Semantic Feature Structure Extraction from Documents Based on Extended Lexical Chains

Terry Ruas
University of Michigan - Dearborn
Dearborn, MI, USA
truas@umich.edu

William Grosky
University of Michigan - Dearborn
Dearborn, MI, USA
wgrosky@umich.edu

Abstract

The meaning of a sentence in a document is more easily determined if its constituent words exhibit cohesion with respect to their individual semantics. This paper explores the degree of cohesion among a document's words using lexical chains as a semantic representation of its meaning. Using a combination of diverse types of lexical chains, we develop a text document representation that can be used for semantic document retrieval. For our approach, we develop two kinds of lexical chains: (i) a multilevel flexible chain representation of the extracted semantic values, which is used to construct a fixed segmentation of these chains and constituent words in the text; and (ii) a fixed lexical chain obtained directly from the initial semantic representation from a document. The extraction and processing of concepts is performed using WordNet as a lexical database. The segmentation then uses these lexical chains to model the dispersion of concepts in the document. Representing each document as a high-dimensional vector, we use spherical k-means clustering to demonstrate that our approach performs better than previous techniques.

1 Introduction

Since the late 1980's, when there was a burst of research in dimensional reduction techniques (Dumais et al., 1988), information retrieval has been concerned with semantics. Since multimedia entities, including text, have multiple meanings, examining the context in which they appear became of significant importance to their overall disambiguation.

An important example of this in the natural language processing community was the formulation of lexical chains (Morris and Hirst, 1991). A lexical chain is a contiguous portion of text which has semantic cohesion. Such chains are of variable length and have been used throughout the intervening years in many ways; e.g., for semantic characterization of the underlying document, for question and answers tasks, for document summarization, and for clustering documents into semantically uniform groups.

In this paper, we propose two new types of lexical chains based on semantic representation: the first is called Flexible-to-Fixed Lexical Chains (Flex2Fix) and the second, Fixed Lexical Chains (FixLC). In the first, these representations follow and extend the model proposed by (Ruas and Grosky, 2017), transforming their flexible lexical chains into fixed structures, which are later transformed into vectors of semantic values. In the second, we build fixed lexical structures directly from their initial semantic value.

First, we start by identifying the most suitable semantic representation for each word, considering their context. Second, we use these semantic abstractions and find the flexible lexical chains in a document. This approach extracts and builds cohesive sequences of ideas with respect to the semantic value shared among words in a dynamic way. Third, we develop an approach to transform flexible lexical chains into fixed lexical chains. All these chains are used to construct a vector representation corresponding to a document's semantic structure. This is done to represent the document's semantic value at a higher level of abstraction. We also investigate how fixed lexical chains obtained directly from the document's semantic representation perform against traditional approaches (e.g. Bag-of-Words (BOW)) and the derived fixed structures from the flexible ones.

The remainder of this paper appears as follows. Section 2 reviews existing work in lexical chains and provides additional information on our technique. In Section 3, we present our methodology and proposed algorithms for content-based retrieval using lexical chains. Section

4 concerns the experimental validation of our approach, while in Section 5, we offer some final considerations and potential future work.

2 Related Work

The term *lexical chains* was first proposed by (Morris and Hirst, 1991) as an extension of *lexical cohesion*, a concept introduced by (Halliday and Hasan, 1976). A text in which many of its words are semantically connected often produces a certain degree of continuity in its ideas, providing good cohesion among its words. Lexical cohesion is more likely to occur between words close to each other in a text, especially those contiguously ordered. The sequence of related words, tied by a common semantic affinity is classified as a lexical chain (Morris and Hirst, 1991).

The use of lexical chains in document processing and analysis (e.g. text similarity, word disambiguation, document clustering) has been widely studied in the literature. In (Barzilay and Elhadad, 1997; Silber and McCoy, 2000), lexical chains are used to summarize texts. The former extracts and classifies lexical chains and discovers significant sentences to represent documents from them. The latter proposes a linear-time algorithm for constructing the lexical chains that will capture the meaning of a text. Some authors use WordNet (WN) to improve the search and evaluation of lexical chains. (Budanitsky and Hirst, 2001; Budanitsky and Hirst, 2006) compare several measurements of semantic distance and relatedness using lexical chains in conjunction with WN. (Moldovan and Novischi, 2002) studies the use of lexical chains for finding topically related words. This is done considering the glosses for each synset in WN. (Hotho et al., 2003) explores the benefits of using WN to improve document clustering based on an explicit matching between terms found in the text and the lexical database. (McCarthy et al., 2004) presents a methodology to categorize and find the most predominant synsets in untagged texts using WN. In (Sedding and Kazakov, 2001), WN is used for document clustering, exploring the benefits of incorporating hypernyms and synonyms into their approach. In (Pedersen et al., 2004), an application developed in Perl is proposed to calculate the relatedness of concepts via WN through different measures of similarity. (Guo and Diab, 2011) hypothesizes that if the semantics of words are known in advance, it is possible to get a better statistical inference concerning a document's overall idea.

In more recent works, (Navigli, 2009) presents an extensive study in the Word Sense Disambiguation (WSD) arena, in which he proposes an unsupervised WSD algorithm based on generating Spreading Activation Networks (SANs) from senses of a thesaurus and the relations between them. (Meng et al., 2013) explores the theory behind state-of-the-art techniques for semantic similarity measures in four main categories: path length-based, information content-based, feature-based, and hybrid measures. (AlAgha and Nafee, 2014) proposes an approach to improve document clustering by exploring the semantic knowledge offered by Wikipedia. The authors discuss this hypothesis, comparing the results using WN and Wikipedia, claiming that the latter is more robust. In (Pradhan et al., 2015) several measures of similarity (e.g. normalized Google distance, normalized compression distance, cosine distance, latent semantic similarity) are applied to categorize sentences, words, paragraphs and documents according to their lexical and semantic similarities. In (Bär et al., 2015) an extensive study about available text similarity measures is done as part of semantic evaluation, and for the detection of text reusability. They argue that text similarity cannot be considered as a static and absolute notion. Instead, one should carefully define in which levels and perspectives two documents are similar or not. (Wei et al., 2015) combines lexical chains and WN to extract a set of semantically related words from texts and then uses them for clustering. Their approach uses an ontological hierarchical structure and relations to provide a more accurate assessment of the similarity between terms for WSD. In (Tekli, 2016), they conduct a comprehensive review of the methods related to XML-based semi-structured semantic analysis and disambiguation. Although focused in the XML arena, this work provides an overview of the semantic disambiguation field, as well. They cover traditional WSD methods and potential application scenarios that could benefit from them (e.g. data clustering, semantic-aware indexing) while discussing current on-going challenges in the area.

Although extensively studied, the concept of lexical chains still has much to be explored. Besides the fact that each idiom has its own identity, most of the presented work either relies solely on statistical approaches (e.g. *tf-idf*, BOW) or focuses on one aspect of word relatedness. Some research groups focus their efforts on exploring

algorithms and tools to calculate distances between several entities, such as words, paragraphs, synonyms and lexical chains. A few rely on annotated text and/or machine learning techniques to extract semantic-like features from documents. Others expand the set for each word, considering their immediate synonyms or hypernyms to improve corpus or query. The ones inspecting lexical chains, build them using the words individually, or often using some common/direct synonym. Although these are interesting approaches, they are only focused on the word itself, leading to an alternative BOW representation. They still do not explicitly consider the context of a given word in relation to its location or surroundings in the text. Semantic and contextual aspects are difficult to track, but are important aspects of effective human communication. In the last eleven years, the interest in these topics and their contributions to traditional approaches have been increasing among distinct scientific communities. For example, (Grosky and Ruas, 2017) examined the research conducted in the multimedia arena, consisting of 2,872 items (e.g. papers, journals, reports) in the last 11 years, and found an increasing number of publications exploring *semantics* and *contextual* aspects in different areas, pointing to a trend in these areas.

Our approach contributes to this topic by expanding the notion of WSD, considering all synsets of a given word, including the influence among them. Furthermore, our chains are produced by using the most suitable synset for a word, which is a result of the evaluation of its contiguous neighbors (context). In addition, our lexical chains consider all desired hypernyms in WN, given a certain threshold, which can be adjusted to obtain higher (more general) or lower (less general) semantic representations.

3 Building Extended Lexical Chains

As stated by (Morris and Hirst, 1991) (Budanitsky and Hirst, 2006), there are multiple categories in which lexical chains can be classified. These concepts are explored in our approach through WN by using synsets and hypernyms. WN is a lexical database that provides a complex structure of how words and their meanings are related. The following is a small summary of the main terms necessary to understand our work using extensible lexical chains and WN (Fellbaum, 1998):

- *Synonym* – a one-to-many mapping from concepts to words;
- *Synset* – a set of cognitive synonyms (one or more) of a given word that share a common concept;
- *Synset ID* – an ID that represents the entire synset;
- *Sense* – the elements in each synset;
- *Hypernym* - a general abstraction of synset, corresponding to *a-kind-of* relation;
- *Lowest Common Subsumer* – is the most specific synset in the hypernym hierarchy which is an ancestor of the given synsets;
- *Root* – initial synset in WN, called *entity*.

A *synset* is a set of synonyms (one or more) for a given word, while *hypernyms* are sets of more general synsets. For example: pug and bulldog are each a kind of dog. A mammal is a generalization of dog, and so on.

Our model consists of exploring documents through their lexical structure. This will be provided by evaluating the semantic value of each word in a text (Ruas and Grosky, 2017). The main idea can be divided into four major tasks: (i) Document Extraction Process, (ii) Best Synset Disambiguation Module, (iii) Lexical Synset Chain Extraction Module and (iv) Distributed Semantic Mapping.

In (i), we select the documents to be processed and clean the data, eliminating noise, such as stopwords, special characters, punctuation, and html tags, among others. In this paper, the source of data was a set of webpages from Wikipedia, so an enhanced stopwords' list had to be used. Once the documents are preprocessed, we filter only those words that have a synset match in WN. If a word in the text has no match in WN it will not contribute to the formation of lexical chains, so we assume that it can be discarded.

3.1 Best Synset Disambiguation Module

In (ii), the Best Synset Disambiguation Module is a subroutine that applies and extends the concept of WSD, but considers the synsets extracted from w_i, w_{i-1} and w_{i+1}. WSD is the problem in which one must decide which synset is better suited for a word in a sentence, given that this word has multiple meanings and each one of these meanings may be affected by other nearby words. Most works in the lexical chains arena try to build these structures by considering only the words within the document, while some use an auxiliary annotated corpus for learning. Others have used the most common synset for each

word (first synset provided by WN for each word) as well as keeping track of word pair occurrences and their distribution in a document. Our approach considers the influence of immediate neighbors for each word w_i, evaluated using all synsets available in WN, for the word itself as well as for its hypernyms. For each word w_i, with $i=1,2,\ldots,n$, there are 0 or more synsets available in WN. In our experiments, only the nouns existing in WN are considered, so nouns not present in WN are discarded. The current version of WN used in this paper (3.1) has approximately 117,000 synsets, divided into four major categories: 81,000 noun synsets, 13,600 verb synsets, 19,000 adjective synsets, and 3,600 adverb synsets. Since the number of nouns comprise almost 70% of all information available, we choose to work with this category of synsets (Fellbaum, 2010). In addition, nouns allow us to use interesting relationships between synsets, such as hypernyms.

We represent the best synset ID (BSID) of a word w_i by analyzing the effects of its predecessor (w_{i-1}) and successor (w_{i+1}), called *FormerSynsetID*(w_i) (FSID(w_i)) and *LatterSynsetID*(w_i) (LSID(w_i)), respectively. FSID(w_i) and LSID(w_i) are selected based on the score obtained by all possible combinations between all synsets of the pairs (w_i,w_{i-1}) and (w_i,w_{i+1}). The synsets for w_i with the highest score value in comparison with w_{i-1} and w_{i+1} will be represented by FSID(w_i) and LSID(w_i) respectively. We use Jiang & Conrath's algorithm, which is an information content-based measure used to calculate the similarity between two synsets. This value is obtained by calculating the distance of two synsets (c_1, c_2), as shown in Equation 1 (Jiang and Conrath, 1997; Meng et al., 2013),

$$dis_{jiang}(c_1, c_2) = \big(IC(c_1) + IC(c_2)\big) - 2IC(lcs(c_1, c_2)) \quad (1)$$

where c_1 and c_2 represent the synsets for word 1 and word 2; $IC(c_k)$ is the information content calculated for c_k and $lcs(c_1,c_2)$ is the lowest common subsumer (hypernym) of synset c_1 and synset c_2. In our implementation, the information content is provided by the *ic-semcor.dat* [1] file, which is based on the *cntlist* file distributed with WN 3.0. The semantic similarity score is calculated for all synsets available for each word evaluated. Finally, every word will hold two prospective synsets (FSID(w_i) and LSID(w_i)), which represent the synsets with the highest score (except the first and last word of the document). These will be used to produce the BSID for (w_i). There are other measures (e.g. Lin (Lin, 1998), Hirst (Hirst and St-Onge, 1998), Resnik (Resnik, 1995), Wu & Palmer (Wu and Palmer, 1994)), besides Jiang & Conrath, that can be used to calculate the relatedness of two synsets. They are divided into four main categories: path based, IC based, feature based, and hybrid methods (Meng et al., 2013).

Jiang & Conrath's algorithm was chosen because of its execution time and robustness, since it considers the IC of the synsets. In addition, according to (Budanitsky and Hirst, 2001) Jiang & Conrath's measure outperformed other known techniques used for semantic similarity. Further experiments using different IC files (e.g. BNC, Treebank, Brown, Shaks) in comparison with path, feature, and hybrid approaches are still necessary to improve our current findings. More details about Jiang & Conrath and the others algorithms can be found in (Jiang and Conrath, 1997; Meng et al., 2013). The latter provides a small survey about the most popular WSD algorithms available as well.

After the FSID and LSID for each word w_i has been found, it is necessary to find the BSID for the given word w_i. For this task, we use the *Best Synset Disambiguation Algorithm* (BSD) (Ruas and Grosky, 2017), which will identify what is the BSID, using as input parameters LSID and FSID. Three cases are considered prior to its selection: (a) if FSID(w_i) and LSID(w_i) are equal, then BSID(w_i) = FSID(w_i) = LSID(w_i); (b) the lowest common subsumer of FSID(w_i) and LSID(w_i), given a depth threshold; and (c), if (b) produces an empty set, the deepest synset among FSID(w_i) and LSID(w_i) is chosen. In case both have the same depth, one is chosen randomly. In (b), we used the depth of 6 (root being the initial point) as the limit to look for common hypernym extraction. This value was obtained by experimental tests considering factors like: execution time, diversity of synsets, diversity of chains, specificity of synsets, and others. This algorithm mitigates the fact that words with multiple meanings (polysemy) might have an unstable representation, by performing a two-level disambiguation process. In the first level, we apply known WSD techniques to obtain prospective pairs of synsets with the highest score, considering the context of each word. The second level extends the concept of WSD to synsets (BSD). More details about this algorithm are explained in (Ruas and Grosky, 2017).

[1] http://wn-similarity.sourceforge.net/

The identification of the BSID for each term w_i considers its surroundings, so the most suitable semantic representation can be used to construct our lexical chains. In (Ruas and Grosky, 2017), BSID and flexible lexical chains have been used to suggest keywords that represent the main concepts embedded in a document.

As we traverse the graph in WN for the lowest common subsumer (hypernyms) extraction (b), we consider the first hypernym on each level, for each synset. Since WN organizes its synsets from most to least frequent usage, and we are generalizing the concepts as we move towards the root, it is only natural that we extract a hypernym that will provide the most diffused element with respect to its frequency in the lexical database. In other words, the first hypernym in every upper level will provide greater probability of an intersection with another synset when we build our lexical chains.

3.2 Lexical Chain Extraction Module

Once all words have their BSID selected, we start building our lexical chains in a two-phase subroutine called Lexical Synset Chain Extraction Module. To the best of our knowledge, this module (iii) is introducing two novel contributions. First, the extension of flexible chains into fixed structures to better represent the semantic values extracted from these synsets, and second, we construct parametrized fixed lexical chains, considering the BSID representation obtained in Section 3.1.

We use the *Flexible Lexical Chains Algorithm* (FlexLC) (Ruas and Grosky, 2017), which extracts lexical chains, evaluating if a new word, represented by the BSID(w_i), or its hypernyms, present lexical cohesion among themselves and the current chain under construction. If the evaluated synset has semantic affinity with the chain being constructed, then this new synset is incorporated to the chain. Otherwise, a new chain must be initialized so that the next semantic representation can be captured.

The idea behind the algorithm presented in (Ruas and Grosky, 2017) is quite simple. As long as synsets have a common meaning (even a more general one), they will be part of the same set (chain), otherwise a new set must be created. To illustrate the FlexLC algorithm, consider the sentence "*the dog and the cat run with the child and her mom in the park, this Summer*". After cleaning the data and applying the BSD algorithm, we only keep the BSIDs for the words that have a match in WN, producing the following list {*dog, cat, child, mom, park, summer*}. The chain starts with BSID(*dog*) and evaluates BSID(*cat*), both of which have the hypernym "*carnivore*" in common, so BSID("*cat*") is added to the chain and BSID(*carnivore*) is set as the ID for the current chain under construction. Next, BSID(*carnivore*) is evaluated with BSID(*child*), which has the hypernym "*organism*" in common. BSID(*child*) is then added to the current chain and BSID(*organism*) is set as its new ID. Next, the other BSIDs are processed following the same idea. Since the hypernym *organism* (ID for the chain under construction) is also shared by BSID(*mom*), the latter BSID is also added to the chain. However, BSID(*park*) and BSID(*summer*) do not share any common synset with the current chain, or themselves, other than WN's root (entity). In that case, they will have their own chain, resulting in the following structure {{*dog*, *cat*, *child*, *mom*}, {*park*}, {*summer*}}, where *organism*, *park* and *summer* represent, respectively, each flexible chain. More details about FlexLC algorithm is available in (Ruas and Grosky, 2017).

After all FlexLC are produced, we convert these flexible chains into fixed structures (Flex2Fix) to reduce the high dimensionality caused by the number of single-synset-chains produced in the previous step. We also want to mitigate the problem of two or more long flexible chains being separated by one single-synset-chain occurrence.

Each flexible chain in this step will have an ID (FlexLCID) that will be assigned to all component words (w_i) of this chain. For example, consider the flexible chain {{*dog*, *cat*, *puppy*}, {*park*}, {*summer*}, {*dog*, *cat*, *puppy*}} represented by the IDs {{*animal*}, {*park*}, {*summer*}, {*animal*}}. These IDs are propagated to the BSIDs of the original chain, resulting in a new one with the following structure {{*animal*, *animal*, *animal*}, {*park*}, {*summer*}, {{*animal*, *animal*, *animal*}}, which will be processed into fixed structures. In this project, we divide the FlexLCIDs in sets of 4 units, so considering our example, the new chains would have the following construction {{*animal, animal, animal, park*}, {*summer, animal, animal, animal*}}. Both, the first and the second chain, have the synset *animal* as the dominant one, causing the ID for these fixed chains to be recalibrated to {{*animal*}, {*animal*}}. In our experiments, using the chunk size of 4 provided the most diverse set of chains. Since the chains are originated from the FlexLC in this algorithm, there will not be a

common synset shared between different chains within our threshold, so we do not need to traverse WN for hypernyms again. Therefore, to track the dominant synset in each fixed chunk is enough. Figure 1 shows in detail the Flex-to-Fixed algorithm (Flex2Fix), while Figure 2 is a pictorial representation of the process itself.

```
for each word occurrence wi in D, where i={1,…,n}:
   set synset(wi) = p, where wi occurs in FLC(k)and p = FLCID(k)
split D into fixed-sized chunks of k words each
for each chunk cwj, where j={1,…,NChunks}:
   let wj,1,…,wj,k be the word occurrences in chunk cwj
   let θj = <synset(wj,1),synset(wj,2),…,synset(wj,k)>
   represent chunk cwj by the dominant synset of θj, dominant(cwj)
      if θj has no dominant synset, choose one randomly
return dominant(cwj) for j=1,…,NChunks
```

Figure 1. Flex2Fix Algorithm.

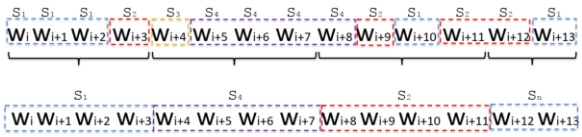

Figure 2. Flex2Fix Process.

In this paper, we also propose a variation to build fixed size chains called *Fixed Lexical Chains Algorithm* (FixLC), which is derived directly from the BSIDs found in Section 3.1. Differing from the previous algorithm (Flex2Fix), this does not use any pre-processed lexical chains, as its construction is entirely based on the BSIDs for each word. We develop this technique to compare which lexical chain structure would present better results, the one derived from FlexLC or obtained directly from BSIDs (FixLC). The latter "forces" a fixed dimensionality in the size of each chain from BSID's, so we will need to consider the hypernyms in each fixed chunk.

The main idea behind the FixLC algorithm is to divide the BSIDs, for every document, in chunks of size n (c_n), and evaluate what is the synset that best represents each one of them these chunks. As in the previous approach (Flex2Fix), the size of 4 synsets was chosen, so both techniques could be better compared. For each chain c_n, we extract all hypernyms (including the initial synsets) from all the BSID in each chunk and select the dominant synset to represent the entire chain. If there is no dominant BSID, we select the deepest one in the chain. In case there are more than one, the choice for its representative synset is done randomly, since all of them could represent the given chain.

It is important to mention that hypernyms beyond a certain threshold are not considered in our approach. The closer to the root we get, the more common our synsets become, contributing poorly to the semantic diversity of a chain. Therefore, hypernyms with depth below 5 (Hotho et al., 2003) are discarded. Figure 3 illustrates the FixLC algorithm in details.

```
select the set λ(cd) of all hypernyms from each chain cd
select the set β of all synsets that appear in at least half of cd
if (β = ϕ) then δ = λ(cd) else δ = β
perform cut-off items in δ based on a chosen limit, producing ε
if (ε = ϕ) then Ω = δ else Ω = ε
extract the set α of all maximally occurring synsets in Ω
select the set γ of maximally deepest synsets in α
return a random synset from γ as representing the chain cd
```

Figure 3. Fixed Lexical Chains Algorithm (FixLC).

3.3 Semantic Dispersion

To explain (iv), we consider a document d. For each $1 \leq i \leq NSynsets$, we define $h(d,i)$ to be the histogram of relative distances (between 0 and 1) between consecutive occurrences of syn_i in document d. For this process, the number of bins of $h(d,i)$ and $h(e,j)$ will be the same for any 2 documents d, e, and synsets i, j. Also, for $h(d,i)$, if syn_i does not occur in document d, then the histogram consists of all 0's. Document d is then represented by the normalized concatenation of $h(d,syn_1)$, $h(d,syn_2)$, …, $h(d,syn_{NSynsets})$.

We note that synsets occurring once present a problem, so we treat them in two ways: we either ignore them or not. To make sure these issues were covered, we explored three variations, considering the distances of synsets for each kind of chain (FlexLC, FixLC and Flex2Fix): (i) ignoring single occurrences of synsets, (ii) single occurrences of synsets have distance 0 from themselves and (iii) not ignoring single occurrences and treating all synsets as having a 0 relative distance from themselves. An example for each approach is shown in Table 1, which uses a 4-bin histogram for the same vector of 4 synsets illustrating (i), (ii) and (iii). For every synset, each histogram bin is initialized to 0.

Each bin is represented by a half-closed, half open set of relative distance ranges. Bin 1 corresponds to the set [0,0.25), bin 2 to the set [0.25,0.5), bin 3 to the set [0.5,0.75), and bin 4 to the set [0.75,1). Since each distance occurring in a synset string of length n is at most $n-1$, the largest relative distance possible is $(n-1)/n$, which approaches 1 as $n \rightarrow \infty$. Synsets which do not occur in a string, will have 0's in all bins. In a nutshell, what our approach does is to characterize the spatial distribution (dispersion) of synsets in a document, using a histogram to keep track of those synsets by their relative distances. We note that using relative distances levels the representation playing field for all sizes of documents and treats them equally.

Map Type	Sequence of Synsets	Raw Distances	4-Bin Histogram Representation
I	$S_1S_2S_2S_4S_2S_3S_1$	S_1<6>S_2<1,2>S_3<>S_4<>	<0,0,0,1>|<1,1,0,0>|<0,0,0,0>|<0,0,0,0>
II	$S_1S_2S_2S_4S_2S_3S_1$	S_1<6>S_2<1,2>S_3<0>S_4<0>	<0,0,0,1>|<1,1,0,0>|<1,0,0,0>|<1,0,0,0>
III	$S_1S_2S_2S_4S_2S_3S_1$	S_1<0,6>S_2<0,1,2>S_3<0>S_4<0>	<1,0,0,1>|<2,1,0,0>|<1,0,0,0>|<1,0,0,0>

Table 1. Example of Mapping Distribution of Synsets in a 4-Bin Divided Document.

4 Experiments

To evaluate the proposed approaches, we used a corpus of 30 distinct documents from Wikipedia[2]**.** These are distributed equally in three major categories: *dogs*, *computers* and *sports*. The *html* files of these pages were saved and parsed, so stopwords (e.g. "a", "an", "the") could be removed. One might point out the small number of documents that comprise our corpus, in comparison with datasets used by statistical approaches in document similarity. However, we are proposing a semantic approach, in which every word has all its synsets examined by our algorithm. For our synset experiments, the number of synsets in our term/document matrix ranged between 1284 and 7490. In addition, the documents considered in this paper have, on average, 7,200 words each, which can produce a considerable dataset to process.

As explained in Section 3, during the preprocessing step we only maintain the nouns for each document having a synset match in WN. These steps help to remove features that do not contribute to our approach. By the end of this phase, our corpus has a total of approximately 216K words, of which 68K (nouns) have a match in WN. Table 2 shows in detail the documents/words used.

Wikipedia Category	Number of Documents	Number of Words	Nouns Matched in WN	Avg. of Nouns in WN (%)
Dogs	10	48,650	16,239	34.37
Computers	10	79,332	24,331	31.11
Sports	10	88,532	28,266	32.38
Total	**30**	**216,514**	**68,836**	**32.62**

Table 2. Wikipedia Dataset Details.

After all datasets are properly cleaned, we extract the BSID representation (Section 3.1), which is used as a base for all our lexical chains scenarios: FlexLC, FixLC and Flex2Fix. Once all flexible lexical chains are extracted from the documents, they are used to map into a fixed lexical chain structure and to create the corresponding vector representations. We also derive FixLC directly from BSID vectors, using a fixed chain size, as shown in Section 3.2.

In our experiments, we validated our various approaches by performing a clustering task, using 256 bins for our synset-based techniques. As mentioned previously, we had documents from 3 major categories, so we performed a variant of *k-means* clustering for *k=3* clusters and evaluated the resulting clustering using both the *Adjusted Rand Index* and the *Mean Individual Silhouette* values. The former metric is a measure of similarity between two clusters. We compared the derived clusters to the 3 ground truth clusters, consisting of all the *dog* documents, all the *computer* documents, and all the *sport* documents. The latter metric sees how well the clusters are designed, determining whether documents in the same cluster are close together, while documents in different clusters are far apart.

We used *spherical k-means clustering* (Hornik et al., 2012), as this technique uses the cosine distance (Han and Karypis, 2000) rather than Euclidean distance, and which has shown good results in clustering documents.

To validate the proposed algorithm, we also designed, implemented, and extended traditional approaches for document similarity, such as: BOW with all words (minus the stop-words) in the documents (BOWR), BOW with only matched nouns in WN (BOWN), BOW with the first synset match (most commonly used by other researchers) in WN (BOWS) and BOW with the BSID (BOWB) extracted from the BSD algorithm. Since the traditional approaches are variations of counts, only one bin is considered for these histograms. Table 3 provides a summary of all experiments performed. Figure 4 shows a scatter plot of these results. These results show that various permutations of our general approach worked better than others, and that four of our approaches stand out as better than the others.

[2] https://doi.org/10.7302/Z26W980B

Label	Algorithm	Adjusted Rand Index	Mean Individual Silhouette
A	Pure Flex--Method III	1	0.1908
B	Pure Flex--Method II	1	0.1775
C	BOW-N--Nouns in Wordnet	1	0.1757
D	BOW-B--Best Synsets	1	0.1686
E	Flex-2-Fixed--Method I	0.8981704	0.3964
F	Flex-2-Fixed--Method III	0.8981704	0.3878
G	BOW-R--Raw Words	0.8981704	0.1591
H	Flex-2-Fixed--Method II	0.8066667	0.3578
I	BOW-S--WordNet First Synset	0.6671449	0.1542
J	Pure Flex--Method I	0.6590742	0.1826
K	Pure Fixed--Method I	0.6044735	0.2137
L	Pure Fixed--Method III	0.5165853	0.2734
M	Pure Fixed--Method III	0.40252	0.2743

Table 3. Adjusted Rand Index and Mean Individual Silhouette.

The following observations are quite apparent:

- Three out of the four results with perfect clustering are from our techniques. Two of these perfect clusterings use flexible chains (considering their variations) while the third perfect clustering results from the methodology of finding the best synset representation for a document.
- The only perfect clustering result which is on the Pareto front (not dominated by another result), is the one which uses the third approach (iii) for extracting flexible chains.
- The clustering with the maximum silhouette value results from our first approach (i) to our technique for extracting Flex2Fix. This clustering is also on the Pareto front.
- The only clusterings on the Pareto front result from our techniques.

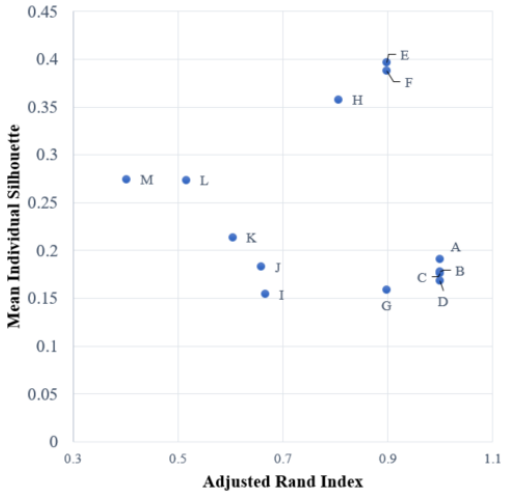

Figure 4. Scatter plot for Table 3 data.

5 Final Considerations and Future Work

In this paper, we explored how extracted semantic features can aid in document retrieval tasks. Furthermore, we presented several contributions on how these features can be extracted to form more robust lexical chains. First, we explored the notion of WSD and how to represent words, considering the effect of their immediate neighbors in their meaning (BSD). Second, a new methodology to transform variable length size semantic chains (FlexLC) into fixed parametrized structures is proposed through the Flex2Fix algorithm. Third we proposed an algorithm to derive FixLC directly from semantic representations. Also, three variations of how to calculate the relative distance of those chains were explored. To establish a comparison with the proposed approaches, we compared them with traditional ones, such as BOW and its variations (R/N/S/B). The comparisons showed that several of our approaches were the best performers.

Even though our model presents good results, we only touched the many possibilities that semantic features can offer in document retrieval and analysis. In future work, we intend to extend current algorithms for a more accurate representation of synsets and more solid lexical chains (fixed and flexible). In addition, we can also explore the effects of different WSD algorithms (e.g. Palmer, Leakcock & Chodorow, Lin, Resnik, Li) in the BSID choice and the construction of lexical chains (Meng et al., 2013). Other interesting linguistics challenges can be explored through the use semantic content extraction, such as: authorship identification, authorship profiling, clustering by concept structure, document summarization through concepts, and many other questions. The use of concepts, indeed, brings an interesting set of options that demands more time invested, so that its full potential can be reached.

References

Iyad AlAgha and Rami Nafee. 2014. An Efficient Approach For Semantically-Enhanced Document Clustering By Using Wikipedia Link Structure. *International Journal of Artificial Intelligence & Applications*, 5(6):53–62.

Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2015. Composing Measures for Computing Text Similarity. Technical report, Darmstadt.

Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, 17(48):10–17.

Alexander Budanitsky and Graeme Hirst. 2001. Semantic distance in WordNet : An experimental , application-oriented evaluation of five measures. *Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics*, 2(12):29–34.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Journal Computational Linguistics*, 32(August 2005):13–47.

Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman. 1988. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285, New York, New York, USA. ACM Press.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*.volume 71. MIT Press, Cambridge.

Christiane Fellbaum. 2010. Theory and Applications of Ontology: Computer Applications. *Media*(2000):231–243.

William I. Grosky and Terry L. Ruas. 2017. The Continuing Reinvention of Content-Based Retrieval: Multimedia Is Not Dead. *IEEE MultiMedia*, 24(1):6–11, January.

Weiwei Guo and Mona Diab. 2011. Semantic Topic Models : Combining Word Distributional Statistics and Dictionary Definitions. In *EMNLP '11 Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 552–561, Edinburgh.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in english*. Longman Group, London.

EH Han and George Karypis. 2000. Centroid-based document classification: Analysis and experimental results. *Principles of Data Mining and Knowledge Discovery*.

Graeme Hirst and David St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet - An Electronic Lexical Database*(April):305–332.

Kurt Hornik, Ingo Feinerer, Martin Kober, and Christian Buchta. 2012. Spherical k-Means Clustering. *Journal of Statistical Software*, 50(10):1–22.

Andreas Hotho, Steffen Staab, and Gerd Stumme. 2003. Wordnet improves Text Document Clustering. In *In Proc. of the SIGIR 2003 Semantic Web Workshop*, pages 541–544.

Jay J Jiang and David W Conrath. 1997. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *Proceedings of International Conference Research on Computational Linguistics*(Rocling X):19–33, September.

Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. *Proceedings of ICML*:296–304.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, page 279–es, Morristown, NJ, USA. Association for Computational Linguistics.

Lingling Meng, Runqing Huang, and Junzhong Gu. 2013. A Review of Semantic Similarity Measures in WordNet. *International Journal of Hybrid Information Technology*, 6(1):1–12.

Dan Moldovan and Adrian Novischi. 2002. Lexical Chains for Question Answering. *Coling*:1–7.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17:21–48.

Roberto Navigli. 2009. Word sense disambiguation. *ACM Computing Surveys*, 41(2):1–69, February.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity. In *Demonstration Papers at HLT-NAACL 2004 on XX - HLT-NAACL '04*, number July, pages 38–41, Morristown, NJ, USA. Association for Computational Linguistics.

Nitesh Pradhan, Manasi Gyanchandani, and Rajesh Wadhvani. 2015. A Review on Text Similarity Technique used in IR and its Application. *International Journal of Computer Applications*, 120(9):29–34.

Philip Resnik. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. , 1, November.

Terry Ruas and William Grosky. 2017. Keyword Extraction Through Contextual Semantic Analysis of Documents. In *Proceedings of the 9th International Conference on Management of Emergent Digital EcoSystems*, Bangkok. ACM Press (To Appear).

Julian Sedding and Dimitar Kazakov. 2001. WordNet-based Text Document Clustering. *Proceedings of the 3rd Workshop on RObust Methods in Analysis of Natural Language Data*(1999):104–113.

H Gregory Silber and Kathleen F McCoy. 2000. Efficient Text Summarization Using Lexical Chains. *Proceedings of the ACM Conference on Intelligent User Interfaces*:252–255.

Joe Tekli. 2016. An Overview on XML Semantic Disambiguation from Unstructured Text to Semi-Structured Data: Background, Applications, and Ongoing Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1383–1407, June.

Tingting Wei, Yonghe Lu, Huiyou Chang, Qiang Zhou, and Xianyu Bao. 2015. A semantic approach for text clustering using WordNet and lexical chains. *Expert Systems with Applications*, 42(4):2264–2275, March.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* -, pages 133–138, Morristown, NJ, USA. Association for Computational Linguistics.