely many specific workflows for text analysis application tasks. A general text analysis system can thus be implemented to support users in performing such text analysis tasks.

Such a system would be extremely useful as text data play an increasingly important role in all big data application domains. As people communicate in natural language all the time, text data are produced constantly wherever people are present, which means that text data play an important role in all domains of data mining applications. However, as mentioned before, due to the difficulty in natural language understanding by computers, how to effectively mine and analyze text data remains a significant open challenge and involving humans in the loop is generally required both to leverage human intelligence in an analysis task and to allow humans to control the analysis flexibly as needed. Given that the application needs vary significantly across different domains, an important question is thus: can we design a general system that can support many different applications?

The analysis of set operators above motivates us to address this question by designing a programming language for text analysis; just as a general programming language such as C++ or Java is flexible to allow us to write infinitely many different programs each solving a different problem, our goal here is to design a special programming language that can be used to write infinitely many different text analysis programs each solving a different text analysis task. Specifically, we propose SOFSAT, a general text analysis framework that can support set-like operators for comparative analysis of natural language text data. We introduce three basic set-like operators in this framework—TextIntersect, TextUnion, and TextDifference—which are the natural text analogues of the original set operators they are named after. SOFSAT provides a single unified framework, which, once implemented, would be able to support an infinite number of different applications by combining the three individual basic operators. As in the case of a general programming language, frequently used sequences of operators in SOFSAT can also be treated as a "compound operator" which can be made available to users through a library. Furthermore, SOFSAT can be potentially extended to include additional user-defined operators as needed.

To see why SOFSAT can be potentially useful for many applications, it is instructive to consider the following examples.

1. **Review Analysis:** Consider the peer review practice widely adopted in assessment of complex assignments, especially in online education systems. To help an instructor or student understand the common comments made by all the reviewers of a student work, we only need to apply the TextIntersection operators to all the reviews. The unique perspective of Reviewer R can be obtained by applying a TextDifference operator to the result of TextUnion of all the other reivews. Clearly similar analysis can also be done for reviews of conference or journal submissions as well as grant proposal submissions.

2. **Bias Analysis of News:** Consider the task of analyzing potential bias in news reporting. Letting A and B be two news articles reporting the same event from two news agencies, $A-B$ or $B-A$ would be useful for understanding any potential bias in each article, whereas comparing the articles reporting the same event in different time periods would help understand the evolution of the event.

3. **Knowledge Discovery from Literature:** SOFSAT can also be used to mine biomedical literature to potentially reveal interesting hypotheses. For example, the well-known example of discovering the hypothesis of fish oil for treating Raynaud's syndrome using pure text mining [39] can be easily supported by the proposed set-like operators. Specifically, we can first retrieve relevant text information from literature articles about a supplement such as "fish oil" (denote this text as X), and then retrieve relevant information about Raynaud's syndrome (denoted by Y). Once we have X and Y, we can apply TextIntersection to see what text information is shared by X and Y and assess whether there is any interesting connection between "fish oil" and Raynaud's syndrome.

How exactly should the three text analysis operators be designed and implemented? What architecture should be used to implement a general system based on SOFSAT? How can we use the framework to solve some representative real-world applications? The purpose of this position paper is to introduce the SOFSAT framework and take an initial step toward addressing such general questions about text analysis. We hope this will facilitate the actual design of a programming language for text analysis based on set-like operators and actual implementation of a compiler or interpreter of the language, and eventually a deployment of the language and system to allow many text applications to be easily developed across diverse application domains.

Specifically, in the rest of the paper, we will first introduce and discuss the SOFSAT programming language followed by some representative application scenarios of the language in section 2. We then discuss the overall architecture of the framework and instantiation guidelines in section 3. Next, in section 4, we provide a roadmap for future research in this direction. Finally, we briefly discuss related work in section 5 and conclude with section 6.

2. SOFSAT: A TEXT ANALYSIS LANGUAGE

We first present SOFSAT as a general application programming language for supporting a variety of text analysis applications. Our main motivation is to have such a language so that we can have a general text mining system for supporting a wide range of text analysis applications by allowing application developers and users to program different workflows needed for different applications using the same programming language. The benefit is that once we have such a general system, it can be deployed immediately in all application domains to support many different text mining tasks, accelerating applications of text mining. In some sense, the benefit would be similar to that of SQL language for database applications. Similar to SQL, we also want our text analysis language to be declarative so that we can potentially separate the optimization of an implementation from the application semantics. We now present the SOFSAT language in more detail.

2.1 Definition

As a programming language, SOFSAT is conceptually simple as it is completely analogous to the set operators with three basic operators for text objects: TextIntersect, TextUnion, and TextDifference which can be combined with each other flexibly provided that the types of the data that those operators are applied to are compatible. However, text analysis is a sophisticated task and different tasks may require a different way to represent text data. Thus, all the operators must also be applicable to any preferred representation of text data.

For example, the simplest representation is to use set theory directly by assuming the "objects" are keywords extracted from pieces of text, and we can then perform set operations on these sets of keywords. Such a simple representation has the advantage of being efficient and is often also sufficient for simple text analysis tasks, notably topic-related analysis. However, such a representation is deficient for a number of reasons. First, it assumes that each word is independent of the others, but in natural languages, many words are semantically related, and it is desirable to capture those semantic relations. Second, it fails to capture the relative ordering of words in the text which may also be important as the order may affect the meaning (e.g., " John gave a book to Mary" is very different from "Mary gave John a book"). Finally, it also ignores duplicated words—a word is ether present in the set or absent, as there is no model for "degree of membership."[1] However, a word occurring very frequently in an article may be regarded as better representing the content of the article than a word that occurred just once.

To improve over such a simple method, it is more desirable to define the operators at the level of appropriate *semantic* representation of text data, and implement them based on various representation transformation functions. Thus, a general framework must accommodate different ways to represent text data.

A sophisticated text analysis task also often requires integration of analysis using multiple representations. To accommodate this need, SOFSAT must also allow operators working on different representations to be combined with each other. We solve this problem by introducing two additional operators (we call them transformation functions) to map natural language text to and from a representation, respectively. One of them is called *TextInterpretation* and would map text to a given representation; the other is called *TextGeneration* and would map a representation (back) to text. With these two additional operators, we can map one representation to another by going through text as a "bridge", thus enabling operators defined on different representations to be combined with each other.

The *TextInterpretation* and *TextGeneration* operators also enable derivation of different representations from the same text data as needed as well as facilitates interpretation of any computed intermediate representation by users by transforming an intermediate representation to text.

The following summarizes the key components in the SOFSAT text analysis language:

Representation of Text: We assume that there is a finite set of text representations this framework can handle and we denote this set by $R = \{r_1, r_2, \ldots, r_n\}$ where n is the cardinality of set R and $r \in R$. For more details on different representations of text, see section *3.2.1*.

TextInterpretation operator: The framework provides a *TextInterpretation* operator, also called representation transformation function ψ_i, corresponding to each representation r_i where ψ_i transforms a natural language text into the representation r_i. Thus, the set of representation transformers is $\Psi = \{\psi_1, \psi_2, \ldots, \psi_n\}$ and there is a one-to-one correspondence between R and Ψ.

TextGeneration operator: SOFSAT also provides a *TextGeneration* operator $\hat{\Psi} = \{\hat{\psi}_1, \hat{\psi}_2, \ldots, \hat{\psi}_n\}$, which is essentially a set of reverse transformation functions with a one-to-one correspondence between the elements of Ψ and $\hat{\Psi}$. While a ψ function transforms natural language into some internal representation $r \in R$, a $\hat{\psi}$ function transforms the internal representation back into the natural language form.

Set-Like Operators: Finally and most importantly, the framework provides a finite number of set-like operators that can be applied to conduct comparative analysis of multiple pieces of text generally represented using a particular representation from the representation set R. The operators include set-like operations such as TextIntersect, TextUnion, and TextDifference of natural language text. Note that the specific implementations of these operators will vary based on the particular representation of the text (see section *3.2.2*).

As in set theory, once implemented in an interactive analysis system, such operators can be combined flexibly by users to perform potentially very complex semantic analysis tasks as we will further discuss next.

[1]While multi-sets can model duplicated objects, they still assume object independence and fail to preserve relative ordering.

2.2 Cascading Multiple Operations

One beneficial feature of SOFSAT language is that multiple text segments as well as operators can be processed in a cascading fashion. Figure 1 shows such an example. Here, operator ξ_1 is applied on T_1 and T_2 represented in r_1 form to generate $T_3 = T_1\ \xi_1\ T_2$. On the successive iteration, T_3 is passed along with a new text T_4 using representation r_2 to generate $T_5 = T_3\, \xi_2\, T_4$. This kind of cascading operation can go on infinitely and represent complex semantic operations on multiple text segments.

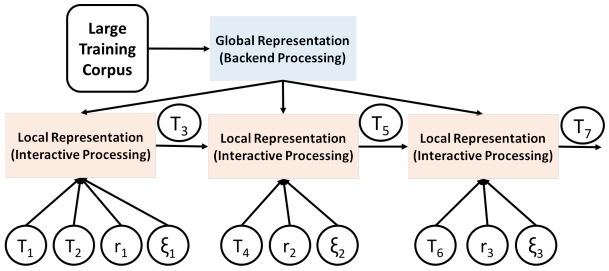

Figure 1: An example of cascading multiple operators. Here, we obtain a new text object T_3 by applying ξ_1 to T_1 and T_2. This is then fed as the first input to another set-like operator ξ_2 along with T_4 to generate T_5, which is again used with a third operator ξ_3 with T_6 to finally produce T_7.

2.3 Examples of Applications

Even with just a few operators, SOFSAT can be regarded as providing a simple programming language for writing an application program for text analysis tasks since there are infinite possibilities of iteratively combining operators to process any given set of text data sets. Moreover, frequently used sequences of operators can be stored as subroutines, which can be later easily reused by other users. In an interactive analysis environment, a user can wait to see the intermediate results (which can be stored in a workspace) and then decide which operator to use next, offering maximum flexibility for customizing the workflow as needed. The generality of SOFSAT allows it to support many different applications; below we very briefly present three different applications as specific examples.

Education: Peer assessment of student papers is now commonly used in MOOCs and other educational settings [33; 37]. In such systems, peers submit their papers to reviewers (also peers) who generate individual reviews for the paper that include not only scores, but also written comments connected with the criteria in the grading rubric. Concerns remain regarding the variability and thus reliability of such reviews [13], but SOFSAT can help by revealing the common concerns raised by multiple reviewers using TextIntersection (of multiple reviews). TextDifference can also be useful for revealing how a student has revised an essay by comparing the original version with the revised one.

Next, we demonstrate another useful application of the difference operation: let us assume there is a course being taught at a university where the instructor posts an assignment for the students which requires answer in natural language form. Also assume that the students can submit two different versions of their answer, namely, V_1 and V_2. However, after they submit V_1, the instructor give them some feedback and based on the feedback received, they submit V_2. Now, from the instructors perspective, it is interesting to see what changes were made in V_2 with respect to V_1. A deeper thinking would also reveal that $V_2 - V_1$ would allow us to see the additions made in version V_2, while $V_1 - V_2$ would give us the deletions made. Thus, $(V_2 - V_1) \cup (V_1 - V_2)$ would represent the total changes made in version V_2 with respect to V_1. Thus, by applying these operators, the instructor can quickly get a sense about the changes made in V_2 that would help him/her to grade V_2 more efficiently.

Now, lets look at a more involved case with a corresponding complex workflow. Suppose the instructor, instead of analyzing the additions made in the second version by a single student, wants to analyze the common additions made in version V_2 by a group of n students. In the language of SOFSAT, this can be represented as:

$$(V_2^1 - V_1^1) \cap (V_2^2 - V_1^2) \cap ... \cap (V_2^n - V_1^n)$$

In addition to that, suppose the instructor wants to see if there are new patterns in the common additions made by students in the current semester compared to the students in the previous semester. To express this in the language of SOFSAT, let us denote the student submissions in the current semester by V and student submissions in the previous semester by W. Then, common additions made by students in the current semester that were not made by the students in the previous semester can be expressed as follows:

$$\begin{aligned}&\left\{(V_2^1 - V_1^1) \cap (V_2^2 - V_1^2) \cap ... \cap (V_2^n - V_1^n)\right\} \\ &\quad - \left\{(W_2^1 - W_1^1) \cap (W_2^2 - W_1^2) \cap ... \cap (W_2^n - W_1^n)\right\}\end{aligned}$$

Now, let us look at an another complex case. Suppose that a research paper has been reviewed by four different reviewers and let the individual reviews (in text) be represented by A, B, C and D, respectively, and our goal is to generate a meta-review by combining the four reviews in some way. This is challenging for a few reasons: (1) it is possible that $A \cap B \cap C \cap D$ is an empty set, i.e., there is nothing that is common across all four reviews, and (2) there are often many comments mentioned by a single reviewer that are not relevant to incorporate into a meta-review. Thus, one reasonable solution is to incorporate all the concerns raised by at least two reviewers. SOFSAT can achieve this goal through the following simple operation:

$$(A \cap B) \cup (B \cap C) \cup (C \cap D) \cup (A \cap D) \cup (B \cap D) \cup (A \cap C)$$

Thus, in general, set-like operators in SOFSAT would allow us to do intelligent processing of text data which will enable new application tasks as well as enhance the existing application tasks.

Health Informatics: SOFSAT can be applied to compare clinical notes in patient records so as to reveal the changes in a patient's diseases condition or perform comparative analysis of patients with the same diagnosis. For example, TextDifference can be applied to the clinical notes from two consecutive visits of a patient to assess the effectiveness of the treatment provided to the patient in between the two visits. TextIntersection can then be further applied to the results of TextDifference from all the patients provided with the same treatment to understand the overall impact of the treatment. For an example, assume that four patients A, B, C, and D have the same medical condition and have gone through the same treatment plan. Also, let A_i denote the

clinical note of patient A before the treatment started and A_f be the clinical note after the treatment was provided. Similarly, B_i, C_i, D_i and B_f, C_f, D_f denote the before and after treatment clinical notes respectively for patient B, C and D. Now, to understand the effectiveness of the provided treatment, the following function can be invoked using the SOFSAT framework:

$$(A_i - A_f) \cap (B_i - B_f) \cap (C_i - C_f) \cap (D_i - D_f)$$

Now, further assume that all the patients took a particular medicine during this treatment period. The doctors might be interested to know if that particular medicine has some common side-effects on its patients. These side-effects can easily be extracted using the following SOFSAT expression:

$$(A_f - A_i) \cap (B_f - B_i) \cap (C_f - C_i) \cap (D_f - D_i)$$

Note that, effects and side-effects of treatment are essentially the removal of existing symptoms and addition of new symptoms after going through the treatment plan. Thus, SOFSAT would be very useful identify these removals and additions to understand the effects and side-effects of a treatment plan.

News Bias Analysis: Assume that there are two news agencies reporting the same event, and that each news agency has some political bias which is reflected to some extent within the articles they write. If A and B are the two news articles reporting the same event from two different news agencies, then a TextIntersection operation $A \cap B$ would provide all the common statements which are reported by both A and B (which are likely revealing facts about the event); in other words, $A \cap B$ is expected to surface the facts about the event they are reporting. On the other hand, the TextDifference operator $A - B$ would reveal any bias of A in reporting the event, and $B - A$ the bias of B. Finally, $A \cup B$ can provide a summary of all the statements made by either of A and/or B.

Let us take a look at a more complicated case. Assume that we now have three news agencies instead of two. Again, they are reporting about the same event and the corresponding text is denoted by A, B, and C, respectively. To find out the bias of each agency in reporting the event, it is necessary to find out all unique statements reported by each agency that were not reported by any of the other two agencies. Thus, the bias of A can be found by the SOFSAT expression: $A - (B \cup C)$. Finally, to find out all such biased statements from any of the reports, we can use the following expression:

$$\{A - (B \cup C)\} \cup \{B - (A \cup C)\} \cup \{C - (A \cup B)\}$$

In summary, SOFSAT can support many interactive text analysis applications. Specially, if a sequence of operators are often combined by users, they can form a "subroutine" to allow future users to call such a subroutine without using the tedious low-level operators every time. This demonstrates the potential of SOFSAT for *programming* with these operators that will simplify accomplishing very complex tasks.

3. IMPLEMENTATION OF SOFSAT

The proposed SOFSAT language can be potentially implemented in many different ways. In this section, we discuss some possibilities, highlighting the need for a combination of a Backend and a Frontend Interactive Module.

3.1 Architecture

In order to fully leverage existing research results on text representation and transformation, we believe that the SOFSAT system should have a *Backend* (offline) module and an *Interactive* (online) module as illustrated in Figure 2. Such a design is based on the following observations:

1. **Sparsity:** One particular issue with text data is the sparsity associated with it, especially in case of short text. As the primary goal of SOFSAT is to enable non-experts to explore text pieces of arbitrary lengths, SOFSAT must be able to deal with short text frequently. One way to deal with the sparsity challenge, especially in the case of short text, is to exploit publicly available large text corpora to extract complicated semantic relations among words and augment these relations along with the input text data to reduce the sparsity problem. However, extracting complicated semantic relations from large text corpora is computationally expensive and time consuming, making it unsuitable for interactive analysis of text data. Thus, it is reasonable to split SOFSAT into two modules, namely, *Backend* (offline) module and *Interactive* (online) module where the *Backend* module would precompute the semantic relations of different words beforehand in an offline fashion and then, at query time, the *Interactive* module will augment the input text data with semantic relations learned by the *Backend* module to create a more dense representation of the input text and further apply the set-like operators on that dense representation.

2. **Background Knowledge:** Another issue associated with text data is the background knowledge it assumes on the "consumer" of the text data. Background knowledge consists of knowledge about different things such as entities, locations, historical events, cultural practice that are not explicitly articulated in the text itself. For example, any video-game lover reading a text article containing the word "Xbox" would immediately realize that it is a gaming device manufactured by Microsoft, although the word "Microsoft" may not be present in the actual text. Similarly, any soccer lover reading a text article containing the bigram "El Classico" would immediately realize that its a soccer game between two popular clubs, i.e., Barcelona and Real Madrid. However, it can be the case that none of the words "Barcelona", "Real Madrid", or "Soccer" are actually present in the text. The writer of the article in this case assumes that the reader knows what "El Classico" is and how it is related to "Barcelona", "Real Madrid" or "Soccer". This is a common phenomenon with every text document that is written by a human reporter targeting a particular reader community as in general, in order to increase the efficiency of communication. Writers tend to omit much of the background knowledge that they can assume that the consumer of the text data already possesses.

   Thus in order to understand text data, it is also desirable for computers to incorporate this background

knowledge. Publicly available large text corpora can again help in this case by allowing computers to extract useful information and build a knowledge graph that can allow the computer to more intelligently make sense of human written text articles. The *Backend* (offline) module can again take the responsibility of precomputing such knowledge graphs and provide them to the *Interactive* (online) module as needed. The general justification for separating a backend from a frontend is to enable both complex processing of text data needed for incorporating background knowledge and semantic interpretation as well as efficient interactive analysis needed for many text mining applications.

We now describe how the two synergistic modules (i.e., Interactive Module and the Backend Module) work in more detail.

3.1.1 Interactive Module

The Interactive Module is the primary module where users interact with the framework. It takes one or more natural language text(s) as input and applies different set-like operators. Without loss of generality, assume that the Interactive Module takes as input two pieces of natural language text of arbitrary lengths. Let us denote these two pieces of text by T_1 and T_2. The Interactive Module also takes two other inputs: the representation of the text r and the intended set-like operator ξ. Now, based on the input r, the framework selects the right representation transformation function (denoted by ψ^r) and applies ψ^r on both input texts T_1 and T_2 and outputs the local representation $L(T_1)$ and $L(T_2)$, respectively, where, $\psi^r(T_1) = L(T_1)$ and $\psi^r(T_1) = L(T_2)$. We call these the *local representations* to distinguish them from the *global representations* which we discuss in the next section. The next task of the Interactive Module is to take these two local representations $L(T_1)$ and $L(T_2)$ and apply the operator ξ to produce $L(T_1 \, \xi \, T_2)$, which is the local representation of $T_1 \ \xi \ T_2$. Finally, to get back the natural language text, the Interactive Module applies the reverse transformation function $\hat{\psi}^r$ on the result $L(T_1 \, \xi \, T_2)$ which yields our desired $T_1 \ \xi \ T_2$.

3.1.2 Backend Module

While the Interactive Module can apply the set-like operators and generate the intended results by itself, it suffers from the sparsity problem associated with any natural language text, especially short text segments. For example, two text segments T_1 and T_2 may represent two independent descriptions of the same event, but there may be very few exactly overlapping words between T_1 and T_2. However, at the semantic level, they might be very similar. To capture such semantic relations between words, which is very hard to learn from two small pieces of text, we need to exploit large available text corpora to learn these semantic relations from global co-occurrences of words. Training with large text corpora requires longer time, demanding a Backend Module that can pre-compute different global representations of words based on the co-occurrences within large training corpora. These global representations can then be directly applied on top of the local representations created by the Interactive Module to address the sparsity problem. Note that both the Interactive and Backend Modules offer the same set of representations $R = \{r_1, r_2, \ldots, r_n\}$. However, the Backend Module learns these representations from the global corpus (we represent it by G_r), while the Interactive Module computes them based on the input text data (we represent it by $L(T)$).

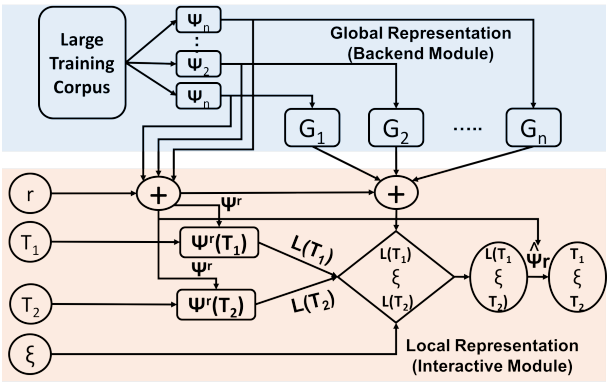

Figure 2: A visual overview of the SOFSAT framework. The backend module (top) consists of a set of text transformation functions $\psi_{1:n}$ and their corresponding global text representations $G_{1:n}$. At query time, the interactive module (bottom) takes a query in the form of two text objects (T_1 and T_2), a desired representation r, and set-like operator ξ, and uses the transformation corresponding to r to obtain a local text representation, which is then used to obtain a result for the operator ξ in the local text representation, which is finally transformed back into natural language text via the inverse text transformation function (The TextGeneration operator), $\hat{\psi}^r$.

3.2 Implementation of Operators

Once the architecture is fixed, the next task is to implement various operators, which we discuss in this section. Our discussion is brief as our goal is to lay out the possibilities rather than going in depth in any specific direction, which would be out of the scope for this position paper. We hope the ideas we discussed here are sufficiently informative to stimulate more research work in this direction.

3.2.1 Representation of Text

There is a large body of literature surrounding text representations [17; 16]. Bag-of-words is the simplest representation, where a document is represented by the frequency counts of its words. However, there are a wide variety of other representations including the Vector Space Model [35], Binary Representation [26], Ontology Based Representations [22], N-Gram Models [9], Topic Models [42; 8], Graphical Models [11], Word Embedding Vectors [30], Paragraph Vectors [14] etc. Different representations may be advantageous for different kinds of analysis applications. The benefit of using our proposed framework is that the user can select any representation for text according to their choice and can also use different representations at different stages of the cascade (Figure 1).

3.2.2 Implementing Set-Like Operators

The implementation of set-like operators can in general be categorized into two different types: *retrieval based* and *generation based*. The retrieval based implementations of an operator basically selects/retrieves relevant words/sentences from the input text to construct the output text. The Vector Space Model [35], N-Gram Language Model [9], Topic

Model [42; 8], and Graphical Model [11] representations can be handy for retrieval based implementations.

Generation-based implementations would automatically generate text according to the operator being applied, thus they are not restricted to the keywords provided inside the input text. Sequence generation models like recurrent neural networks (LSTMs [20]), Hidden Markov Models [34], etc. can be exploited for generation based implementations. Finally, a hybrid implementation is also possible that combines both retrieval-based and generation-based implementations.

In connection to the existing works related to this field, the *TextUnion* operation is similar to the idea of sentence fusion, which has been vastly studied in the literature [5; 6; 15; 28]. The *TextIntersection* operation can be thought of a special case of sentence fusion, where output must only contain the information present in all input texts [40]. Levy et al. modeled retrieval based Sentence Intersection via Subtree Entailment [24]. All these ideas can contribute to the implementation of the set-like operators and by allowing flexible combination of these operators, SOFSAT provides a general framework which would be a very powerful interactive text mining tool.

4. A ROADMAP FOR FUTURE WORK

The SOFSAT framework opens up many interesting new directions for future research, which we discuss in this section.

1. **Full specification of the SOFSAT text analysis language:** The first direction is to study how to define an appropriate SOFSAT text analysis language, which can then be the basis for implementing a system to support the analysis functions provided by the language. While the three basic operators are the core functions for text analysis, in order to fully support the workflow of a text analysis application, the SOFSAT text analysis language must also be able to support operations such as how to load the text data from disks into the system, how to store intermediate analysis results, and how to save any interesting analysis results to the disks. Of particular importance is the support of workflow management so that a user can keep track of any intermediate results and further combine results as needed. We thus envision that the SOFSAT language would need to support variables of different types corresponding to different ways to represent text data. For example, raw text representation may be one type, bag-of-words representation may be another, while topic-based representation can be yet another type. The language should allow for extension of data types so as to accommodate new ways to represent text data. The variables can then be used to hold intermediate results generated from applying various operators. Naturally, some basic input and output operators should also be supported. Furthermore, subroutines can be defined so that frequently used sequences of operators can be captured as a function; once a function is defined, it can be called as needed to invoke a whole sequence of operations. Existing knowledge about how to design a programming language can be leveraged to design the SOFSAT language. It is desirable to first design a very basic SOFSAT language that can support three set-like operators with a basic text representation such as bag-of-words representation, but would otherwise be as "small" as possible. Such a basic SOFSAT language would minimize the amount of effort needed to evaluate the promise of the overall idea of using a fixed set of operators to support potentially many different text analysis applications.

2. **Implementation of a basic version of SOFSAT:** Once a basic SOFSAT language is specified, the next task is to implement a system to support such a basic language. Given the unpredictable nature of the workflow of text analysis applications, we may initially implement an "interpreter" system that can interactively support a user in text analysis where the system would execute a command given by the user expressed in SOFSAT language. This allows a user to see what the results look like from some previous steps of analysis before deciding what to do next, thus providing the needed flexibility to adjust the workflow dynamically. The implementation involves implementing those three basic operators using appropriate text representations. A very first version of the SOFSAT system can be based on a simple, yet powerful representation, for example, the bag-of-words representation.

3. **Evaluate SOFSAT with multiple applications:** Once a basic SOFSAT system is implemented, we can use the system to evaluate the general idea of the SOFSAT framework – allowing users to "program" text analysis applications by using the three set-like text processing operators. Besides the several specific application scenarios discussed earlier in this paper, it can also be used for many other applications. The SOFSAT system can be made open source to allow many users to test it with many real world text analysis applications. The feedback from those users would be extremely useful for further improving the basic system; in particular, it would inform the design of future versions of the language where advanced operators may be added. Without application feedback, however, it may be unclear what kind of advanced operators are most useful.

4. **Implementation of advanced operators:** Based on the feedback from testing the basic SOFSAT with many applications, we can further design and implement potentially many advanced operators, mostly corresponding to more advanced text representation than the bag-of-words representation. The implementations of those advanced operators will widely vary case by case based on the specific representations and technical details of methods. Each operator may demand a separate investigation since it may raise a novel challenge associated with performing a certain kind of semantic analysis of text data. With iterative testing and improvement, the eventual goal would be to materialize SOFSAT with a rich set of set-like operators that would support a wide variety of text representations, which can then be released as a general tool that can be used in many different application domains to support semantic analysis of text data.

5. **Optimization of SOFSAT system:** Finally, since SOFSAT is a declarative language, it has an important advantage in optimizing the SOFSAT system as the application logic and the implementation detail

can be separated. This is similar to the benefit of a relational data model that enables the separation of database query semantics from the actual execution of a database query. For example, if the system can see a whole sequence of operators, it may attempt to optimize the execution of those operators so as to maximize the efficiency without compromising the quality of analysis results. For example, in the case of commutable operators, intersection operators may be executed first to reduce the size of the candidate text objects in early stage so as to make it more efficient to further process the text objects using other operators later. Existing work on database query optimization can provide much insight about how to optimize SOFSAT system.

5. RELATED WORK

There has been so much work done on text mining and analytics that even a complete summary of the major lines of research goes beyond the scope of this paper; the reference [3] provides a comprehensive review of most of the research work in this area. From practical viewpoints, many text mining toolkits are available, including but not limited to Lucene [1], MeTA [27], NLTK [7] etc. To the best of our knowledge, this paper is the first to propose a formal general framework for potentially designing a programming language and implementing a system to support flexibly many different text analysis applications using a finite number of basic operators. In some way, those operators resemble the operators supported by a declarative query language for a database (like SQL [10]), making our work related to the Relational Data Model [12]. As the Relational Data Model provides a common foundation for database querying tasks, SOFSAT also provides a common foundation for many text analysis tasks.

Measuring the semantic similarities among words has long been a popular research topic among the NLP community [25; 23; 36; 4; 29; 19; 32]. Recently, researchers have also focused on how the word level similarities can be extended to sentence level similarity measures [38; 41; 2; 31; 18]. However, what is missing is a general framework or tool that can exploit these semantic relations to support intelligent text processing and comparison. This is the goal of our proposed SOFSAT framework.

Many powerful general text analysis algorithms have been proposed, notably those based on probabilistic topic models [21; 8]. They can be used to discover topics from text data and analyze variations of topics. These algorithms can be used as a basis for representing text data and thus can be potentially incorporated into SOFSAT as a way to provide an alternative representation of text, which further enables incorporation of set-like operators defined on such an alternative representation. Topic models have also been proposed for comparative analysis of text data; SOFSAT provides an alternative way of doing similar analysis with much more flexibility. However, it is possible that some of those customized models for comparative analysis may be more effective for specific analysis tasks than the general SOFSAT. Such a tradeoff between generality and effectiveness for a specific task is inevitable, but it is generally infeasible to enumerate all the different kinds of analysis tasks to develop a customized algorithm, and a main benefit of a general framework such as SOFSAT is its applicability to a wide range of applications, enabling many applications to be developed using a single framework easily. As we identify a specific kind of application, we may further develop more effective, customized analysis algorithms for that particular kind of application. In this sense, the general framework is complementary with those specific advanced algorithms. They can also be potentially combined in a hybrid system.

6. CONCLUSIONS

In this paper, we proposed a new general framework (SOFSAT) with set-like operators that can potentially support a wide range of text analysis applications by allowing for flexible combination of multiple operators to iteratively analyze arbitrary text data sets. We presented the general framework, discussed different ways to instantiate the framework, proposed an architecture for implementing an interactive text analysis system based on SOFSAT, and discussed a few specific applications. We laid out a roadmap for future work and hope that this position paper would stimulate research in this novel direction so as to accelerate widespread applications of text data mining.

7. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under grant: "Assessing 'Complex Epistemic Performance' in Online Learning Environments" (Award 1629161).

8. REFERENCES

[1] Apache lucene. https://lucene.apache.org/. Accessed: 2018-05-14.

[2] P. Achananuparp, X. Hu, and X. Shen. The evaluation of sentence similarity measures. In *International Conference on data warehousing and knowledge discovery*, pages 305–316. Springer, 2008.

[3] C. C. Aggarwal and C. Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[4] A. D. Baddeley. Short-term memory for word sequences as a function of acoustic, semantic and formal similarity. *Quarterly journal of experimental psychology*, 18(4):362–365, 1966.

[5] R. Barzilay and K. R. Mckeown. *Information Fusion for Multidocument Summerization: Paraphrasing and Generation*. PhD thesis, Columbia University, 2003.

[6] R. Barzilay and K. R. McKeown. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328, 2005.

[7] S. Bird and E. Loper. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics, 2004.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[9] W. Cavnar. Using an n-gram-based document representation with a vector processing retrieval model. *NIST SPECIAL PUBLICATION SP*, pages 269–269, 1995.

[10] D. D. Chamberlin and R. F. Boyce. Sequel: A structured english query language. In *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, pages 249–264. ACM, 1974.

[11] B. Choudhary and P. Bhattacharyya. Text clustering using universal networking language representation. In *Proceedings of Eleventh International World Wide Web Conference*, 2002.

[12] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[13] B. Cope, M. Kalantzis, S. McCarthey, C. Vojak, and S. Kline. Technology-mediated writing assessments: Principles and processes. *Computers and Composition*, 28(2):79–96, 2011.

[14] A. M. Dai, C. Olah, and Q. V. Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, 2015.

[15] K. Filippova and M. Strube. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185. Association for Computational Linguistics, 2008.

[16] C. Geigle, Q. Mei, and C. Zhai. Feature engineering for text data. In G. Dong and H. Liu, editors, *Feature Engineering for Machine Learning and Data Analytics*, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, pages 15–45. CRC Press, 2018.

[17] B. S. Harish, D. S. Guru, and S. Manjunath. Representation and classification of text documents: A brief review. *IJCA, Special Issue on RTIPPR (2)*, pages 110–119, 2010.

[18] H. He, K. Gimpel, and J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, 2015.

[19] H. He and J. Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, 2016.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] T. Hofmann. Probabilistic latent semantic indexing. In *ACM SIGIR Forum*, volume 51, pages 211–218. ACM, 2017.

[22] A. Hotho, A. Maedche, and S. Staab. Ontology-based text document clustering. *KI*, 16(4):48–54, 2002.

[23] B. Lemaire and G. Denhiere. Effects of high-order cooccurrences on word semantic similarity. *Current psychology letters. Behaviour, brain & cognition*, (18, Vol. 1, 2006), 2006.

[24] O. Levy, I. Dagan, G. Stanovsky, J. Eckle-Kohler, and I. Gurevych. Modeling extractive sentence intersection via subtree entailment. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2891–2901, 2016.

[25] Y. Li, Z. A. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on knowledge and data engineering*, 15(4):871–882, 2003.

[26] Y. H. Li and A. K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[27] S. Massung, C. Geigle, and C. Zhai. Meta: A unified toolkit for text retrieval and analysis. *Proceedings of ACL-2016 System Demonstrations*, pages 91–96, 2016.

[28] K. McKeown, S. Rosenthal, K. Thadani, and C. Moore. Time-efficient creation of an accurate sentence fusion corpus. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–320. Association for Computational Linguistics, 2010.

[29] R. Mihalcea, C. Corley, C. Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. 2013.

[31] J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792, 2016.

[32] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[33] C. Piech, J. Huang, Z. Chen, C. Do, A. Ng, and D. Koller. Tuned models of peer assessment in moocs. In *Proceedings of the 6th International Conference on Educational Data Mining (EDM 2013)*, 2013.

[34] L. R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. 1990.

[35] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[36] R. Sinha and R. Mihalcea. Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 363–369. IEEE, 2007.

[37] H. K. Suen. Peer assessment for massive open online courses (moocs). *The International Review of Research in Open and Distributed Learning*, 15(3), 2014.

[38] M. A. Sultan, S. Bethard, and T. Sumner. Dls @ cu: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 148–153, 2015.

[39] D. R. Swanson. Fish oil, raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18, 1986.

[40] K. Thadani and K. McKeown. Towards strict sentence intersection: decoding and evaluation strategies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 43–53. Association for Computational Linguistics, 2011.

[41] D. Wang, T. Li, S. Zhu, and C. Ding. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314. ACM, 2008.

[42] C.-P. Wei, C. C. Yang, and C.-M. Lin. A latent semantic indexing-based approach to multilingual document clustering. *Decision Support Systems*, 45(3):606–620, 2008.

[43] C. Zhai and S. Massung. *Text data management and analysis: a practical introduction to information retrieval and text mining*. Morgan & Claypool, 2016.