Enhanced word embeddings using multi-semantic representation through lexical chains

Terry Ruas[a,c,*], Charles Henrique Porto Ferreira[b], William Grosky[a], Fabrício Olivetti de França[b], Débora Maria Rossi de Medeiros[b]

[a] *University of Michigan - Dearborn, 4901 Evergreen Rd, Dearborn, MI 48128, USA*
[b] *Federal University of ABC, Av. dos Estados, 5001, Santo André', SP, 09210-580, Brazil*
[c] *University of Wuppertal, Rainer-Gruenter-Str. 5, 42119, Germany*

Abstract

The relationship between words in a sentence often tells us more about the underlying semantic content of a document than its actual words, individually. In this work, we propose two novel algorithms, called *Flexible Lexical Chain II* and *Fixed Lexical Chain II*. These algorithms combine the semantic relations derived from lexical chains, prior knowledge from lexical databases, and the robustness of the distributional hypothesis in word embeddings as building blocks forming a single system. In short, our approach has three main contributions: (i) a set of techniques that fully integrate word embeddings and lexical chains; (ii) a more robust semantic representation that considers the latent relation between words in a document; and (iii) lightweight word embeddings models that can be extended to any natural language task. We intend to assess the knowledge of pre-trained models to evaluate their robustness in document classification task. The proposed techniques are tested against seven word embeddings algorithms using five different machine learning classifiers over six scenarios in the document classification task. Our results show the integration between lexical chains and word embeddings representations sustain state-of-the-art results, even against more complex systems.

*Corresponding author
Email addresses: truas@umich.edu (Terry Ruas), charles.ferreira@ufabc.edu.br (Charles Henrique Porto Ferreira), wgrosky@umich.edu (William Grosky), folivetti@ufabc.edu.br (Fabrício Olivetti de França), debora.medeiros@ufabc.edu.br (Débora Maria Rossi de Medeiros)

Keywords: Lexical chains, natural language processing, word embeddings, document classification, synsets

1. Introduction

Text Mining is a sub-area of Data Mining focused on extracting knowledge from text documents automatically. Its usefulness is present in several areas in the Natural Language Processing (NLP) domain, such as detecting spam email [17], authorship identification [50], and text summarization [24]. Among these areas, text classification has received considerable attention and has been the subject of several recent research topics [49, 20, 11, 46]. In the document classification task, the goal is to create a model using features extracted from a set of text data, often referred to as a training set, capable of inferring the correct labels of unseen text documents. The success of this model depends on how it performs on a specific task, and it generalizes to other scenarios. Feature extraction from text data is a challenge on its own in document classification.

In the last few years, word embeddings-based approaches have become popular to represent words in low-dimensional spaces [2, 30, 37]. The core element of these techniques relies on using neural network models to represent words according to their context (neighboring words) in a dense vector space, with each of its values corresponding to an intrinsic property of the word. Word embeddings techniques have alleviated some problems (e.g., scalability, lack of semantic relationships) of traditional count-based methods, such as bag-of-words (BOW). BOW-inspired approaches often use all the words in a corpus as features to represent documents, usually considering term-frequency and inverse-document-frequency (tf-idf) as a normalization factor in their weighting scheme. Nevertheless, neural network-based models require a considerable amount of data to accurately produce a good representation for words and their contexts, which can be troublesome for specific domains. Thus, methods that can derive results from more than one domain are beneficial.

Transfer learning presents itself as an exciting alternative to mitigate the

data scarcity issue, relaxing the underlying assumption used in machine learning that requires training and test data to be from the same domain, vector space, and similar distribution [34]. The same way humans take advantage of their previous experiences to solve future ones, transfer learning allows us to use what was learned from one task to do another. All the proposed and compared systems in this work make use of transfer learning by producing word vector representations moving from one domain to another.

One of the main challenges in representing documents is the ability to incorporate semantic aspects and to extract the relationships between words. While BOW techniques fall short in obtaining semantic knowledge from a domain, word embeddings often ignore the relationship effects of word order. One direction that tries to mitigate such issues is the construction of lexical chains [13, 41, 42]. *Lexical chains* is a sequence of related words delineating portions of text for a specific topic in a cohesive manner [32]. A document often produces continuity in its ideas if its sentences are semantically connected, providing good cohesion among them. *Lexical cohesion* usually occurs in words close to each other in a text, especially those adjacent to one another.

In this work, we propose two novel algorithms that combine the benefits of word embeddings, transfer learning, and lexical chains to solve the document classification. For this, we provide a model that allows us to represent any text document using the semantic relation between its words. The main idea behind lexical chains is to capture, in a more concise way, essential features to connect the meanings between words in the text. This semantic representation is obtained by one of the two proposed lexical chain algorithms, *Flexible Lexical Chain II* (FLLC II) and *Fixed Lexical Chain II* (FXLC II). While the former takes advantage of an external lexical database to derive its word relations, the latter defines them within a pre-defined semantic space. Besides, both proposed techniques incorporate state-of-the-art word embeddings algorithms to represent their chains. We tested our approaches on six multi-class datasets and compared our techniques against seven state-of-the-art pre-trained models over five machine learning classifiers. Our experiments show that our lexical

chain semantic representation improves the overall results in five out of the six datasets. The main contributions of our work are threefold:

1. Two new algorithms to extract the semantic relations between words of text documents combining the benefits of lexical chains, word embeddings, and the prior knowledge of external lexical databases.

2. Detailed experimental validation of our techniques in the document classification task against seven different systems and six multi-class datasets.

3. A collection of lightweight word embeddings models (75% smaller) and the source code for the proposed techniques are publicly available[1].

2. Related work

The idea to extract semantic information from a text and from this to construct features to refine document representations has been receiving much attention in the NLP arena. Techniques that can use the words in a document and their relations provide a more robust representation than count-based ones since they penetrate beyond the syntax layer. Moreover, systems that can combine prior knowledge, obtained from different tasks or problems, can leverage even more the representation of words and documents.

The adoption of lexical chains is present in several NLP tasks, such as: word similarity [35], keyword extraction [41], and document clustering [15]. More recently, some publications use lexical chains in a more specific context. Bär et al. [1] explore the use of text similarity metrics in the context of semantic text similarity in text re-usability detection. They argue text similarity cannot be considered a unified entity. Instead, one should evaluate at which level two documents are semantically similar or not.

Bag-of-words (BOW) uses words as features to represent documents in a vector space model [16]. Typically, the elements' weights can be adjusted according

[1] `https://github.com/truas/LexicalChain_Builder`

to their relevance, such as the number of occurrences and normalized counting. Albeit BOW proposes a simple but robust approach, it has some drawbacks, such as lack of semantic information, sparsity, and scaling problems. In this context, Latent Dirichlet Allocation (LDA) [3] comes as an alternative to represent documents as a collection of topics according to their content. LDA provides a statistical model distribution over the many hidden topics in a text document based on the distribution of the co-occurrences of its words. The distributions of these words will compose the topics of the actual corpus, which is expected to represent the entire text collection in a much smaller vector space.

The robustness of context-predictive models against classic count-vector-based distributional ones has been tested in several NLP tasks and proven to be quite effective [26]. Among the neural network models (context-predictive), some should receive special attention, such as word2vec [30]. Word2vec embeds words in a dense vector space of low dimensionality through one of the two following techniques: continuous skip-gram and continuous bag-of-words (CBOW). In the skip-gram model, one tries to predict the neighboring context of a word given a target word, while CBOW does the converse, predicting a word given its context. As influential as word2vec, GloVe [37] builds a co-occurrence matrix of the words in a corpus, to calculate the ratio between the probabilities of these words. While word2vec is focused on fixed context windows to derive its vectors, GloVe takes advantage of a global perspective. As for their results, both word2Vec and GloVe present similar findings in various NLP tasks [18, 6].

Even though the applicability of lexical chains is diverse, we see little work exploring them with recent advances in NLP, more specifically with word embeddings. In [45], lexical chains are built using specific patterns found on WordNet [10] and used for learning word embeddings. Their resulting vectors, as ours, are tested in the document similarity task. Gonzales et al. [13] use word-sense embeddings to produce lexical chains that are integrated with a neural machine translation model. The combination of word-sense vectors and lexical chains improves their translation accuracy by 0.43% and 0.6% for German $\rightarrow$ English and German $\rightarrow$ French, respectively. Even though the papers mentioned use

lexical chains, they do not explore them in the document classification task. We provide two novel algorithms that are built in a bottom-up approach, inspired by the Morris & Hirst [32] definition of lexical chains (Section 3.1). Additionally, we derive a multi-sense embedding representation for our lexical chains and compare them with state-of-the-art word embeddings techniques.

Extending word2vec's techniques (skip-gram and CBOW), *Paragraph Vectors* (PV) is an unsupervised framework that learns continuous distributed vector representations for any size of text portion (e.g., paragraphs, documents) [22]. This technique alleviates the inability of word2vec to embed documents as a unique object. Differently than word2vec, PV produces a fixed-length n-dimensional vector representation for each entire-textual segment, instead of just the words in the corpus. Le & Mikolov [22] algorithm is also available in two different forms: Distributed Bag-of-Words of Paragraph Vectors (PV-DBOW) and Distributed Memory Model of Paragraph Vectors (PV-DM), which are analogous to the skip-gram and CBOW models in word2vec respectively.

Even though PV [22] has reported superior results when compared to the original word2vec [31] approach, its impact was smaller than its predecessor in the NLP community. The widespread use of word2vec is due to, among other things, its efficient log-linear neural network language model, low-dimensional vector representation, resource-friendly implementation, and effectiveness in several NLP downstream tasks [18, 25, 43, 39]. For now, we decided to compare the proposed techniques in this paper with more accessible, robust and disruptive approaches that work at least in a word-level.

Following the opposite direction of paragraph vectors, some publications moved from a direct document representation to a sub-word one. In fastText [4], they extend the skip-gram model by proposing a word representation obtained from a sum of the n-grams of its constituent sub-word vectors. For example, using $n = 3$, the word *kiwi* would be represented as $\{< ki, kiw, iwi, wi >\}$ and the word *kiwi* itself as a particular case. Adopting a similar method, Peters et al. [38] propose to represent words through its constituent characters with ELMo (Embeddings from Language Models). ELMo uses a two-layer deep

bi-directional Language Model (biLM) with character convolutions as a linear function of their internal states. Because of its architecture, ELMo does not keep a word-dictionary in their model, only characters. Therefore it can handle any out-of-vocabulary (OOV) words by averaging all biLM layers for the constituent characters of a word. More recently, Universal Sentence Encoder (USE) [5] proposes two encoding models to represent any text as vectors, one focused on accuracy (Transformer architecture) and the other on inference (Deep Averaging Network). The former builds a sub-graph that uses attention mechanisms to compute the context representations of words in a text, considering their ordering and identity. The latter average words and bi-grams embeddings and feed them in a feed-forward deep neural network to produce new embeddings. In Section 4, we compare our proposed techniques, trained in a simple word2vec implementation, against the aforementioned state-of-the-art models.

One limitation with current word embeddings techniques is they condense all meanings of a word in single vector representations. Traditional algorithms rely on the context of the words to help express their multiple purposes in the n-dimensions of their vectors. An alternative to mitigate such a problem is to use labeled word-senses in a corpus to train word embedding models. However, labeled word-senses on a large scale are scarce, and human disambiguation is expensive, time-consuming, and subjective. Recently, Ruas et al. [43] proposed a new technique called *Most Suitable Sense Annotation* (MSSA) that performs unsupervised word-sense disambiguation and annotation in words from any part-of-speech (POS), which are later used in a word embeddings algorithm to improve the quality of traditional word embeddings model generation. Ruas et al. [43]'s technique finds the most suitable word-sense using the word's context, the semantic relationships expressed in WordNet [10], and a pre-trained word embeddings model (e.g., Google Vectors [30]). In addition to the vanilla MSSA technique, the authors also presented two other variants of their approach, named MSSA-1R and MSSA-2R. Their objective is to refine the word embeddings model from MSSA through an iterative process. The MSSA algorithm and its variations were tested in a word similarity task in six different

benchmarks against a large number of systems, producing state-of-the-art results [43]. The lexical chain algorithms proposed in this paper (Sections 3.1.1 and 3.1.2) are built using the disambiguation process from MSSA and their multi-sense word embeddings models, which are all publicly available[2].

In [12], they propose a technique to generate document representations called Bag-of-Meta-Words (BoMW). In this technique, they use pre-trained word embeddings models (e.g., word2vec, GloVe) to retrieve and average the vector representation of the constituent words of a document. They map each vector into a different feature space and sum the vectors from this new space to represent a document. They propose two approaches, namely the Naive Interval Meta-Word Model and Features Combination Meta-Word Model. The former discretizes features from the initial word embedding, while the latter clusters related features under the same group. They compare their results on three datasets against BOW, neural networks models (e.g., recurrent neural networks), and an average of word embedding models. Their approach shows an improvement in accuracy over the baselines on the document classification task.

Some techniques make use of context information in the document, to incorporate semantic aspects into traditional word embeddings models. In [21], they build a content tree word embedding to capture local information from the words in a document. Their word vectors are calculated as a weighted average of a word's vector and its immediate parent. They assume that the context, represented by a word parent node, influences its neighboring words. The insertion of a new word into an existing content tree only happens if there is a high correlation between all nodes in that tree and the new word. Enríquez et al. [9] explore the complementary information of BOW and word2vec to represent their document. Their technique uses a simple voting system that averages the confidence values returned by BOW and word2vec to classify a text either in a negative or positive class. They concluded that BOW provides the best representation over word2vec, while their combination improves the overall results.

[2] `https://github.com/truas/MSSA`

To the best of our knowledge, this is the first work that combines lexical chains and word embeddings applied to the document classification problem. We expect the proposed algorithms to produce a robust semantic representation through the use of lexical chains. Furthermore, we build our lexical chains using several synset objects[3] in the lexical database in addition to hypernyms and hyponyms, which are commonly used in the literature [13, 27, 45] (detailed in Section 3). The main idea is to bring the semantic relations of lexical chains to traditional word embeddings techniques, leveraging their vector representation, and improving the overall result in the document classification task.

3. Semantic embeddings through lexical chains

In this work, we propose a novel approach to capture the semantic relationship between tokens from a text. Our techniques combine the use of word embeddings, lexical chains, and the prior knowledge of lexical databases, to derive the relations between words. This is done through two algorithms: *Flexible Lexical Chains II (FLLC II)* (Section 3.1.1) and *Fixed Lexical Chains II (FXLC II)* (Section 3.1.2). Both algorithms are inspired by the approaches proposed in [41, 42]. While the lexical chains explored in [41, 42] consider only one part-of-speech (POS) (nouns) and *hypernyms*, ours, on the other hand, are able to deal with any POS tag, incorporate word embeddings, and also include other 19 lexical objects[3] from the English WordNet[4], [10]. As a result, we leverage the semantic representation of words, sentences, paragraphs, and entire documents through the use of lexical chains.

We illustrate an overview of the entire process in Figure 1. In both cases, FLLC II and FXLC II, the constructed chains are represented by one of their

[3] *hypernyms, instance_hypernyms*, *hyponyms*, *instance_hyponyms*, *member_holonyms*, *substance_holonyms*, *part_holonyms*, *member_meronyms*, *substance_meronyms*, *part_meronyms*, *attributes*, *entailments*, *causes*, *also_sees*, *verb_groups*, *similar_tos*, *topic_domains*, *region_domains*, and *usage_domains*

[4] `https://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html`

constituent elements, which is selected by considering their vector representation in a pre-trained token embeddings model (e.g., word2vec). Later, we feed the output of our techniques, i.e., the synset-annotated corpus, into a word embeddings algorithm, which produces a lexical chain embeddings model. Later, the synset-vectors are used as features in the document classification task.

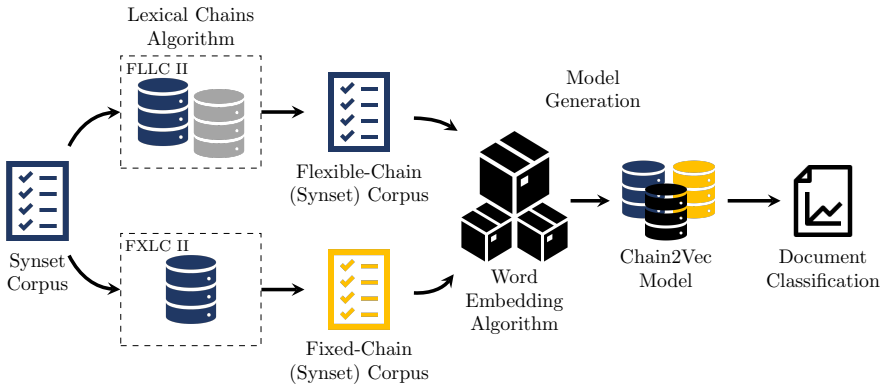


Figure 1: Overview architecture for building lexical chains

FLLC II and FXLC II techniques are, at their core, reducing the number of valid elements (words) in a document. With fewer features, our approaches can reduce the number of necessary tokens to represent a document and still keep their semantic content. Given the characteristics mentioned above, we can classify the FLLC II algorithm as a soft-dimension reduction technique, while FXLC II as a hard-dimension reduction. Another critical difference between the proposed methods is their databases and how they are used. FLLC II chains are generated using a lexical database (e.g., WordNet) and a pre-trained token embeddings model represented by the gray and blue figures, respectively. While the former is responsible for providing the semantic relationship between the tokens in the document, the latter helps us to decide which token will represent our chain. In both algorithms, the blue figure represents a trained synset

embeddings model[5]. In the FXLC II technique, since we do not group tokens directly by their semantic affinity, we do not require a lexical database with word relationships, such as WordNet [10]. Instead, we need a compatible pre-trained token embedding model, concerning the document processed, to build our chains. Therefore, FXLC II can be exported to other scenarios using any pre-trained token embeddings model (e.g., words, synsets), as long as it contains the elements from the documents. In the next sub-sections, both techniques are explored in-depth, followed by a simple toy example illustrating their operation.

3.1. From document to lexical chains

Lexical chains are built according to a series of relationships between words in a text document. In the seminal work of Morris & Hirst [32] they consider an external thesaurus as their lexical database to extract these relations. A lexical chain is formed by a sequence of words $\{w_1, w_2, \ldots, w_n\}$ appearing in this order, such as any two consecutive words w_i, w_{i+1} possess the following properties[6] [32]: (a) two words share one common category in their index; (b) the category of one of these words points to the other word; (c) one of the words belongs to the other word's entry or category; (d) two words are semantically related; and (e) their categories agree to a common category.

The main goal of FLLC II and FXLC II is to represent a collection of words by their semantic values more concisely. Even though FLLC II and FXLC II outputs format are the same, they explore different aspects in capturing the lexical cohesion in a text. In FLLC II, the semantic sets (lexical chains) are assembled dynamically according to the semantic content for each token evaluated and the relationship with its adjacent neighbors. As long as there is a semantic relation that connects two or more words, they should be combined into a unique concept. If a word without any semantic affinity with the current chain presents itself, we must start a new lexical chain to capture this new idea.

[5] A word embedding algorithm using an annotated-synset corpus.

[6] Where category, indexes, and pointers are attributes in the lexical database considered.

On the other hand, in FXLC II, text documents are broken down into predefined chunks, with C words each, describing their semantic content. Different from FLLC II, the FXLC II technique groups a certain number of words into the same structure, regardless of their semantic affinity.

A requirement for both, FLLC II and FXLCII, is a disambiguated and synset-annotated corpus. We build our lexical chains according to how the word-senses (synsets) are related to each other in a sentence. For this reason, we make use of the MSSA algorithm [43] to transform our word-based corpus to its word-sense version composed of synsets. We also compared our approach against MSSA in our experiments (Section 4), so we can evaluate how our lexical chains compare with a state-of-the-art technique that produces multi-sense embeddings. Nonetheless, our proposed algorithms can be applied to any method that can disambiguate and annotate words according to their word senses.

3.1.1. Flexible Lexical Chains II (FLLC II)

The FLLC II algorithm works building semantic sets of tokens that present any level of semantic relation between them. The decision of incorporating a new word into a chain is dynamic and based on 19 lexical objects[3] in WordNet [10].

We extend Ruas & Grosky [41] flexible chain algorithm to use all POS and also incorporate word embeddings containing the vector representation of the tokens in a set of documents d. In its previous version, FLLC I was designed to work in the keyword extraction problem, could not handle POS different than nouns and was not able to incorporate word embeddings. As illustrated in Algorithm 1, we require a set of documents d represented by synsets (i.e. set of synonyms)[7], a lexical database ld, and a pre-trained synset embeddings model tsm[8]. To transform a word-based-document into a synset-based one, we apply the MSSA algorithm proposed by Ruas et al. [43]. MSSA uses Google News vectors[9] to disambiguate words with multiple senses in a text. MSSA produces

[7] `https://wordnet.princeton.edu/documentation/wngloss7wn`
[8] `https://github.com/truas/MSSA`
[9] `https://code.google.com/archive/p/word2vec/`

the most appropriate word-sense representation of a word given its immediate context window. A context-window considers three words at a time, i.e., former $i-1$, center i, and latter $i+1$, in an overlapping manner, and uses the average vector of the words from the glosses to represent each word-sense. The word-sense from i, with the highest cosine similarity concerning its neighbors ($i-1$ and $i+1$), is selected to represent the central word i in the context window. More details about MSSA and its alternatives are provided in [43].

Once the synset embedding models are available, we can feed the system again, using the output vectors from previous passes, improving the disambiguation step in MSSA. Using a synset embedding model allows us to refine our representation and remove FLLC II and FXLC II from Google News vectors' dependency. We call this approach MSSA-NR, where N is the number of feedback iterations used. None of the related work (Section 2) nor compared systems in our experiments (Section 4) explore this recurrent characteristic in their systems.

Once we have a document represented as synsets, we build our flexible chains from the first to the last token. We start our current chain using S_1 (first synset in the document) to initialize the synset list used to represent the current chain *current_chain.synsets*, and a set of related synsets that map the semantic relation between consecutive synsets *current_chain.related* (lines 3:4). The synsets retrieved in $get\_related\_syns(S_i, ld)$ (including S_i) are a collection of extracted synsets from 19 attributes[3] in the lexical database ld (WordNet).

For each new synset evaluated, S_i, we extract their related synsets (including S_i), called *new_rel*, and verify if there are any common synsets with the related synsets in the chain being built (*current_chain.related*) (lines 5:7). In case the intersection between *current_chain.related* and *new_rel* is not empty, we add S_i and *new_rel* to *current_chain.synset* and *current_chain.related* respectively (lines 8:9). Otherwise, it means there are no common related synsets between the current chain and S_i, so, we understand the current chain must be properly represented and added to the list of flexible chains (line 11). Thus, we find the synset with the highest cosine similarity against the average of

all synset vectors in *current_chain.synsets*. The average of all synsets in *current_chain.synsets* is calculated considering a pre-trained synset embeddings model (tsm) for the function $get\_best\_rep(current\_chain.synset, tsm)$. The synset embedding model (tsm) used is produced using a Wikipedia (English) dump from 2010 [44] annotated into synsets with MSSA [43]. After representing and including the current chain in the list of flexible chains, we start to build a new chain, but now considering S_i (lines 12:13), the same method we use to start our algorithm.

After we iterate over all synsets in d, we also verify if there are elements in the current chain not added to our flexible chains list (line 14). This step mitigates the problem in which all synsets are combined in one single chain or the last synset S_n in the document is semantically related to the current chain being built. At the end of the FLLC II algorithm, we return a list of synsets representing all the lexical chains found in a document d (line 16).

The proposed algorithm for flexible chains improves its predecessor [41] in several aspects. We consider all POS when building our chains, instead of just nouns. We also consider 19 attributes[3] in WordNet for each synset evaluated, which is an improvement from most lexical chain systems that often focus on hypernyms and hyponyms only. In our version of lexical chains, we use the transfer knowledge from an external training task of word embeddings learned from the entire English Wikipedia.

3.1.2. Fixed Lexical Chains II (FXLC II)

As in the FLLC II algorithm (Section 3.1.1), FXLC II also builds its chains using a list of synsets. However, the FXLC II algorithm has a more general approach. The lexical chains in FXLC II are defined beforehand and do not require an explicit semantic relation between its synsets. In other words, we enforce the number of synsets for each chain (chunk) throughout the document instead of building each chain according to an existing semantic relation.

We extend the Ruas & Grosky [42] algorithm for fixed chains to all POS and incorporate word embeddings containing the vector representation of the

Algorithm 1 Flexible Lexical Chain II Algorithm (FLLC II)

Require: $d = \{S_1, \ldots, S_n\} : S_i \in ld$; tsm = trained synset embedding model; ld = lexical database

```
function FLLC II(d, tsm, ld)
    flexible_chains_list = ∅
    current_chain.synsets = [S_1]
    current_chain.related = {get_related_syns(S_1, ld)}
    for i = 2 to n do
        new_rel = {get_related_syns(S_i, ld)}
        if current_chain.related ∩ new_rel not ∅ then
            Add S_i to current_chain.synsets
            Add new_rel to current_chain.related
        else
            Add get_best_repr(current_chain.synset, tsm) to flexible_chains_list
            current_chain.synsets = [S_i]
            current_chain.related = {get_related_syns(S_i, ld)}
    if current_chain.synsets not ∅ then
        Add get_best_repr(current_chain.synsets, tsm) to flexible_chains_list
    return flexible_chains_list
```

tokens in our document. In its previous version, FXLC I was tested in a small scenario with only 30 documents, it required a corpus composed exclusively of nouns, and it did not account for word embeddings in its structure. To maintain consistency between our algorithms and experiments, we also use the MSSA algorithm Ruas et al. [43] to produce a list of synsets out of a given document d. Once we have a document d represented as synsets, we create a new document representation called a *chunked_document*, which is composed of chunks C_j of size cs. Analogous to Algorithm 1, we also find the synset to represent each fixed lexical chain C_j in *chunked_document* through the synset with the highest cosine similarity against the average of all synset vectors in C_j.

The proposed technique for fixed chains also surpasses its predecessor [42] in some aspects. As in FLLC II, we also consider all POS when building our

chains. Besides, the FXLC II technique does not rely on distance measures to calculate how far our synsets are from each other to represent each chunk [29]. Instead, based on the information provided from the synset vectors we find the closest semantic candidate in a chain for all its inner elements.

Since the FXLC II approach ignores the direct semantic affinity between synsets and groups them for each chunk C_j, this approach can be extended to other document representations as well. As long as the pair document-tokens and embedding models have the same representation, FXLC II can be applied.

3.2. From lexical chains to embeddings (Chains2Vec)

After transforming our documents into lexical chain representations, which are formed by synsets, we use them as a training corpus input in word2vec. As a result, word2vec uses these annotated documents to produce an embedding model based on synsets. The main idea of FLLC II and FXLC II is to obtain a better semantic representation for words from a large collection of documents that generalizes well enough to any NLP downstream task or problem.

In order to keep our vectors easy to interpret to other systems using synsets, we represent each token in our corpus using the same format as the one in Ruas et al. [43] when applying the MSSA technique: $word\#synset\_offset\#pos$, where $word$ is the word itself, normalized in lowercase; $synset\_offset$ is an 8 digit, zero-filled decimal integer that corresponds to a unique word-sense, and pos is a part-of-speech tag (e.g. n for nouns, v for verbs, a for adjective, s for adjective satellite and r for adverb)[10]. Since we are using an equivalent notation, the synset embeddings models produced with FLLC II and FXLC II results can be incorporated in other systems with the same format as well.

For now, we currently explore the document classification task. Still, we believe our chains can be applied to other domains the same way we used MSSA synset models [43] in our experiments. MSSA was designed for the word similarity task, but, in this work, we use and extend its synset models to the document

[10] `https://wordnet.princeton.edu/documentation/wndb5wn`

classification task. In the following section, we provide a toy example of how FLLC II and FXLC II are used to produce our lexical chains.

3.3. Building lexical chains

In this section, we provide an illustrative example for the FXLC II and FLLC II algorithms, as Figure 2 shows.

Let us consider the sentence: *Beets, carrots, and potatoes are grandma and grandpa's favorite dish for lunch!*(1). As explained before, we require a document composed of synsets to build our lexical chains. After cleaning our sentence example (2), by lowercasing all the words, removing all punctuation and common English stop-words, we apply the MSSA algorithm [43] (3), to obtain the proper synset for each word. Next, we use either the FLLC II or FXLC II.

For the FLLC II, we first extract all related synset for each synset in the sentence and group the ones with at least one overlap between them ((4a) - highlighted connections). The gray square represents the document synsets, and the dashed squares represent the related synsets. In the flexible version of the algorithm, we build the chains dynamically, so as long there is a common synset between two consecutive tokens, they will be part of the same chain. If there are no overlapping synsets in the documents, the chain will have just one element, i.e., the synset itself (**chain3**). Later, from a pre-trained synset model, we extract the vectors of the synset in each chain and calculate their average (centroid) (5a). Our final document representation will have only one synset per chain. Therefore, we select the closest synset-vector to its centroid according to their cosine similarity. We hypothesize that the closer a synset is to a chain centroid, the better semantic representation it provides (6a).

For FXLC II, we first define how many synsets each chunk will have and group them sequentially (4b). In this example, we are using a chunk-size of 3. In case there are fewer synsets than the actual chunk-size, then all synsets are included (**chain3**). Once the chains are formed (5b), we select one synset per chain as we do with the FLLC II algorithm. We average the vectors and select the synset closest to its centroid (6b).

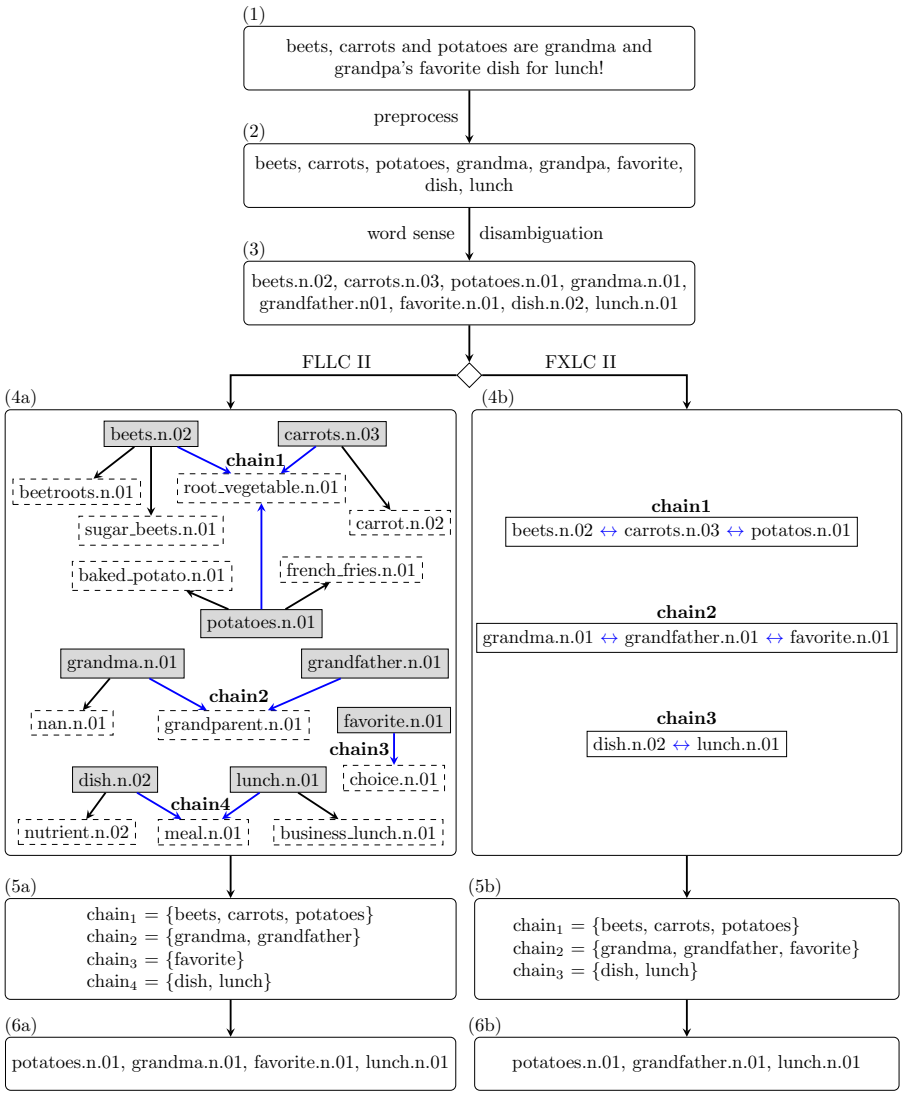


Figure 2: Practical example applying FLLC II and FXLC II algorithms on a single document.

4. Experimental details

In this section, we explain all the details and constraints of our experiments. We first describe the different aspects of each dataset used, their characteristics, references, and availability. Next, we provide an accurate description of the

chosen metrics, machine learning classifiers, and hyperparameters adopted. We include the procedures to fine-tune all experiments, classifiers, and discuss the main aspects of state-the-art systems compared to our proposed techniques.

4.1. Datasets Details

Our experiments considered 6 different datasets with specific characteristics which impose a particular challenge on each classification. Among these datasets, 4 of them (Ohsumed[11], 20NewsGroup[12], Reuters[11]and BBC[13]) represent benchmark datasets widely used on text classification problems and the other 2 (ScyGenes and ScyClusters) are formed by scientific papers abstracts in biology. Table 1 describes the main characteristics of each dataset.

Table 1: Technical details about the datasets after preprocessing.

Corpus	Subject	#docs	#classes	#tokens	#synsets
Ohsumed [19]	Medical abstracts	56984	23	64154	36395
20Newsgroups [19]	News	18846	20	129782	43413
Reuters-21578	News	9980	10	24273	21747
BBC [14]	News	2225	5	29126	29151
ScyClusters [28]	Biological abstracts	1655	7	13265	11428
ScyGenes [28]	Biological abstracts	1114	7	10553	10045

Table 1 shows the details of each dataset concerning their theme, number of documents, classes, tokens, and synsets. We use the term *token* instead of words because some features do not represent a proper word itself. As an example, we have the token *housd* that does not constitute an English word, but it is present in the dataset. Typos and malformed word-tokens are not able to be processed by techniques relying on proper syntax, but recent word embeddings models such as ELMo [38] and USE [5] can handle these issues. Column *#synsets* shows the number of synsets generated applying the MSSA algorithm [43]. As

[11] `http://disi.unitn.it/moschitti/corpora.htm`http://disi.unitn.it/moschitti/corpora.htm
[12] `http://qwone.com/~jason/20Newsgroups/`http://qwone.com/ jason/20Newsgroups/
[13] `http://mlg.ucd.ie/datasets/bbc.html`http://mlg.ucd.ie/datasets/bbc.html

a result, the number of synsets and tokens are different, and in most cases, it is smaller. Not every token (e.g., *housd*) has a mapping to WordNet, which forces their exclusion since no semantic relation available.

4.2. Machine learning classifiers

We considered five classifiers in our experiments. These classifiers were chosen among the most popular ones in the document classification arena: K-Nearest Neighbors (K-NN), Support Vector Machine (SVM), Logistic Regression (LR), Random Forests (RF), and Naive Bayes (NB).

All classifiers had their parameters fine-tuned to provide the best configuration for each technique. For this reason, we performed an extensive grid search on the training corpus to find out the most suitable hyperparameters for each classifier. Except for Reuters-21578 and 20Newsgroups, all data sets do not include a training and test split, so we applied k-cross-validation for k equal to 10. The classification performance for all machine learning classifiers and datasets was validated in the Friedman test and Nemenyi's posthoc test for a p-value of less than 0.05, thus certifying the statistical significance of our experiments [7].

We evaluate our models in the text classification task through their classification F1-Micro score as a metric. We also generated the accuracy and F1-Macro scores for all experiments. However, their variation was minimal, almost all values of accuracy and F1-Micro were the same, and the differences between these metrics followed the same delta among different systems. Due to space constraints and the aspects above, we decided to report just the F1-Micro score.

4.3. Word embedding models characteristics

This section presents the main characteristics of the compared systems used in the document classification task. We compared our techniques against seven state-of-the-art approaches which incorporate transfer learning at some level in their implementation. Table 2 summarizes the algorithms used in our experiments, named: Latent Dirichlet Allocation (LDA) [3], word2vec (w2v) [30], Global Vectors (GloVe) [37], fastText [4], Universal Sentence Encoder (USE) [5],

Table 2: Word embeddings used and their main characteristics. * For USE, [5] reports its training data as a collection of sources from Wikipedia, web news, web question-answer pages discussion forums and Stanford Natural Language Inference (SNLI) corpus.

Algorithm	Main Characteristics	Training Corpus	Dimensions
LDA	Probability distribution	Wikipedia Dump 2010	300
word2vec	Continuous Bag-of-Words (CBOW)	Google News	300
GloVe	Word-word co-occurrence matrix	Wikipedia dump 2014 + Gigaword 5	300
fastText	Skip-gram	Wikipedia Dump 2017 + UMBC	300
USE	Deep Average Network	Various sources*	512
ELMo	Bidirectional Long Short Term Memory	1 Billion Word Benchmark	1024
MSSA-1R	Most Suitable Sense Annotation -1R	Wikipedia Dump 2010	300
FL-1R	Flexible Lexical Chains II + MSSA-1R	Wikipedia Dump 2010	300
FX2-1R	Fixed Lexical Chains II + MSSA-1R	Wikipedia Dump 2010	300

Embeddings from Language Models (ELMo) [38], MSSA-1R (-1R) [43], FLLC II using the MSSA-1R synset embeddings model (FL-1R), and FXLC II with a chunk size equal to 2 and using the MSSA-1R embeddings model (FX2-1R).

For each system, we transformed each word in the documents into their vector representation according to an external pre-trained word embedding model. Once we obtained these word vectors, we averaged them to represent the entire document as one single entity. The systems compared in our experiments can be categorized into two major groups: (i) pre-trained word embeddings models and (ii) explicitly trained word embeddings models. In (i), we use out-of-the-shelf pre-trained models: word2vec[14], GloVe[15], fastText[16], USE [17], ELMo[18], and MSSA (-1R and -2R)[19]. For (ii), we trained the embedding models from scratch for the following techniques: LDA, FL-1R, and FX2-1R. All techniques have a 300 dimension word vector representation, except for USE and ELMo, which are only available in 1024 and 512 dimensions, respectively.

[14] https://code.google.com/archive/p/word2vec/
[15] https://nlp.stanford.edu/projects/glove/
[16] https://fasttext.cc/docs/en/english-vectors.html
[17] https://tfhub.dev/google/universal-sentence-encoder/2
[18] https://tfhub.dev/google/elmo/2
[19] https://github.com/truas/MSSA

For (ii), we produced our multi-sense embeddings using the same hyperparameter configuration in their word2vec training phase: CBOW training model, 300 dimensions, a window size of 15, the minimum word count of 10, and hierarchical softmax. Any attribute not mentioned was set to its default value according to the *gensim* API[20][40]. We also compare our techniques with the BOW approach in a separate experiment using the same classifiers and the original datasets (i.e., words). In addition to the presented models explored in the document classification task, we include variations of our techniques under a different scope to evaluate their characteristics more carefully (Section 4.6).

Since one of the main objectives in this paper is to evaluate pre-trained word embeddings models, Deep Learning (DL) techniques such as BERT [8] and XLNet [48] are not readily applicable. Additionally, the datasets in our task are considerably smaller when compared with those used in DL. Transformer-based architectures [47] require many epochs in their fine-tuning phase and significant processing time in dedicated hardware (i.e, GPU), which make them costly restrictive. Therefore, we did not investigate such methods in our experiments but are working towards their inclusion in the future.

4.4. Experiment configurations

In Section 4.5, we present the results for all variations of our models against a traditional BOW approach. In this scenario, we can analyze the different chunk sizes for our lexical chains and how the varieties of MSSA [43] affect our constructed lexical chains. This evaluation also supports the choice of our models' configurations for the comparison against the state-of-the-art systems. Thus, we compare our techniques (FL-1R, FX2-1R) with different word embeddings approaches that also use transfer learning. In Section 4.6, we provide a more in-depth assessment of how our techniques behavior. The idea is to offer two different perspectives: (a) chunk size for the FXLC II variations and (b) the effects of MSSA models (-1R and -2R) in both FLLC II and FXLC II.

[20] `https://radimrehurek.com/gensim/models/word2vec.html`

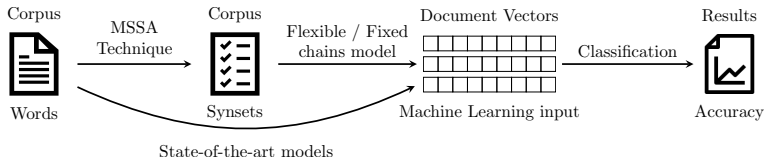


Figure 3: Pipeline for the document classification process in our experiments.

The same preprocessing steps are applied to all datasets and techniques so we can guarantee more consistency in our experiments. For preprocessing, we lower-case all words in the document, remove all common English stop-words and punctuation. We represent each document as the average of its constituent word vectors using a pre-trained embeddings model. Averaging the word vectors of a document to obtain its representation is a common strategy adopted in the document classification task [22, 46]. Next, we feed them to several machine learning classifiers and evaluate their F1-Micro score, as shown in Figure 3.

While the compared systems derive vectors directly from the words in each dataset, our proposed techniques require a disambiguated synset-annotated corpus. Thus, an additional step before the machine learning classifiers is necessary. We disambiguate the words from each document to obtain their respective synsets, using the MSSA [43] algorithm. Next, as in all the compared systems, we derive the synset vectors using the pre-trained lexical chain embedding models (FLLC II and FXLC II) created from the Wikipedia dump.

We compare our findings against classical and state-of-the-art word embedding techniques that consider transfer knowledge. For traditional word embeddings techniques, we use LDA [3], word2vec [30] and GloVe [37]. For the state-of-the-art word embedding techniques, we include fastText [4], ELMo [38], and USE [5]. Furthermore, we also provide a comparison of our results against a traditional BOW technique, which relies only on statistical information of the training corpus. Even though BOW does not necessarily use transfer learning from a prior task, it is probably the most used technique to represent a collection of documents in NLP. Also, the curse of dimensionality affects BOW because of the number of words in the considered vocabulary. In this manner,

depending on the size of the dataset, its adoption might not be suitable. To mitigate the dimensionality problem and allow a fair comparison among the compared systems, we created a BOW representation considering the top 300 features (words), ordered by term frequency, and applying tf-idf as its weighting scheme [36]. The only two models that do not follow this vector-length representation are ELMo and USE, with 1024 and 512 dimensions, respectively.

4.5. Document classification task results

Tables 3 and 4 present the results of our experiments considering the document classification task. The results are organized as follows. Each block illustrates the results of applying all classifiers on a specific dataset. Each column represents a different word embeddings model benchmark to compare against our techniques (last two columns). Values in **bold** represent the best results in a row, and underlined values the best results in a column for a specific dataset.

As explained in Section 3, our proposed techniques are built considering a synset-annotated corpus obtained through the MSSA algorithm [43]. For the FXLC II algorithm (Section 3.1.2), we also examined the sizes of 2, 4, and 8 for the number of synsets in each chain (*chunk_size*). Because of space limitations in Table 3, the results for the chunk size of 8 for FXLC II were not included. However, we analyze their behavior for all chunk sizes in Section 4.6. To evaluate the best configuration for the proposed techniques, we first compared them with a traditional BOW with a tf-idf weighting scheme and 300 dimensions. We decided to keep 300 dimensions to maintain all models under similar constraints.

As Table 3 shows, BOW sustains good results for ScyGenes, and a few for ScyClusters. This behavior seems reasonable since ScyGenes and ScyClusters are the smallest datasets considered. These datasets might contain unique keywords among their categories, resulting in a reasonable classification for the BOW approach. On the other hand, as the number of documents increases, our semantic embedding representations start to overcome BOW. We believe this is due to the number of words in the corpora, as BOW is unable to extract features that lead to better classification in ambiguous and large size datasets.

Table 3: Classification F1-Micro score for BOW approach against the proposed techniques for each classifier and dataset. Values in **bold** represent the best result of that row. Underline values represent the best value in that column.

	BOW	FL-0R	FL-1R	FL-2R	FX2-0R	FX2-1R	FX2-2R	FX4-0R	FX4-1R	FX4-2R
Ohsumed	0.3486	0.4402	0.4412	0.4389	0.4383	0.4416	0.4399	0.4326	0.4376	0.4316
K-NN	0.3187	0.3953	0.3967	0.3936	0.4007	**0.4023**	0.4012	0.3940	0.3993	0.3986
RF	0.3486	0.3407	0.3397	0.3380	0.3486	**0.3540**	0.3495	0.3462	0.3505	0.3483
LR	0.3441	0.4123	0.4169	0.4151	**0.4283**	0.4196	0.4182	0.4097	0.4137	0.4148
SVM	0.3364	0.4402	0.4412	0.4389	0.4383	**0.4416**	0.4399	0.4326	0.4376	0.4316
NB	0.1662	0.3091	0.3065	0.3079	0.3157	0.3218	**0.3224**	0.3097	0.3199	0.3158
20News	0.5458	0.7135	0.7147	0.7167	0.7151	0.7272	0.7196	0.7066	0.7168	0.7059
K-NN	0.4316	0.6312	0.6329	0.6298	0.6353	**0.6447**	0.6371	0.6333	0.6377	0.6321
RF	0.5458	0.6701	0.6742	0.6642	0.6706	**0.6802**	0.6792	0.6722	0.6706	0.6749
LR	0.5252	0.7067	0.7147	0.7131	0.7151	**0.7272**	0.7187	0.7066	0.7168	0.7059
SVM	0.5096	0.7135	0.7114	0.7167	0.7035	0.7192	**0.7196**	0.6994	0.7115	0.7038
NB	0.3946	0.5860	0.5884	0.5890	0.5821	0.5916	0.5839	0.5883	**0.5928**	0.5834
Reuters	0.8683	0.8719	0.8726	0.8719	0.8701	0.8708	0.8664	0.8672	0.8690	0.8672
K-NN	0.8378	**0.8558**	0.8554	0.8539	0.8489	0.8518	0.8493	0.8443	0.8468	0.8425
RF	0.8561	0.8543	0.8525	0.8504	0.8522	0.8550	0.8532	0.8492	**0.8558**	0.8540
LR	0.8683	0.8698	**0.8726**	0.8676	0.8672	0.8708	0.8637	0.8622	0.8644	0.8647
SVM	0.6232	**0.8719**	0.8640	**0.8719**	0.8701	0.8637	0.8664	0.8672	0.8690	0.8672
NB	0.7718	0.8116	**0.8120**	0.8044	0.7937	0.7980	0.7998	0.7973	0.8027	0.7987
BBC	0.9524	0.9784	0.9784	0.9771	0.9775	0.9757	0.9784	0.9771	0.9766	0.9771
K-NN	0.9097	0.9604	0.9600	0.9627	0.9627	0.9573	0.9627	0.9595	0.9623	**0.9650**
RF	0.9421	0.9645	0.9667	0.9636	0.9686	**0.9703**	0.9667	0.9667	0.9667	0.9649
LR	0.9524	**0.9784**	**0.9784**	0.9753	0.9757	0.9757	**0.9784**	0.9757	0.9766	0.9748
SVM	0.9510	**0.9780**	0.9771	0.9771	0.9775	0.9766	0.9775	0.9771	0.9753	0.9771
NB	0.9137	0.9474	0.9456	0.9465	0.9447	0.9469	**0.9501**	0.9402	0.9469	0.9461
ScyClusters	0.6997	0.6930	0.6822	0.6827	0.6990	0.6936	0.6991	0.6738	0.6723	0.6833
K-NN	**0.6470**	0.6430	0.6165	0.6340	0.6283	0.6400	0.6424	0.6146	0.6322	0.6216
RF	**0.6997**	0.5975	0.5940	0.5885	0.6077	0.6073	0.6065	0.6053	0.5968	0.6077
LR	0.6616	0.6845	0.6822	0.6827	**0.6990**	0.6936	0.6916	0.6670	0.6711	0.6833
SVM	0.6610	0.6930	0.6701	0.6815	0.6802	0.6905	**0.6991**	0.6738	0.6723	0.6678
NB	0.5329	0.5354	**0.5385**	0.5324	0.5233	0.5329	0.5348	0.5069	0.5154	0.5311
ScyGenes	0.9767	0.8474	0.8521	0.8457	0.8502	0.8456	0.8420	0.8394	0.8494	0.8411
K-NN	**0.9094**	0.7829	0.7750	0.7783	0.7856	0.7903	0.7731	0.7551	0.7758	0.7785
RF	**0.9767**	0.7727	0.7603	0.7597	0.7756	0.7659	0.7674	0.7612	0.7694	0.7711
LR	**0.9309**	0.8454	0.8521	0.8457	0.8502	0.8456	0.8420	0.8394	0.8494	0.8411
SVM	**0.9282**	0.8474	0.8475	0.8450	0.8494	0.8402	0.8413	0.8279	0.8402	0.8267
NB	**0.9229**	0.6858	0.6895	0.6967	0.7021	0.7047	0.7048	0.6894	0.7083	0.7001

Except for the ScyClusters and ScyGenes datasets, our algorithms seem to be stable within their different variations for the same classifiers. Even though BOW achieves the best results for ScyClusters and ScyGenes, our techniques are still comparable. However, FLLC II and FXLC II do not require dedicated training in these datasets. Specifically for ScyClusters, our F1-Micro for FX2-2R deviates less than 1% from BOW. If we consider each row in Table 3, FLLC II and FXLC II together present a better score in 23 out of 30 cases. We can observe that FXLC with a chunk size of 2 and FLLC II built over MSSA-1R had the majority of best results (FX2-1R and FL-1R). Thus, we used FX2-1R and FL-1R to compare against the other state-of-the-art models.

In Table 4, we present the best results achieved by our Chains2Vec embedding model against state-of-the-art techniques using word embeddings. Considering the Ohsumed dataset, our approaches outperform all the other methods for all classifier baselines, which have a considerably inferior result. The Ohsumed text collection represents a difficult real-world scenario because of its size, the number of classes, and constituent words. In this dataset, our fixed lexical chains of size 2 (FX2-1R) achieves the best results for all classifiers. For the 20Newsgroups dataset, our techniques overcome the baselines for the K-NN, RF and NB, while fastText was superior considering LR and SVM.

Considering the Reuters-21578 dataset, GloVe achieves the best results in 4 out of 5 classifiers. If we think how GloVe uses a co-occurrence approach to build its vectors, this outcome is expected. Several documents in this dataset are composed of short phrases, which prevents our techniques from deriving a good semantic representation. As a consequence, our chains performed poorly. However, even affected by Reuters' content, we outperform methods able to embed any given token, such as ELMo and USE. On BBC, the synset embedding models present the highest F1-Micro score in 4 of the 5 classifiers, in which our lexical chain representation has the best scores for RF and LR. The documents in the BBC dataset are composed of BBC News articles, which use a formal and cohesive English. A coherent text structure contributes to the semantic representation in our techniques and M1R since they are based on WordNet.

Table 4: Classification F1-Micro score for word embeddings models against proposed techniques for each classifier and dataset. Values in **bold** represent the best result of that row. Underline values represent the best value in that column.

	LDA	w2v	GloVe	fastText	USE	ELMo	MSSA-1R	FL-1R	FX2-1R
Ohsumed	0.2262	0.4223	0.4136	0.4324	0.3009	0.4172	0.4357	0.4412	0.4416
K-NN	0.2138	0.3822	0.3731	0.3912	0.3009	0.3209	0.3975	0.3961	**0.4023**
RF	0.1689	0.3302	0.3258	0.3447	0.2899	0.2851	0.3411	0.3397	**0.3540**
LR	0.2262	0.3981	0.4136	0.4171	0.2792	0.4172	0.4185	0.4169	**0.4196**
SVM	0.2056	0.4223	0.3286	0.4324	0.2542	0.3194	0.4357	0.4412	**0.4416**
NB	0.0558	0.2820	0.2771	0.2785	0.1947	0.1865	0.3118	0.3065	**0.3218**
20News	0.6340	0.7110	0.7153	0.7485	0.6476	0.6895	0.7201	0.7147	0.7272
K-NN	0.5013	0.5737	0.5425	0.6134	0.5588	0.4574	0.6320	0.6329	**0.6447**
RF	0.6340	0.6300	0.6386	0.6794	0.6476	0.5389	0.6753	0.6742	**0.6802**
LR	0.5498	0.6657	0.7152	**0.7288**	0.5447	0.6895	0.7167	0.7147	0.7272
SVM	0.5388	0.7110	0.7153	**0.7485**	0.5171	0.4981	0.7201	0.7114	0.7192
NB	0.4624	0.4426	0.4133	0.5104	0.4401	0.2677	0.5895	0.5884	**0.5916**
Reuters	0.8260	0.8805	0.8830	0.8802	0.8278	0.8780	0.8719	0.8726	0.8708
K-NN	0.7919	**0.8680**	0.8640	0.8593	0.8267	0.8436	0.8568	0.8554	0.8518
RF	0.8260	0.8561	**0.8619**	0.8536	0.8278	0.8281	0.8550	0.8525	0.8550
LR	0.7875	0.8698	**0.8776**	0.8705	0.7370	0.8751	0.8651	0.8726	0.8708
SVM	0.8041	0.8805	**0.8830**	0.8802	0.7951	0.8780	0.8719	0.8640	0.8637
NB	0.6024	0.7740	**0.8224**	0.8034	0.7725	0.7603	0.8009	0.8120	0.7980
BBC	0.9552	0.9708	0.9784	0.9766	0.9672	0.9743	0.9780	0.9784	0.9766
K-NN	0.9304	0.9591	**0.9622**	0.9618	0.9577	0.9497	0.9596	0.9600	0.9573
RF	0.9552	0.9532	0.9631	0.9663	0.9672	0.9573	0.9681	0.9667	**0.9703**
LR	0.9241	0.9478	0.9690	0.9681	0.9370	0.9649	0.9766	**0.9784**	0.9757
SVM	0.9295	0.9708	**0.9784**	0.9766	0.9474	0.9743	0.9780	0.9771	0.9766
NB	0.8680	0.9218	0.9483	0.9469	0.9442	0.9195	**0.9501**	0.9456	0.9469
ScyClusters	0.4814	0.6410	0.6391	0.6645	0.4966	0.6612	0.7027	0.6822	0.6936
K-NN	0.4137	0.5919	0.5903	0.5777	0.4835	0.5675	0.6267	0.6165	**0.6400**
RF	0.4814	0.5479	0.5469	0.5890	0.4966	0.5317	0.5889	0.5940	**0.6073**
LR	0.3692	0.5879	0.6168	0.6162	0.3475	0.6612	0.6839	0.6822	**0.6936**
SVM	0.3439	0.6410	0.6391	0.6645	0.3439	0.3650	**0.7027**	0.6701	0.6905
NB	0.2362	0.4769	0.5185	0.4790	0.3910	0.4670	0.5373	**0.5385**	0.5329
ScyGenes	0.6104	0.7988	0.7961	0.8301	0.6460	0.8761	0.8556	0.8521	0.8456
K-NN	0.4573	0.7085	0.7120	0.7281	0.6105	0.7480	0.7866	0.7750	**0.7903**
RF	0.6104	0.6849	0.7363	0.7308	0.6460	0.7423	0.7649	0.7603	**0.7659**
LR	0.3488	0.6346	0.7746	0.7477	0.3044	0.8509	**0.8556**	0.8521	0.8456
SVM	0.3471	0.7988	0.7961	0.8301	0.2442	**0.8761**	0.8538	0.8475	0.8402
NB	0.3766	0.6326	0.6743	0.6249	0.5725	0.7028	0.6977	0.6895	**0.7047**

When we compare the results of word2vec, GloVe, and fastText with USE and ELMo we see the former group outperforms the latter in almost all cases in their best configuration (except for GloVe and ELMo in Ohsumed). We consider two reasons for this behavior, the difference in their base training corpus and the lack of fine-tuning in the neural network inspired models. While GloVe and fastText mainly use Wikipedia dumps as their training corpus, ELMo and USE are trained in the 1 Billion Word Benchmark and various sources, as described in Table 2. All pre-trained word embeddings models in our experiments were not fine-tuned since the main goal was to compare their behavior when applied directly to our task. As a result, ELMo and USE showed a significant disadvantage when compared to the other models since they were using the frozen weights from their general base training tasks. In the future, we plan to explore the benefits of fine-tuning neural network models (e.g., Bi-LSTM, Transformers) towards a specific task by comparing them with our techniques and incorporating the most prominent ones to the best performing neural architectures.

Finally, on ScyClusters and ScyGenes datasets, our lexical chains show superior results in 7 out of 10 experiments. If we consider word2vec and FX2-1R, for the K-NN classifier, our technique shows an improvement of 15% and 9% for the ScyClusters and ScyGenes datasets respectively. These results suggest that the semantic relations extracted by our algorithms improve the quality of a standard word embeddings. As in BBC and Oshumed, ScyClusters and ScyGenes contain formal, cohesive, and typo-free English documents. In an overall analysis, we sustain good results throughout the explored datasets. The synset techniques, especially the proposed ones, are able to extract semantic relations within the documents properly. Besides, the generated synset embeddings models are also significantly smaller (75%) than the ones compared in this paper.

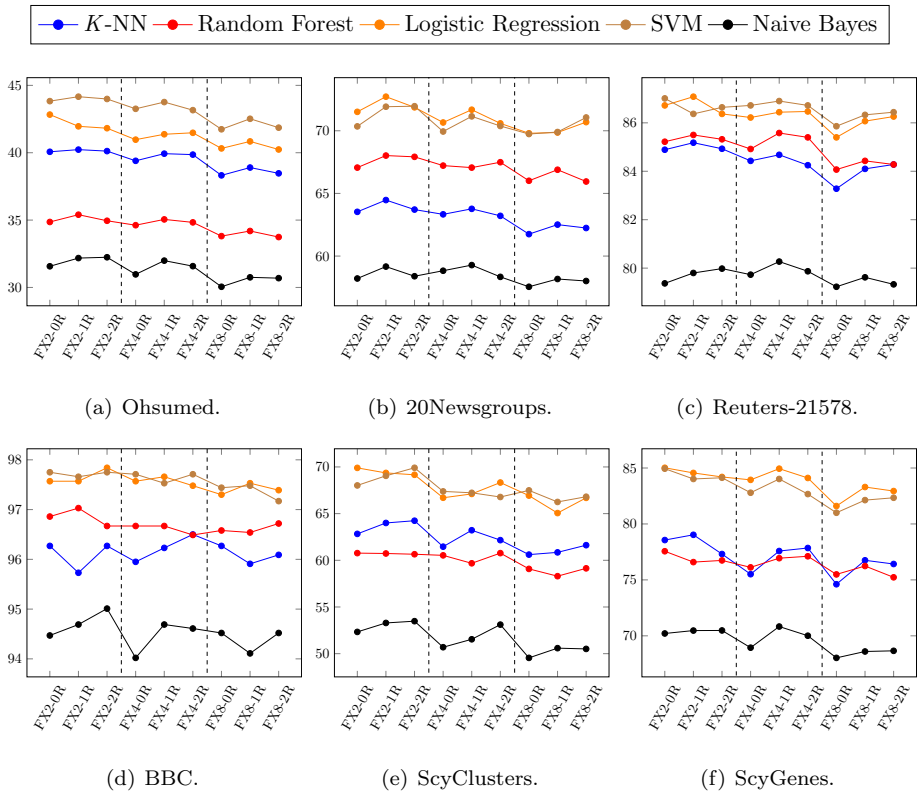
4.6. Lexical chains behavior analysis

In this section, we investigate how our proposed techniques are affected by their internal configurations. We still keep the same machine learning classifiers and datasets to maintain consistency in our comparisons. The main idea is

to provide a different perspective on how the fixed chains are affected by the chosen chunk size and the pre-trained models in which they are based (MSSA-1R and MSSA-2R). The results in Figure 4 show the F1-Micro score over different variations of our approaches, for six datasets and five classifiers.

The idea behind both techniques (FLLC and FXLC) is to provide a more concise and robust document representation for text documents. Techniques such as word2vec and GloVe have two significant limitations: (i) they represent all word senses (e.g., *java-coffee*, *java-island*, *java-programming-language*) in a single vector and (ii) ignore the relationship between words; both of which our proposed techniques do. While MSSA solves the former (i), FLLC and FXLC focus on the latter (ii). MSSA fixes the multi-sense representation, but the relationship between word works at a local level. On the other hand, FLLC and FXLC work on the relationship between consecutive words of the entire document at a multi-sense level. Regardless of how many synsets each chain in FLLC and FXLC has, we represent them with just one. Thus, reducing the necessary features for the document classification task and still keeping all benefits of the word embeddings algorithms in which they were trained.

For FXLC II in Figure 4, all models built on top of MSSA-1R (1R) present an improvement if compared with their base version (0R). However, this improvement is not valid for the -2R variant. Our fixed chains seem to be more affected by the change in their chunk-size than the pre-trained model used (-1R and -2R). If we consider each block separately (vertical dashed lines), the F1-Micro score for each fixed chain size remains quite stable. Nevertheless, moving from 2 to 8 synsets per chain affects our technique significantly. In other words, the chunk size parameter in our algorithm is inversely proportional to the quality of semantic representation in our chains. Thus, resulting in a decrease in the score when we move from 2 to 8 synsets per chain. For a chunk size of 1, FXLC is reduced to MSSA, since each word-sense is its own chain. Since the lexical chains produced with the FLLC II algorithm present the same behavior as in the FXLC, concerning the pre-trained MSSA model used, but on a much smaller variation scale, we decided not to include its graphical analysis.



(a) Ohsumed. (b) 20Newsgroups. (c) Reuters-21578.

(d) BBC. (e) ScyClusters. (f) ScyGenes.

Figure 4: F1-Micro score for FXLC II with variable chunk size and pre-trained MSSA models.

5. Further discussions and limitations

In this section, we provide a more in-depth discussion about the main aspects of our techniques, point out their strengths, and discuss possible alternatives to mitigate some of their limitations.

The main objective of our proposed algorithms is to combine the semantic relations of the lexical chains to provide lightweight pre-trained models. These models generate synset-vectors suited to improve the predictive abilities of classifier algorithms. We represent our lexical chains through synset-tokens. For the FLLC II technique (Section 3.1.1), we use WordNet to identify how the synsets in our documents are connected. As a result, we remove tokens not mapped in WordNet [10], which might not reflect the true semantic value in the

document. Other lexical databases, such as ConceptNet [23] and BabelNet [33], might provide alternative structures for the synsets. However, ConceptNet does not structure its components in sets of synonyms, which would require drastic changes in our algorithm. On the other hand, BabelNet uses a similar synset structure to WordNet that could be explored by our algorithm. BabelNet integrates different resources[21] (including WordNet). Unfortunately, because of their proprietary license, BabelNet's access is not as facilitated as ConceptNet and WordNet. It is important to mention that, for research purposes, BabelNet indices are available upon an application request to their company. The application requires affiliation with a research institution or a Ph.D. student status, besides the non-commercial nature of the project.

One might point out that the proposed techniques only use a word2vec implementation to embed the synset corpora produced by FLLC II and FXLC II. Although this brings an interesting perspective, we decided to validate our algorithms using a straightforward method before moving to more complex ones, such as fastText [4], ELMo [38], and USE [5]. In fastText, they propose to learn word representations as a sum of the n-grams of its constituent sub-words. Sub-words would incorporate a high number of tokens that do not exist in WordNet. Thus, our approaches would not take advantage of these extra computations. In ELMo, the sub-word issue is even stronger since their words vectors are a linear combination of their characters. We also examined the recently published USE [5], but their implementation only allows us to access and retrieve word vectors from their pre-calculated model, not train a new corpus. Another factor that prevents us from using ELMo[22], and any Transfomer-inspired architectures (e.g., BERT) for now, is their expensive training and fine-tuning processes.

The synsets in our chains and embeddings models are built using proper English. For that reason, our approach does not generalize that well for documents containing informal English (e.g., jargon). A way to mitigate the lack of matches

[21] `https://babelnet.org/about`

[22] `https://github.com/allenai/bilm-tf`

between document tokens and the embeddings model would be to incorporate multiple pre-trained word embeddings, similar to what Sinoara et al. [46] adopt. However, this can lead to an overhead as significant as the pre-trained models considered. Additionally, if several word embeddings models have the same word, a ranking system would need to be used as well.

Even though we incorporate as much as 19 semantic objects[3] in WordNet for the FLLC II algorithm 3.1.1, there are other artifacts in WordNet. Nevertheless, during the early stages of our research, we discovered that many of synset's attributes (*related_synsets* in Algorithm 1) would rarely return any synset. By decreasing the number of related synsets, we could gain more performance during the construction of our flexible chains. In addition to the number of available attributes, we did not explore deeper levels of relations for each object (e.g., hypernyms of hypernyms). In other words, for each synset in the related synsets, we did not investigate their related synsets.

During our experiments, we noticed that datasets composed of paper abstracts had the best performance. These results reinforce the strength of our technique on documents in well-written English. Even so, we still had excellent results on other datasets of other nature (e.g., news). Especially, with 20Newsgroups we had 3 out of 5 best results. During our investigations on the experiments against BOW, we could understand that our performance tends to decrease when we use chains with large chunks sizes. A possible explanation may be because with larger chain chunks, we have a stronger dimensionality reduction, and consequently, we might lose too much information in the process.

6. Final considerations

In this work, we proposed two techniques which combine the areas of lexical chains and word embeddings to generate a lightweight pre-trained model capable of improving the document classification. FLLC II and FXLC II help us to extract implicit semantic relations between words in a synset-based text document using different methods. For FLLC II, we build our lexical chains

with the assistance of a lexical database to extract the relations among the constituent synsets of a document. Additionally, FLLC II can handle any POS and considers 19 synset objects[3] in WordNet. In FXLC II, we pre-define a specific number of synsets for each chain. In both methods, FLLC II and FXLC II, we produce a more concise and robust semantic representation than word-based (e.g., BOW) techniques and traditional word embeddings (e.g., word2vec).

Our experiments compared the traditional BOW approach and seven other state-of-the-art techniques: LDA, word2vec, GloVe, fastText, USE, ELMo, and MSSA. To explore the stability of our systems, we evaluated all methods on six distinct datasets with specific characteristics that impose a particular challenge on each classification. Furthermore, we considered five machine learning classifiers in our experiments so we can guarantee our findings are not bound to one specific classification technique. Our results showed the proposed methods for building lexical chains leverage the semantic representation offered in previous contributions. As Table 4 shows, FLLC II and FXLC II often improve the results of MSSA and traditional word2vec, which are their building blocks. Therefore, we believe the proposed techniques applicable to other NLP downstream tasks that also require a refined semantic representation, such as sentiment analysis, text summarization, and plagiarism detection. To facilitate an extension of the research on the document classification task, the data and code of our study are openly available[1].

An exciting aspect of our architecture (Figure 1) is that FLLC II and FXLC II use synset embeddings models to decide how to represent their lexical chains. Once a synset embeddings model is available, using the results from FLLC II and FXLC II as a corpus, we can feed it back to our algorithms and create a more refined output corpus (using the original input one). This process can be done recurrently over many iterations, the same way MSSA-1R and MSSA-2R were generated in [43]. We believe after many passes, the representation of our chains will get more accurate, better defined, and stable.

Considering FLLC II and FXLC II outputs have an identical format for their token representation (synset-annotated corpus), we can use them recurrently.

Running the FLLC II algorithm iteratively will not affect its chain structure because once the flexible chains are represented, there is no lexical relationship that connects two separate chains. Otherwise, they would be placed together in the first place. However, for the FXLC II algorithm, if we use the output chain synset-annotated corpus as an input, the size of our corpus would decrease according to chunk size (cs). By doing so, FXLC II semantic representation would provide less meaningful chains.

In our techniques, we chose to represent each chain using the closest synset to their centroid, so we could still relate it to essential components in the lexical database. Therefore, less dominant synset are often not chosen as chain representatives. A clear direction would be to represent our chain as a direct vector-average of its constituents synsets (at the cost of some interpretability). Another alternative would be to use the average vector of the chain and look for the most similar(s) synsets in a pre-trained model. We leave the investigation of this and other variations of our algorithms for future work.

In this work, we chose to embed our lexical chains using a word2vec implementation. However, as explained in Section 4, recently published embeddings techniques (e.g., ELMo, USE) bring new directions for our work. We believe our proposed approach of combining lexical chains and embeddings algorithms can leverage the semantic features in these neural network models. Moreover, we also intend to investigate how Transformer-based architectures [47] such as BERT [8] and XLNet [48] can be applied to FLLC and FXLC. However, these architectures will require additional effort in their implementation since their fine-tuning, training process, and hardware are more restrictive than traditional word embeddings techniques (e.g., word2vec, GloVe).

7. Acknowledgments

This work was partially supported by the Science Without Borders Brazilian Government Scholarship Program, CNPq [grant number 205581/2014-5]; Charles Henrique Porto Ferreira was supported by Coordenação de Aperfeiçoamento

de Pessoal de Nível Superior - Brasil (Capes), Programa de Doutorado Sanduíche no Exterior (PDSE), [grant number 88881.186965/2018-01].

8. References

[1] Bär, D., Zesch, T., & Gurevych, I. (2015). Composing measures for computing text similarity. URL: http://tuprints.ulb.tu-darmstadt.de/4342/.

[2] Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

[3] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, *3*, 993–1022. URL: http://dl.acm.org/citation.cfm?id=944919.944937.

[4] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

[5] Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., & Kurzweil, R. (2018). Universal sentence encoder. *CoRR*, *abs/1803.11175*. URL: http://arxiv.org/abs/1803.11175. arXiv:1803.11175.

[6] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, . URL: http://dx.doi.org/10.18653/v1/D17-1070. doi:10.18653/v1/d17-1070.

[7] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, *7*, 1–30. URL: http://dl.acm.org/citation.cfm?id=1248547.1248548.

[8] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

[9] Enríquez, F., Troyano, J. A., & López-Solaz, T. (2016). An approach to the use of word embeddings in an opinion classification task. *Expert Systems with Applications*, *66*, 1 – 6. doi:https://doi.org/10.1016/j.eswa.2016.09.005.

[10] Fellbaum, C. (Ed.) (1998). *WordNet: an electronic lexical database*. MIT Press.

[11] Ferreira, C. H. P., de França, F. O., & de Medeiros, D. M. R. (2018). Combining multiple views from a distance based feature extraction for text classification. In *2018 IEEE Congress on Evolutionary Computation, CEC 2018, Rio de Janeiro, Brazil, July 8-13, 2018* (pp. 1–8). URL: https://doi.org/10.1109/CEC.2018.8477772. doi:10.1109/CEC.2018.8477772.

[12] Fu, M., Qu, H., Huang, L., & Lu, L. (2018). Bag of meta-words: A novel method to represent document for the sentiment classification. *Expert Systems with Applications*, *113*, 33 – 43. doi:https://doi.org/10.1016/j.eswa.2018.06.052.

[13] Gonzales, A. R., Mascarell, L., & Sennrich, R. (2017). Improving word sense disambiguation in neural machine translation with sense embeddings. In O. Bojar, C. Buck, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno-Yepes, P. Koehn, & J. Kreutzer (Eds.), *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017* (pp. 11–19). Association for Computational Linguistics. URL: https://aclanthology.info/papers/W17-4702/w17-4702.

[14] Greene, D., & Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd Inter-*

national Conference on Machine learning (ICML'06) (pp. 377–384). ACM Press.

[15] Guo, W., & Diab, M. (2011). Semantic topic models: Combining word distributional statistics and dictionary definitions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (pp. 552–561). Edinburgh, Scotland, UK.: Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/D11-1051.

[16] Harris, Z. (1954). Distributional structure. *Word*, *10*, 146–162.

[17] Heydari, A., Tavakoli, M. a., Salim, N., & Heydari, Z. (2015). Detection of review spam. *Expert Syst. Appl.*, *42*, 3634–3642. URL: https://doi.org/10.1016/j.eswa.2014.12.029. doi:10.1016/j.eswa.2014.12.029.

[18] Iacobacci, I., Pilehvar, M. T., & Navigli, R. (2016). Embeddings for word sense disambiguation: An evaluation study. In *ACL (1)*. The Association for Computer Linguistics.

[19] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning* ECML'98 (pp. 137–142). Berlin, Heidelberg: Springer-Verlag. URL: https://doi.org/10.1007/BFb0026683. doi:10.1007/BFb0026683.

[20] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, .

[21] Kamkarhaghighi, M., & Makrehchi, M. (2017). Content tree word embedding for document representation. *Expert Systems with Applications*, *90*, 241 – 249. doi:https://doi.org/10.1016/j.eswa.2017.08.021.

[22] Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. arXiv:1405.4053.

[23] Liu, H., & Singh, P. (2004). Conceptnet — a practical commonsense reasoning tool-kit. *BT Technology Journal*, *22*, 211–226. URL: http://dx.doi.org/10.1023/B:BTTJ.0000047600.45421.6d. doi:10.1023/B:BTTJ.0000047600.45421.6d.

[24] Liu, L., Lu, Y., Yang, M., Qu, Q., Zhu, J., & Li, H. (2017). Generative adversarial network for abstractive text summarization. arXiv:1711.09357.

[25] Mancini, M., Camacho-Collados, J., Iacobacci, I., & Navigli, R. (2017). Embedding words and senses together via joint knowledge-enhanced training. In *CoNLL* (pp. 100–111). Association for Computational Linguistics.

[26] Marco Baroni, Georgiana Dinu, G. K. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, *1*, 238–247. URL: https://www.researchgate.net/publication/270877599_Don%27t_count_predict_A_systematic_comparison_of_context-counting_vs_context-predicting_semantic_vectors.

[27] Mascarell, L. (2017). Lexical chains meet word embeddings in document-level statistical machine translation. In B. L. Webber, A. Popescu-Belis, & J. Tiedemann (Eds.), *Proceedings of the Third Workshop on Discourse in Machine Translation, DiscoMT@EMNLP 2017, Copenhagen, Denmark, September 8, 2017* (pp. 99–109). Association for Computational Linguistics. URL: https://aclanthology.info/papers/W17-4813/w17-4813.

[28] Medeiros, D. M. R., & Carvalho, A. C. P. L. F. (2005). Applying text mining and machine learning techniques to gene clusters analysis. In *ICCIMA '05: Proceedings of the Sixth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'05)* (pp. 23–28). Washington, DC, USA: IEEE Computer Society.

[29] Meng, L., Huang, R., & Gu, J. (2013). A Review of Semantic Similar-

ity Measures in WordNet. *International Journal of Hybrid Information Technology*, *6*, 1–12.

[30] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*. URL: http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781.

[31] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* NIPS'13 (pp. 3111–3119). USA: Curran Associates Inc. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

[32] Morris, J., & Hirst, G. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, *17*, 21–48. URL: http://dl.acm.org/citation.cfm?id=971740.

[33] Navigli, R., & Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, *193*, 217–250.

[34] Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, *22*, 1345–1359. URL: http://dx.doi.org/10.1109/TKDE.2009.191. doi:10.1109/TKDE.2009.191.

[35] Pedersen, T., Patwardhan, S., & Michelizzi, J. (2004). Wordnet::similarity: Measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004* HLT-NAACL–Demonstrations '04 (pp. 38–41). Stroudsburg, PA, USA: Association for Computational Linguistics. URL: http://dl.acm.org/citation.cfm?id=1614025.1614037.

[36] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Van-

derplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

[37] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP* (pp. 1532–1543). volume 14.

[38] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 2227–2237). Association for Computational Linguistics. URL: `http://aclweb.org/anthology/N18-1202`. doi:`10.18653/v1/N18-1202`.

[39] Pilehvar, M. T., & Collier, N. (2016). De-conflated semantic representations. In *EMNLP* (pp. 1680–1690). The Association for Computational Linguistics.

[40] Řehůřek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). Valletta, Malta: ELRA. `http://is.muni.cz/publication/884893/en`.

[41] Ruas, T., & Grosky, W. (2017). Keyword Extraction Through Contextual Semantic Analysis of Documents. In *Proceedings of the 9th International Conference on Management of Emergent Digital EcoSystems* (pp. 150–156). Bangkok: ACM Press.

[42] Ruas, T., & Grosky, W. (2018). Semantic Feature Structure Extraction from Documents Based on Extended Lexical Chains. In *Proceedings of the 9th Global Wordnet Conference*. Nanyang Technological University (NTU), Singapore: Association for Computational Linguistics.

[43] Ruas, T., Grosky, W., & Aizawa, A. (2019). Multi-sense embeddings through a word sense disambiguation process. *Expert Systems With Applications*, *136*, 288–303. URL: https://www.sciencedirect.com/science/article/pii/S0957417419304269?via%3Dihub. doi:doi.org/10.1016/j.eswa.2019.06.026.

[44] Shaoul, C., & Westbury, C. (2010). The westbury lab wikipedia corpus. URL: http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html.

[45] Simov, K. I., Boytcheva, S., & Osenova, P. (2017). Towards lexical chains for knowledge-graph-based word embeddings. In *RANLP* (pp. 679–685). INCOMA Ltd.

[46] Sinoara, R. A., Camacho-Collados, J., Rossi, R. G., Navigli, R., & Rezende, S. O. (2019). Knowledge-enhanced document embeddings for text classification. *Knowledge-Based Systems*, *163*, 955 – 971. URL: http://www.sciencedirect.com/science/article/pii/S0950705118305124. doi:https://doi.org/10.1016/j.knosys.2018.10.026.

[47] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., & Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* NIPS17 (p. 60006010). Red Hook, NY, USA: Curran Associates Inc.

[48] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*. arXiv:1906.08237v1.

[49] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28* (pp. 649–657).

Curran Associates, Inc. URL: http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf.

[50] Zheng, R., Li, J., Chen, H., & Huang, Z. (2006). A framework for authorship identification of online messages: Writing-style features and classification techniques. *J. Am. Soc. Inf. Sci. Technol.*, *57*, 378–393. URL: http://dx.doi.org/10.1002/asi.v57:3. doi:10.1002/asi.v57:3.