A Topological Approach to Compare Document Semantics Based on a New Variant of Syntactic N-grams

Fanchao Meng
Biocomplexity Institute
University of Virginia
mf3jh@virginia.edu

Abstract—This paper delivers a new perspective of thinking and utilizing syntactic n-grams (sn-grams). Sn-grams are a type of non-linear n-grams which have been playing a critical role in many NLP tasks. Introducing sn-grams to comparing document semantics thus is an appealing application, and few studies have reported progress at this. However, when proceeding on this application, we found three major issues of sn-grams: lack of significance, being sensitive to word orders and failing on capture indirect syntactic relations. To address these issues, we propose a new variant of sn-grams named generalized phrases (GPs). Then based on GPs we propose a topological approach, named DSCoH, to compute document semantic similarities. DSCoH has been extensively tested on the document semantics comparison and the document clustering tasks. The experimental results show that DSCoH can outperform state-of-the-art embedding-based methods.

I. INTRODUCTION

In this paper, we primarily propose a new variant of sn-grams (viz., Generalized Phrase; GP) and a document semantics comparison method (Document Similarity based on Cohomology; DSCoH) based on GPs and algebraic topology. Most of state-of-the-art sn-grams are defined on directed paths or subgraphs of dependency parse trees [37]. Such definitions are weak at reflecting the significance of a sn-gram in conveying semantics, detecting semantically similar sn-grams with different directions of relations, and capturing indirect syntactic relations. These issues motivate the invention of GPs. To verify the effectiveness of GP, we apply it in a fundamental NLP task, the document semantics comparison problem. Based on GPs we designed DSCoH, a document semantic similarity method, which utilizes algebraic topology techniques [12].

A GP is essentially an undirected and weighted sub-tree. Based on this concept, comparing semantics of two sentences can be studied over a graph built upon a constituency parse tree (CPT) pair and lexical similarity relations interconnecting the trees. Since graphs are considered as complexes in algebraic topology, then techniques such as cohomology can be utilized to address the problem. This directly motivates our design of DSCoH. We formulate the computation of document semantic similarities as a multi-objective optimization problem. With the help from cohomology, we found that basic cycles imply semantically similar 2-word GPs (i.e. those containing 1 or 2 words), and thus can be used to form approximate solutions. DSCoH implements this idea and computes document similarities. Additionally, we also discuss how the cases with K-word GPs can be solved by generalizations of the 2-word GP case. Since DSCoH is based on GPs which are actual constituents, DSCoH is completely explainable, which is superior to many other existing methods.

To verify their effectiveness of DSCoH, we conducted experiments on the document semantics comparison problem and the document clustering problem, and compared the performance of DSCoH and a set of embedding-based methods. The results are mostly positive for DSCoH especially on actual documents (i.e. those containing multiple sentences).

This paper is organized as follows: Section II reviews some previous work on sn-grams, and discusses how the GP is defined; Section III explains the analysis of document semantic similarities from the cohomology perspective, and elaborates DSCoH; and Section IV shows all experiments and results.

II. A VARIANT OF SYNTACTIC N-GRAMS

A. Previous Work & Issues

Sn-grams are defined to be a type of non-linear n-grams, and are typically constructed from dependency parse trees, each consisting of a path or a sub-tree [37], [38]. Sn-grams have been shown to be useful in various NLP tasks such as authorship attribution [33], [36], [38], machine translation [35], dependency parsing [29], contextual polarity analysis [1] and language modeling [41].

When trying to utilize current sn-grams in comparing document semantics, we found three major issues:

First, a syntactic n-gram needs to be considered its significance in representing semantics. Path lengths in parse trees provide an assessment of the strength of the relationship between words in a sentence. However, dependency parse trees have some limitations compared to constituency-based parse trees. For example, consider the sentence: “We eat pizza when we watch a movie.”, in which “eat” and “movie” do have an (indirect) syntactic relation yet much weaker than that between “watch” and “movie”. Figures [1] and [2] visualize these relations,
Fig. 1: “Eat” and “movie” have an indirect dependency relation, and the path length reflects that the relation is not immediate but is merely 1 greater than “watch” and “movie”.

Fig. 2: The path length between “Eat” and “movie” in the CPT more effectively reflects the weakness of their relation.

and the path lengths in CPT can reflect the significances of n-grams more effectively.

Second, sn-grams are too sensitive to word order. For example, consider the two sentences in Figure 3, in which “rider” and “bike” do not have significant difference in semantics from “bike” and “riding” but they are assigned with opposite directions in the dependency parse trees. On the other hand, constituency parse trees do not rely on directional syntactic relations.

Third, dependency parse tree based sn-grams can only capture immediate dependencies between words rather than indirect syntactic relations, while constituency parse trees are natural and straightforward on both direct and indirect syntactic relations.

B. Generalized Phrases

Based on the preceding discussion, we propose a variant of the sn-grams, named the generalized phrase (GP), and defined below. GPs captures the significance of sn-grams in reflecting semantics, are not sensitive to word order, and are able to capture indirect syntactic relations.

Definition: Generalized Phrase (GP)

A generalized phrase is a minimal non-empty subtree of the CPT containing at least one leaf. The leaves are considered orderless.

Note that the significance of the relatedness between the leaves is determined by the tree structure excluding the leaves. The simpler the structure, the more significant the GP. For 2-word GPs the significance is computed by using the path length between the two leaves, and for K-word GPs there can be multiple approaches to compute the significance, for example, considering the average of path lengths of all pairs of leaves.

III. DOCUMENT SEMANTICS COMPARISON

A. Motivations and Methods

We begin by explaining the motivations for DSCoH. First, we formulate the problem of computing the semantic similarity between two sentences as a multi-objective optimization problem; and second, we propose a framework based on cohomology theory producing approximate solutions to the optimization problem efficiently.

Semantic Similarity as an Optimization Problem:

Considering the GP concept introduced above, to compute the semantic similarity between two sentences, a general idea is to find as many as possible semantically similar GPs across the two sentences. Thus, maximizing the accumulation of similarities contributed by such semantically similar GPs is a core requirement to compute the sentence similarity, which one of our objectives. However, on the other hand, not all GPs are significant as discussed above. Thus, we also need to keep the GPs taken the accumulation as significant as possible, which is the other objective. Formally, let \( s_i \) and \( s_j \) denote two input sentences; let \( GP_i \) and \( GP_j \) denote the set of all GPs in consideration in the two sentences; let \( \text{Sim} : GP_i \times GP_j \to \mathbb{R}_{>0} \) denote a bounded and real-valued function computing the similarity between two GPs, the greater the output value, the more similar the GPs; let \( \text{Sig} : GP \to \mathbb{R}_{>0} \) denote a bounded and real-valued function computing the similarity between two GPs, the greater the output value, the more similar the GPs.

\( \text{Stopwords} \) should always be removed and should never appear in any GP. The parse trees will be pruned if necessary.

\( \text{The similarity between two documents can be straightforwardly computed by the sum of all sentence similarities.} \)
computing the significance weight of a GP, the greater the output value, the more significant the input GP; let $\text{Sig}_m$ denote the supremum of the set of possible significance values; and let $RMS$ denote the root mean square function. Note that $Sim$ may also need to take the significance weights of GPs into consideration. Then the optimization problem is formulated as follows:

$$\max_{A \subseteq \text{GP} \times \text{GP}} \sum_{(GP_x, GP_y) \in A} Sim(GP_x, GP_y)$$

$$\min_{A \subseteq \text{GP} \times \text{GP}} \sum_{(GP_x, GP_y) \in A} RMS \left( (\text{Sig}(GP_x) - \text{Sig}_m), (\text{Sig}(GP_y) - \text{Sig}_m) \right)$$

s.t. $GP_i \times GP_j < \infty$

The semantic similarity between $s_i$ and $s_j$ finally is computed by $\sum_{(GP_x, GP_y) \in A} Sim(GP_x, GP_y)$.

**Framework with Cohomology:**

This problem is NP-hard\(^4\) and the search space can be exponential (which is justified below). Next, we show how this problem can be naturally understood and solved from the cohomology theory\(^5\) perspective, and a framework solving the problem is proposed.

We start with a special yet typical case of the problem: seeking a set $A$ containing only 2-word GPs\(^6\) which produces an optimal solution. As to $Sim$ and $\text{Sig}$, the similarities between GPs primarily come from lexical similarities, and the significance weights of GPs come from syntactic relations between words. Thus, it is intuitive to have these information in a single object which by our design is an undirected and weighted graph. The construction of this graph is as follows. First, given two sentences, a CPT is computed for each of them. The weight on each edge in the parse trees is set to 1. We call the edges in parse trees syn-edges. Second, we compute the lexical similarity (a real value) between each word in a sentence and every word in the other. By applying a predetermined threshold to the similarities, an edge is created between each word pair, across the two sentences, which has a similarity greater than or equal to the threshold, and the weight on this edge is assigned the value of similarity. We call these edges sim-edges. Finally, we union the two parse trees and the set of edges created by lexical similarities, and produce the desired graph. We call this graph the syn-sim graph\(^5\) of the two sentences. An example of syn-sim graph is shown in Figure 4.

Given the syn-sim graph, the candidates for a solution are straightforward. Since trees are acyclic, then every pair of sim-edges induces a cycle in the graph which implies a pairing of two semantically similar GPs. Such GP pairs are candidate elements in a solution. However, in the worst case, there are $mn$ sim-edges, where $m$ and $n$ are the numbers of leaves in the two parse trees; $\binom{m+n}{2}$ GP pairs; and thus $O(2^{\binom{m+n}{2}})$ possible solutions in the search space. Therefore, finding “good” cycles efficiently for the objective functions is the core task in solving the optimization problem, which is the place where the cohomology theory can play a role as cycles in a graph are low dimensional “holes” in a topological space and the cohomology theory concentrates on finding and describing such “holes”\(^7\),\(^12\),\(^27\).

Cohomology is one of the primary perspectives in algebraic topology, and it is convenient in handling pure algebraic objects. By construction, syn-sim graphs are troublesome as a geometric object because of the coexistence of two different types of edges (i.e., syn-edges and sim-edges). Thus, in our analysis syn-sim graphs are considered as algebraic objects (as discussed below), and we utilize cohomology to study cycles in the syn-sim graphs.

To work with cohomology, an abstract simplicial complex\(^8\) (which is an algebraic object) is required. The construction is as follows. Given a syn-sim graph $G$, we contract each sim-edge to a super-vertex. For example, in Figure 4 the super-vertices can be $X_1 = (A, E)$, $X_2 = (B, F)$, $X_3 = (C, F)$ and $X_4 = (D, G)$. Then, the set of all such super-vertices forms an abstract simplicial complex, $S$, $G$ with sim-edges substituted by super-vertices can be considered as a geometric realization\(^27\) of $S$. For convenience, we define the group of $p$-cochains\(^27\) for the abstract simplicial complex to be $\text{Hom}(S, \mathbb{R}_{\geq 0})$ which is a functor\(^27\) containing all homomorphisms of $S$ into $\mathbb{R}_{\geq 0}$. Particularly, $Sim \in \text{Hom}(S, \mathbb{R}_{\geq 0})$ for 0-cochains. Additionally, $\mathbb{Z}_2$ is used as the ground field for coefficients of $p$-cochains. Based on these settings, we propose three lemmas.

\(^{\text{Supplemental Data}}\)

\(^{\text{Supplemental Data}}\)
which show that it is feasible to solve the optimization problem with approximation by utilizing cohomology.

**Lemma #1**: \( \sum_i c^0([X_i]) \) is a 0-cocycle but is not a 0-coboundary (i.e. \( \sum_i c^0([X_i]) \) is generator of the cohomology group \( H^0(S) \)) \(^{[27]}\), where \( c^0 \in Hom(S, \mathbb{R}_{\geq 0}) \) denotes elementary 0-cochain, and \( [X_i] \) denotes 0-simplex. This holds as \( G \) is connected \(^{[12]}\).

**Lemma #2**: For each elementary 0-cochain, its 1-coboundary is a cycle basis of \( G \). To prove this lemma, it is sufficient to show that, first, the elementary 1-cochains are cycles in \( G \); second, the elementary 1-cochains are independent of each other; and third, the 1-coboundary contains \( e - v + c \) (the circuit rank \(^{[2]}\)) elementary 1-cochains, where \( e \) denotes the number of edges in \( G \), \( v \) denotes the number of vertices, and \( c \) denotes the number of connected components. Given the construction of \( S \), the first condition automatically holds. The second condition holds because elementary cochains are defined to be independent \(^{[27]}\). To prove the third condition, we let \( e_{sim} \) denote the number of sim-edges; and let \( e_i \) denote the number of incident sim-edges of the vertices. Additionally, since \( G \) is always connected, then \( e = 1 \). Also, since for a tree the number of vertices is always one more than the number of edges, and \( G \) contains two parse trees, then \( e_i = v - 2 \). Thus, \( e - v + c = (e_i + e_c) - v + 1 = e_c - 1 \). Again, since \( G \) is always connected, then all \( X_i \)'s are reachable from each other. Thus, by the definition of the coboundary operator \(^{[27]}\), \( \delta([0^i([X_i])) = \sum_{i<j} c^1([X_i, X_j]) \), where \( \delta \) is the coboundary operator and \( i \sim j \) denotes adjacency, it always holds that \( \delta([0^i([X_i])) \) has \( e_c - 1 \) elementary 1-cochains. Figure 5 shows two examples of this lemma.

**Lemma #3**: The set of elementary 1-cochain contained in all \( \delta([0^i([X_i])) \) contains all basic cycles. The proof of this lemma can be done by way of contradiction. If a basic cycle was not contained, then it could be contained in \( \delta([0^i([X_i])) \) for any \( X_i \), which implies that this basic cycle had to be disconnected from all \( X_i \). This contradicts the connectivity of \( G \).

The three lemmas above imply that each \( \delta([0^i([X_i])) \) is an approximate solution to the optimization problem, and no candidate pair of GPs to the solutions is missed. Thus, finding the best solutions from basic cycles of \( G \) can be expected to have fairly good approximate solutions, for example, solving the minimum cycle basis problem on \( G \). Figure 6 summarizes the framework with cohomology. The time complexity of this framework is dominated by the one that is of the higher order between computing constituency parse trees and computing desired cycle basis. Theoretically, for the former task, methods based on the Cocke-Younger-Kasami algorithm can be higher than \( O(s^3|G|) \), where \( s \) is the sentence length and \( G \) is the CNF grammar \(^{[14]}\). For the latter task, the state-of-the-art method proposed in \(^{[25]}\) runs in \( O(b^2a/\log a + ba^2) \), where \( a \) is the number of vertices and \( b \) is the number of edges, and thus in our case it is \( O(a^3) \).

**Generalization to K-word GPs**:

It has been shown that a pairing of 2-word GPs is induced by two sim-edges in a syn-sim graph. Analogously, a pairing of \( K \)-word GPs is induced by \( K \) sim-edges. Thus, it is straightforward to induce from Lemma #2 that a pairing of \( K \)-word GPs always corresponds to \( (K - 1) \) basic cycles, and naturally such a pairing can be expressed as a formal sum of the \( (K - 1) \) basic cycles. This conclusion directly suggests an approach to compute semantic similarity between two sentences by utilizing \( K \)-word GPs. That is, first, we solve the optimization problem with 2-word GPs and obtain a cycle basis (e.g. a minimum cycle basis); second, we collect all pairings of \( K \)-word GPs, and for each pairing we decompose it into a subset of basic cycles; and finally, we sum the similarity results contributed by all \( K \)-word GP pairings and produce a

\(^{10}\)Similar to 2-word GPs, when considering \( K \)-word GPs, in some cases, we need to expand a single word into a chain of copies of the word with each edge in the chain being assigned a weight 1.
solution. The time complexity of this method is the running time of the framework in the 2-word GP case plus $O(\binom{K}{x})$, where $x$ is the number of sim-edges. However, if a minimum cycle basis is utilized, a subset of basic cycles may not form a minimum cycle basis for the subgraph induced by a $K$-word GP pairing. Thus, a better solution is to solve the optimization problem on this subgraph in the 2-word GP case, and finally sum the solutions to all sub-problems. The time complexity of this refined method is dominated by the higher order one between computing CPTs and $O(\binom{K}{x}K^3)$. It can be concluded from these generalization methods that a generalization may run much slower than the 2-word GP case and it may not gain much benefit in comparing semantics of sentences as the fundamental elements that contribute to the sentence similarities are still pairings of 2-word GPs (i.e. basic cycles). Therefore, we suggest to use the 2-word GP case to compute sentence similarities, and we leave the study of more advanced generalization methods to future work.

Next, we propose a concrete algorithm to compute the semantic similarity between two documents utilizing this framework in the 2-word GP case.

B. Algorithms

In this section, we propose an algorithm computing the semantic similarity between two documents. We name it DSCoH. DSCoH is an implementation of the framework with cohomology considering only 2-word GPs. This algorithm solves the minimum cycle basis problem to obtain pairings of semantically similar GPs. To simplify syn-sim graphs, we also propose a tree pruning algorithm and integrate it into DSCoH. DSCoH also contains a designed Sim function taking lexical similarities, GP significances and sentence lengths into consideration. We elaborate on DSCoH below.

Algorithm 1: DSCoH

**Given:**
- Two documents, $D_i$ and $D_j$.
- A threshold for lexical similarities, $\theta_w$.

**Seek:**
A real value as the semantic similarity between $D_i$ and $D_j$, denoted by $\phi_D(D_i, D_j)$.

**Stage 1: Constituency Parse Trees**
For each document $D_i$ and for each sentence $S_{ik} \in D_i$, where $k$ indexes the sentences, compute a CBPT for $S_{ik}$, denoted by $T_{ik}$.

**Stage 2: Pruned Parse Trees**
For each $T_{ik}$, apply Algorithm 2 to prune $T_{ik}$. The pruned tree is denoted by $\hat{T}_{ik}$.

**Stage 3: Lexical Similarity Relations**
For each pruned tree pair $(\hat{T}_{ik}, \hat{T}_{jh})$, where $\hat{T}_{ik}$ and $\hat{T}_{jh}$ are in $D_i$ and $D_j$ respectively, identify all word pairs $\{(t^a_{ik}, t^b_{jh})\}$, where $t^a_{ik} \in \hat{T}_{ik}$ and $t^b_{jh} \in \hat{T}_{jh}$, with lexical similarities $\phi_w(t^a_{ik}, t^b_{jh}) \geq \theta_w$.

**Stage 4: Syn-Sim Graphs**
For each pair of pruned trees $\hat{T}_{ik}$ and $\hat{T}_{jh}$, union the trees and create an edge for each identified word pair obtained from Stage 3 to construct a syn-sim graph, denoted by $G_{ik,jh}$.

**Stage 5: Minimum Cycle Basis**
For each syn-sim graph $G_{ik,jh}$ over $k$ and $h$, set the weight on each sim-edge to $\infty$ (only for this stage) and compute a minimum cycle basis, $\mathcal{B}_{kh}$.

**Stage 6: GP Significance Weights**
For each basic cycle $c_l = \{GP_{1l}, GP_{2l}\}$, compute a weight for $c_l$ based on the path lengths of $GP_{1l}$ and $GP_{2l}$, called the GP significance weight, denoted by $w_{c_l}$:

$$w_{c_l} = \frac{\theta_{c_1}}{p_{c_1}} + \frac{\theta_{c_2}}{p_{c_2}}$$

where $\theta_{c_1}$ and $\theta_{c_2}$ are predetermined parameters which empirically are set to 3, $p_{c_1}$ and $p_{c_2}$ are path lengths of $GP_{1l}$ and $GP_{2l}$.

**Stage 7: Sentence Length Weights**
For each pruned tree pair $(\hat{T}_{ik}, \hat{T}_{jh})$, compute a weight, $w_{skh}$, reflecting how much the basic cycles have covered the semantics of the two sentences. The weight is called the sentence length weight, and computed by

$$w_{skh} = \frac{2|\mathcal{B}_{kh}|}{L_{ik} + B_{kh} + L_{jh}}$$

where $L_{ik}$ and $L_{jh}$ denote the numbers of leaves in $\hat{T}_{ik}$ and $\hat{T}_{jh}$ respectively.

**Stage 8: Document Semantic Similarity**
The similarity between two GPs in a basic cycle is computed by

$$\phi_c(c_l) = w_{c_l} \cdot \min \left(\phi_w(t^a_{ik}, t^b_{jh}), \phi_w(t^a_{jk}, t^b_{ik})\right)$$

The similarity between $D_i$ and $D_j$ thus is computed by

$$\phi_D(D_i, D_j) = \sum_{k,h} \left( w_{skh} \cdot \sum_{c_l \in \mathcal{B}_{kh}} \phi_c(c_l) \right)$$

Algorithm 2: Parse Tree Pruning

**Given:**
- A CPT $T$.
- A stopword list, $W_s$.
- A set of name entities in consideration, $N$.
- A set of POS tags in consideration, $P$.

**Seek:**
A pruned tree.

IF $T$ is a single-node tree, and the only node is denoted by $t_0$:

11 DSCoH is short for “document similarity based on cohomology”.
The final pruned tree is in the simplest form.

Tasks, the document semantics comparison and the document out constituents not in consideration, and can be customized.

Implementations.

To offset this bias.

than the other pair. The sentence length weights are introduced

S

S

S

GP pairings to

example, suppose that sentences

GPs are both significant.

of GP and in our case

similarity. This weight should be induced by the function

sim-edges, which keeps every basic cycle corresponding to a

basic cycle can only contain 2

Note that in general minimum basis cycles in a graph may not be unique.

As to Algorithm 1, a number of comments are added here:

First, to compute lexical similarities there are multiple approaches, for example, approaches based on word sense disambiguation such as ADW [32] and approaches using word embedding models such as LexVec [34].

Second, in Stage 5, the weights on sim-edges are set to \( \infty \) because doing this will force a basic cycle can only contain 2 sim-edges, which keeps every basic cycle corresponding to a pair of semantically similar GPs. Also, to compute a minimum cycle basis, we utilize the method described in [17].

Third, a significance weight measures the significance of the similarity between two paired GPs in reflecting the document similarity. This weight should be induced by the function \( \text{Sig} \), and in our case \( \text{Sig}(GP_x) \) is determined by the path length of \( GP_x \). A significance weight is high only when the paired GPs are both significant.

Fourth, since DSCoH concentrates on “what are similar” rather than “what are not similar”, then it happens that, for example, suppose that sentences \( S_1 \) and \( S_2 \) have the same GP pairings to \( S_1 \) and \( S_3 \), even though \( S_3 \) may be much longer than \( S_2 \), then in this case DSCoH will anyway give the same similarity between \( S_1 \) and \( S_2 \) as that between \( S_1 \) and \( S_3 \), though in fact \( S_1 \) and \( S_3 \) are very likely to be less similar than the other pair. The sentence length weights are introduced to offset this bias.

Fifth, DSCoH follows a modular design. Several components such as \( w_{v_h}, w_{v_{sbh}} \) and \( \varphi_c \) can be substituted by other implementations.

As to Algorithm 2, \( \mathcal{W}_s, \mathbb{N} \) and \( \mathbb{P} \) all act as filters to rule out constituents not in consideration, and can be customized. The final pruned tree is in the simplest form.

IV. Experiments

To verify the effectiveness of DSCoH, we test it into two tasks, the document semantics comparison and the document clustering. In the document semantics comparison task, DSCoH performs as well as, sometimes better than, other methods. In the document clustering task, DSCoH outperforms almost all other methods. We elaborate on each of the tasks below.

A. Document Semantics Comparison

This task requires each method to take every pair of documents in a dataset and produce a value as the similarity. These similarity values are compared to a set of values determined by human judges. The Spearman correlation for each method is then computed to measure how close to the human judges this method performs.

Datasets:

The datasets in use include: Lee60 [21] which contains 60 document pairs, Li30 [23] which contains 30 sentence pairs, STS2017 [6] which contains 250 sentence pairs, and SICK [24] which contains 9840 sentence pairs.

Methods to Compare:

A set of pretrained embedding model based methods are used for comparison including Word2Vec [20] equipped with WMD [18], NASARI [5], Doc2Vec [20], GloVe [31], fastText [3], LexVec [34] and Sent2Vec [30]. Each document is represented as a vector by utilizing the embedding models, and then we use Cosine similarity to compute a similarity value for every pair of documents. Note that Doc2Vec and Sent2Vec can directly take a document as input and produce a vector. For others, we use the conventional average word vector method to represent documents.

Our method is DSCoH utilizing NASARI for lexical similarities. Note that DSCoH can interface with any lexical similarity method, and NASARI is randomly selected without any particular purpose. Additionally, since the lexical similarity threshold is a parameter for DSCoH, we test 10 settings and show the results.

Experimental Results:

The results are shown in Table I

Discussion:

Two important observations are discussed here. One is that DSCoH is relatively stable as the similarity threshold varies, which makes DSCoH friendly in real practice. The other is that DSCoH performs better on actual documents (i.e. those containing several sentences) than sentences. It obtains a good performance on Li30 because Li30 was created by using words’ interpretation sentences in a dictionary. Those sentences are mostly short and contain few expressive words. Consequently the basic cycles, in a comparison, captured by DSCoH would have carried almost the whole semantics if the two sentences are similar. In STS2017 and SICK, sentences are typically longer than those in Li30. Thus, DSCoH may not capture as adequate semantics in two sentences as embedding based methods. On the other hand, for actual documents, DSCoH would be able to capture adequate key semantics while embedding based

\[ \text{IF} \ t_0 \in \mathcal{W}_s \ \text{OR} \ t_0 \notin \mathbb{N} \ \text{OR} \ t_0 \text{ is a punctuation:} \]

\[ \text{RETURN} \ An \ empty \ tree. \]

\[ \text{ELSE:} \]

\[ \text{RETURN} \ T. \]

\[ \text{ELSE:} \]

\[ \text{IF} \ \text{the root} \ r \ of \ T, \ such \ that \ r \notin \mathbb{P}: \]

\[ \text{RETURN} \ An \ empty \ tree. \]

\[ \text{ELSE IF} \ r \ has \ only \ one \ child \ c_0: \]

\[ \text{Apply \ Algorithm \ 2 \ to \ the \ subtree \ rooted \ at \ c_0.} \]

\[ \text{RETURN} \ The \ pruned \ subtree \ rooted \ at \ c_0. \]

\[ \text{ELSE:} \]

\[ \text{Prune \ all \ subtrees \ rooted \ at} \ r \ \text{by \ applying} \]

\[ \text{Algorithm \ 2 \ recursively.} \]

\[ \text{Remove \ from} \ r \ \text{the} \ \text{subtrees \ which \ have \ empty} \]

\[ \text{pruned \ trees.} \]

\[ \text{RETURN} \ \text{The \ pruned \ tree}, \ T'. \]

\[ \text{IF} \ t_0 \in \mathcal{W}_s \ \text{OR} \ t_0 \notin \mathbb{N} \ \text{OR} \ t_0 \text{ is a punctuation:} \]

\[ \text{RETURN} \ An \ empty \ tree. \]

\[ \text{ELSE:} \]

\[ \text{RETURN} \ T. \]

\[ \text{ELSE:} \]

\[ \text{IF} \ \text{the root} \ r \ of \ T, \ such \ that \ r \notin \mathbb{P}: \]

\[ \text{RETURN} \ An \ empty \ tree. \]

\[ \text{ELSE IF} \ r \ has \ only \ one \ child \ c_0: \]

\[ \text{Apply \ Algorithm \ 2 \ to \ the \ subtree \ rooted \ at \ c_0.} \]

\[ \text{RETURN} \ The \ pruned \ subtree \ rooted \ at \ c_0. \]

\[ \text{ELSE:} \]

\[ \text{Prune \ all \ subtrees \ rooted \ at} \ r \ \text{by \ applying} \]

\[ \text{Algorithm \ 2 \ recursively.} \]

\[ \text{Remove \ from} \ r \ \text{the} \ \text{subtrees \ which \ have \ empty} \]

\[ \text{pruned \ trees.} \]

\[ \text{RETURN} \ \text{The \ pruned \ tree}, \ T'. \]
methods typically would have taken “too much” semantics into consideration such that some unimportant semantics become noise. We justify this claim in Section IV-B.

B. Document Clustering

This task requires each method to group a set of documents by their semantics. The grouping results then are compared to the human judgments by computing Adjusted Rand Index (ARI) [15], Normalized Mutual Information (NMI) [40] and Fowlkes-Mallows Index (FMI) [8]. All of these scores range in [0, 1]. They measures how well a clustering produced by a method matches the human judgments, the higher the better.

Datasets

Three datasets are in use including 20Newsgroups [19], Reuters-21578 [22] and BBC [11]. Based on 20Newsgroups, two sampled datasets are created: 20News-M5 containing 5 categories without confusion and 20News-C10 containing 10 categories with confusion. Based on Reuters-21578, a sampled set is created: Reuters-M7 containing 7 categories. Based on BBC, a sampled set is created: BBC-M5. In each category in the four sampled sets, 50 documents are sampled.

Methods to Compare:

The methods for comparison are the same as those in Section IV-A except WMD, and the clustering method is spectral clustering [28] [10].

Experimental Results:

The results are shown in Table I.

Discussion:

The results show that DSCoH outperforms almost all methods except Sent2Vec, though they are very close. These results strongly support the claim proposed in Section IV-A.

Thus, it is optimistic to conclude that DSCoH would perform well in document-based tasks.

V. CONCLUSION & FUTURE WORK

Several takeaways are: first, GPs are effective in representing document semantics; second, DSCoH is effective in comparing document semantics; third, DSCoH is completely explainable and fourth, algebraic topology techniques are not adornments but insightful tools in comparing document semantics.

On the other hand, the work in this paper still has a huge room to grow. First, could we design a single document semantic representation? Second, are K-GPs unnecessary or we have not understood them well? Third, computing constituency parse trees is always a pain as to running time. Could we use dependency parse trees instead? And fourth, could we make DSCoH more effective to sentences?

REFERENCES

[1] Apoorv Agarwal, Fadi Biadsy, and Kathleen R Mckeown. Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, pages 24–32. Association for Computational Linguistics, 2009.
[2] Claude Berge. The Theory of Graphs. Courier Corporation, 2001.
[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5:135–146, 2017.
[4] Chris Buckley and Gerard Salton. Onix Text Retrieval Toolkit Stopword List 2, 2007.
[5] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. NASARI: A Novel Approach to a Semantically-Aware Representation of Items. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 567–577, 2015.
[6] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint arXiv:1706.00055, 2017.
[7] Herbert Edelsbrunner and John Harer. Computational Topology: An Introduction. American Mathematical Soc., 2010.
[8] Edward B Fowlkes and Colin L Mallows. A Method for Comparing Two Hierarchical Clusterings. Journal of the American statistical association, 78(383):553–569, 1983.
[9] Robert W Ghrist. Elementary applied topology, volume 1. Createpace Seattle, 2014.
