IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 7, 2019, accepted August 25, 2019, date of publication August 28, 2019, date of current version September 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938058

ShotgunWSD 2.0: An Improved Algorithm for Global Word Sense Disambiguation

ANDREI M. BUTNARU AND RADU TUDOR IONESCU, (Member, IEEE)
Faculty of Mathematics and Computer Science, University of Bucharest, 010014 Bucharest, Romania

Corresponding author: Radu Tudor Ionescu (raducu.ionescu@gmail.com)

The work of R. T. Ionescu was supported by the Ministry of Research and Innovation, CNCS-UEFISCDI, within PNCDI III, under Project PN-III-P1-1.1-PD-2016-0787.

ABSTRACT ShotgunWSD is a recent unsupervised and knowledge-based algorithm for global word sense disambiguation (WSD). The algorithm is inspired by the Shotgun sequencing technique, which is a broadly-used whole genome sequencing approach. ShotgunWSD performs WSD at the document level based on three phases. The first phase consists of applying a brute-force WSD algorithm on short context windows selected from the document, in order to generate a short list of likely sense configurations for each window. The second phase consists of assembling the local sense configurations into longer composite configurations by prefix and suffix matching. In the third phase, the resulting configurations are ranked by their length, and the sense of each word is chosen based on a majority voting scheme that considers only the top configurations in which the respective word appears. In this paper, we present an improved version (2.0) of ShotgunWSD which is based on a different approach for computing the relatedness score between two word senses, a step that stays at the core of building better local sense configurations. For each sense, we collect all the words from the corresponding WordNet synset, gloss and related synsets, into a *sense bag*. We embed the collected words from all the sense bags in the entire document into a vector space using a common word embedding framework. The word vectors are then clustered using k-means to form clusters of semantically related words. At this stage, we consider that clusters with fewer samples (with respect to a given threshold) represent outliers and we eliminate these clusters altogether. Words from the eliminated clusters are also removed from each and every sense bag. Finally, we compute the median of all the remaining word embeddings in a given sense bag to obtain a sense embedding for the corresponding word sense. We compare the improved ShotgunWSD algorithm (version 2.0) with its previous version (1.0) as well as several state-of-the-art unsupervised WSD algorithms on six benchmarks: SemEval 2007, Senseval-2, Senseval-3, SemEval 2013, SemEval 2015, and overall (unified). We demonstrate that ShotgunWSD 2.0 yields better performance than ShotgunWSD 1.0 and some other recent unsupervised or knowledge-based approaches. We also performed paired McNemar's significance tests, showing that the improvements of ShotgunWSD 2.0 over ShotgunWSD 1.0 are in most cases statistically significant, with a confidence interval of 0.01.

INDEX TERMS Word sense disambiguation, shotgun sequencing, word embeddings, outlier removal.

I. INTRODUCTION

Word Sense Disambiguation (WSD) is a core problem studied in the Natural Language Processing (NLP) community. WSD refers to the task of identifying which sense of a word is used in a given context. It has the potential to improve many NLP applications such as machine translation [1]–[3], text summarization [4], information retrieval [5] or sentiment analysis [6]. Most of the existing WSD algorithms [7], [8] are usually divided into supervised, unsupervised, and knowledge-based techniques. Nonetheless, hybrid methods, e.g. unsupervised and knowledge-based, have also been proposed in the literature [9]. Among these, supervised methods have reached the best disambiguation results [10], [11], but their main disadvantage is that they need large amounts of labeled examples for the supervised learning stage.

The associate editor coordinating the review of this article and approving it for publication was Laxmisha Rai.

Since large annotated corpora are difficult to obtain [12], [13], many researchers have turned their focus on developing unsupervised learning or knowledge-based WSD methods [14]–[24].

In this paper, we present an improved version of a recently introduced WSD algorithm [25], termed ShotgunWSD,[1] which stems from the Shotgun genome sequencing technique [26], [27]. ShotgunWSD is unsupervised, but it also requires knowledge in the form of WordNet synsets and relations [28], [29]. More precisely, for each sense of a word, ShotgunWSD builds a *disambiguation vocabulary* (or *sense bag*) that is an unordered list of words collected from the corresponding WordNet synset, gloss and related synsets which are chosen depending on the part-of-speech of the ambiguous word. To this end, ShotgunWSD can be viewed as a hybrid (unsupervised and knowledge-based) approach.

In general, WSD algorithms can be divided into approaches that work at the local level and approaches that work at the global level. A local WSD approach, such as the extended Lesk measure [30]–[32], is designed to assign the corresponding sense for a target word in a given context window of a few words. The corresponding sense is usually selected from an existing sense inventory. For instance, for the word ''sense'' in the context ''You have a good sense of humor.'', a local WSD algorithm should choose the sense that corresponds to the *natural ability* rather than the *meaning of a word* or the *sensation*. Rather more generally, a global WSD algorithm aims at choosing the right sense for each ambiguous word in an entire text document. The obvious solution for global WSD is the exhaustive evaluation of all sense combinations (configurations) [33], but the time complexity grows exponentially along with the number of words in the document, as also noted by Schwab *et al.* [14], [15]. For example, in the sentence ''You have a good sense of humor.'', we have four ambiguous WordNet [28] entries (considering that the part-of-speech of each word is already known): ''have'' (with 19 senses as verb), ''good'' (with 21 senses as adjective), ''sense'' (with 5 senses as noun) and ''humor'' (with 6 senses as noun). Consequently, there are $19 \times 21 \times 5 \times 6 = 11970$ possible sense configurations. However, if we extend the sentence to ''I think that you have a good sense of humor.'', we have two more ambiguous words: ''I'' (with 3 senses as noun) and ''think'' (with 13 senses as verb). In this case, the number of possible configurations grows to $3 \times 13 \times 19 \times 21 \times 5 \times 6 = 466830$. This example reveals that the brute-force (BF) solution quickly becomes impractical for windows of more than a few words. Therefore, several approximation approaches [14], [15] have been proposed for the global WSD task in order to overcome the exponential growth of the search space. ShotgunWSD is conceived to perform global WSD by combining multiple local sense configurations that are obtained using BF search, thus avoiding BF search on the whole text document. It employs a local WSD algorithm to build the local sense configurations. Butnaru *et al.* [25] alternatively used two methods for this step, namely the extended Lesk measure [31], [32] and an approach based on deriving sense embeddings from word embeddings [34]–[36].

In this paper, we propose a third approach which leads to an improved algorithm termed ShotgunWSD 2.0. Our approach starts by embedding the words collected from all the sense bags in the entire document into a vector space using a common word embedding framework [36]. The word vectors are then clustered using k-means to form clusters of semantically related words. At this stage, we consider that clusters with fewer samples (with respect to a given threshold) belong to outlier (unlikely) senses and we eliminate these clusters from the subsequent steps. Words from the eliminated clusters are also removed from each and every sense bag. Finally, we compute the median of all the remaining word embeddings in a given sense bag to obtain a sense embedding for the corresponding word sense. Hence, the derived sense embedding will not take into account the outlier words and will reduce the chance of selecting unlikely word senses. The main difference between ShotgunWSD 2.0 and its previous version presented by Butnaru *et al.* [25] consists of applying k-means clustering and eliminating smaller (outlier) clusters of semantically related words.

In summary, ShotgunWSD is comprised of three main phases. In the first phase, context windows of fixed length are selected from the document, and for each context window, the top scoring sense configurations constructed by BF search are kept for the upcoming phase. The second phase consists of merging the retained sense configurations based on prefix and suffix matching. Finally, the third phase consists of ranking the configurations obtained this far by their length (the longer, the better), and choosing the sense of each word through a majority vote on a short list of top configurations that cover the respective word.

We present experiments on SemEval 2007 [37], Senseval-2 [38], Senseval-3 [39], SemEval 2013 [40], SemEval 2015 [41], as well as on the unified data sets [11], in order to compare ShotgunWSD 2.0 with its previous version [25], other state-of-the-art unsupervised and knowledge-based approaches [15], [18]–[23], [42]–[45], as well as the Most Common Sense (MCS) baseline.[2] MCS is considered as one of the strongest baselines in WSD [7], surpassing all unsupervised approaches in the recent SemEval 2015 [41] WSD task. The empirical results show that our algorithm compares favorably to these state-of-the-art approaches on most data sets. Furthermore, our algorithm outperforms the Most Common Sense baseline on two data sets.

Our contributions are twofold:

- We propose a new relatedness scoring function for ShotgunWSD, which is based on a novel approach for eliminating outlier word senses by applying k-means

[1]The open source implementation of ShotgunWSD is provided for download at https://github.com/butnaruandrei/ShotgunWSD.

[2]Also known as the Most Frequent Sense (MFS) baseline.

clustering on the word embeddings extracted from a document.
- We perform extensive experiments on six data sets, surpassing the MCS baseline on two of them.

We organize the rest of this paper as follows. We present related work on unsupervised and knowledge-based WSD algorithms in Section II. We describe the ShotgunWSD algorithm in Section III. We present the experiments in Section IV. Finally, we draw our conclusions in Section V.

II. RELATED WORK

Researchers have proposed a wide range of methods to perform WSD [7], [8], [46]. The most accurate techniques are supervised [10], but they require annotated training corpora which are very difficult to obtain [12], [13]. In order to overcome this limitation, some researchers have proposed alternative WSD methods based on unsupervised learning or knowledgde bases [14]–[24], [31], [32], [42], [47]–[50]. Since our algorithm is unsupervised and based on the WordNet knowledge base [28], [29], our main focus is to present related work in the same area. The Lesk algorithm [30] is perhaps the most popular knowledge-based WSD method. It compares the glosses of an ambiguous word with the terms contained in a short context window, calculating an overlap score for each possible sense in order to select the likely word sense. Banerjee and Pedersen [31] extended the gloss overlap algorithm of Lesk [30] by using WordNet relations such as hypernymy or hyponymy, along with WordNet glosses and examples. Patwardhan *et al.* [33] proposed a brute-force algorithm for global WSD by employing the extended Lesk measure [31], [32] to compute the semantic relatedness among senses in a given text. As discussed in Section I, their BF approach is not suitable for whole text documents, because of the high computational time. More recently, Schwab *et al.* [14] proposed and compared three stochastic algorithms for global WSD as alternatives to BF search, namely a Genetic Algorithm, Simulated Annealing, and Ant Colony Optimization. Among these algorithms, the authors have found that the Ant Colony Algorithm [14], [15] yields better results than the other two. Babelfy [20] is a powerful graph-based disambiguation approach which uses random walks with restart over BabelNet, a large semantic network that integrates many resources, including WordNet. Babelfy is based on a dense subgraph heuristic for selecting coherent semantic interpretations of the input text. Panchenko *et al.* [49] proposed an unsupervised and knowledge-free word sense induction and disambiguation approach that relies on induced inventories as a pivot for learning sense feature representations.

Recently, word embeddings have been employed in various works [10], [18], [19], [21], [23], [24], [51] to improve WSD results. Word embeddings are long known in the NLP community [34], [35], but they have recently become more popular due to the work of Mikolov *et al.* [36], which introduced the *word2vec* framework that allows to efficiently build vector representations from words. Basile *et al.* [18] enhanced the Lesk algorithm by using word embeddings to compute the similarity between word definitions and the target context. Chen *et al.* [19] presented a unified model for joint word sense representation and disambiguation. They use the Skip-gram model to learn sense vectors. In the same time, Bhingardive *et al.* [21] used pre-trained word vectors to build sense embeddings by averaging the word vectors produced for each sense of a target word. As their goal was to find an approximation for the MCS baseline, they considered the sense embedding that is closest to the embedding vector of the target word. However, Yuan *et al.* [51] argued that a simple average or concatenation of word vectors loses the sequential and syntactic information of the text. Hence, they proposed to train a Long Short-Term Memory (LSTM) network to predict a held-out word in a sentence, which helps to better capture the sequential and syntactic patterns. Yuan *et al.* [51] extracted context vectors from the trained LSTM. Then, they could classify a word in a given context by finding the sense vector which has maximum cosine similarity to the corresponding context vector. Raganato *et al.* [52] conducted a more extensive study, comparing several neural sequence learning techniques, from bidirectional LSTM to encoder-decoder models. Iacobacci *et al.* [10] proposed different methods through which word embeddings can be leveraged in a *supervised* WSD system architecture. Interestingly, Iacobacci *et al.* [10] found that a WSD method based on word embeddings alone can provide significant performance improvements over a state-of-the-art WSD system that uses standard features for the WSD task.

More recently, Vial *et al.* [23] developed a new way of creating sense vectors for any dictionary using an existing word embeddings model, by summing the vectors of the terms inside a sense definition. They employed the sense vectors in the Cuckoo Search Algorithm, as a supplementary resource for their knowledge-based method. Tripodi and Pelillo [22] proposed to represent each ambiguous word as a node in a graph in which edges represent word relations, while senses are represented as classes. They used distributional information to quantify the influence that each word has on the decisions for other words, and semantic similarity information to measure the strength of compatibility among choices, formulating the WSD problem as a game and computing the equilibrium state of the system to perform disambiguation. Dongsuk *et al.* [24] proposed a knowledge-based WSD approach based on a novel subgraph construction strategy that selectively restricts the context of an ambiguous word. More precisely, contextual words of an ambiguous word are selected by thresholding the word similarities to the ambiguous word. Their word similarity measure is based on word embeddings. Pasini and Navigli [50] described two fully-automatic and language independent methods for computing the distribution of senses given a raw corpus of sentences. From the computed sense distributions, the authors could trivially determine most frequent sense annotations, surpassing other MCS baselines.

While ShotgunWSD performs global WSD using a different approach than other state-of-the-art unsupervised or knowledge-based methods [14]–[24], [31], [32], [42], [48]–[50], its source of inspiration, computational biology, makes it somewhat related to other bio-insipired approaches [14]–[16], [23]. Another important aspect is that ShotgunWSD relies on a local WSD algorithm that uses either the extended Lesk measure, as [31], [32], [42], or word embeddings, as [18], [19], [21], [23], [24]. To our knowledge, the outlier sense removal strategy proposed in this paper for ShotgunWSD 2.0 has not been previously studied in WSD research.

III. METHOD

As illustrated in Section I and also noted by Schwab *et al.* [14], brute-force WSD algorithms based on semantic relatedness [33] are not practical for whole text disambiguation due to their exponential time complexity. In this section, we describe a WSD algorithm that aims to avoid this computational issue. Our algorithm is inspired by the Shotgun genome sequencing technique [26] which is used in genetics research to obtain long DNA strands, a task known as *whole genome sequencing*. For instance, Istrail *et al.* [27] have used this technique to assemble the human genome. Before a long DNA strand can be read, Shotgun sequencing needs to create multiple copies of the respective strand. Then, DNA is randomly broken down into many small segments called *reads* (usually between 30 and 400 nucleotides long), by adding a restriction enzyme into the chemical solution containing the DNA. The reads can then be sequenced using Next-Generation Sequencing techonlogy [53], for example by using an Illumina (Solexa) machine [54]. In genome assembly, the low quality reads are usually eliminated [55] and the whole genome is reconstructed by assembling the remaining reads. One strategy is to merge two or more reads in order to obtain longer DNA segments, if they have a significant overlap of matching nucleotides. Because of reading errors or mutations, the overlap is usually measured using a distance measure, e.g. the edit distance [56]. If a backbone DNA sequence is available, the reads are aligned to the backbone DNA before assembly, in order to find their approximate position in the DNA that needs to be reconstructed. The Shotgun genome sequencing technique is applied because whole genomes cannot be effectively read.

We recognize a similar problem in WSD, as it is impractical to apply BF search at the document level. To this end, ShotgunWSD aims to replicate process involved in the Shotgun sequencing technique in order to build an optimal solution for global WSD. In order to apply Shotgun sequencing for WSD, we first need to make some correspondences. The long DNA strand that has to be sequenced corresponds to the text document that we aim to disambiguate, while the short DNA reads correspond to short context windows that are disambiguated through brute-force. As short DNA reads are assembled into a single long DNA strand, our goal is to combine the brute-force disambiguation solutions into a single solution (sense configuration) for the entire document, thus performing global WSD.

We now describe in detail how we adapt the Shotgun sequencing technique for the task of global WSD. We will make a few observations along the way that will lead to a simplified method, namely *ShotgunWSD*, which is formally presented in Algorithm 1. The three main phases of ShotgunWSD are also illustrated in Figure 1. We use the following notations in Algorithm 1. An array (or an ordered set of elements) is denoted by $X = (x_1, x_2, \ldots, x_m)$ and the length of X is denoted by $|X| = m$. Arrays are considered to be indexed starting from position 1, thus $X[i] = x_i, \forall i \in \{1, 2, \ldots m\}$.

Our goal is to find a configuration of senses G for the whole document D, that matches the ground-truth configuration produced by human annotators. A *configuration of senses* is simply obtained by assigning a sense to each word in the text document D. In this work, the senses are selected from WordNet [28], [29], according to steps 7-8 of Algorithm 1. Naturally, we will consider that the sense configuration of the whole document corresponds to the long DNA strand (whole genome) that needs to be sequenced. In this context, sense configurations of short context windows (less than 10 words) will correspond to the short DNA reads. A crucial difference here is that we know the location of the context windows in the whole document from the very beginning, so our task will be much easier compared to Shotgun sequencing (we do not need to use a backbone solution for the alignment of short sense configurations). At every possible location in the text document D, we select a window of n words according to step 12 of Algorithm 1. The window length n is an external parameter of our algorithm that can be tuned for an optimal trade-off between accuracy and speed. For each context window we compute all possible sense configurations, according to steps 14-15 of Algorithm 1. A score is assigned to each sense configuration by computing the semantic relatedness between word senses (steps 16-19), as described by Patwardhan *et al.* [33]. Butnaru *et al.* [25] alternatively employed two measures to compute the semantic relatedness, one is the extended Lesk measure [31], [32] and the other is a simple approach based on deriving sense embeddings from word embeddings [36]. In this paper, we propose a third approach that is based on clustering word vectors with k-means and on eliminating the smaller clusters (which contain outlier words). For the sake of completeness, all three methods are described in Section III-A. In the new version of ShotgunWSD, we modify step 19 in order to weight the relatedness score by the distance between the two ambiguous words, as in [10]. The reason for weighting the score is that if two words are farther apart from each other, their relatedness score should have a smaller contribution to the total score of the local sense configuration. For the assembly phase (steps 23-39), we keep the top scoring sense configurations, according to step 21 of Algorithm 1. In step 21, we use an internal parameter c in order to determine exactly how many sense configurations are retained per context window. Another important remark is that we assume

Algorithm 1 ShotgunWSD Algorithm

Input:

$D = (w_1, w_2, \ldots, w_m)$ – a document of m words denoted by w_i, $i \in \{1, 2, \ldots, m\}$;

n – the length of the context windows ($1 < n < m$);

k – the number of sense configurations considered for the voting scheme ($k > 0$);

Initialization:

$c \leftarrow 20$;

for $i \in \{1, 2, \ldots, m\}$ **do**

  $S_{w_i} \leftarrow$ the set of WordNet synsets of w_i;

$\mathcal{S} \leftarrow \emptyset$;

$G \leftarrow (0, 0, \ldots., 0)$, such that $|G| = m$;

Computation:

for $i \in \{1, 2, \ldots, m - n + 1\}$ **do**

  $\mathcal{C}_i \leftarrow \emptyset$;

  while *did not generate all sense configurations* **do**

    $C \leftarrow$ a new configuration $(s_{w_i}, s_{w_{i+1}}, \ldots, s_{w_{i+n-1}})$, $s_{w_j} \in S_{w_j}$, $\forall j \in \{i, \ldots, i + n - 1\}$, such that $C \notin \mathcal{C}_i$;

    $r \leftarrow 0$;

    for $p \in \{1, 2, \ldots, n - 1\}$ **do**

      for $q \in \{p + 1, 2, \ldots, n\}$ **do**

        $r \leftarrow r + 0.1^{\frac{|p-q|-1}{n-1}} \cdot \text{relatedness}(C[p], C[q])$;

    $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{(C, i, n, r)\}$;

  $\mathcal{C}_i \leftarrow$ the top c configurations obtained by sorting the configurations in $\mathcal{C}_i$ by their relatedness score (descending);

  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{C}_i$;

for $l \in \{min\{5, n - 1\}, \ldots, 1\}$ **do**

  for $p \in \{1, 2, \ldots, |\mathcal{S}|\}$ **do**

    $(C_p, i_p, n_p, r_p) \leftarrow$ the p-th component of $\mathcal{S}$;

    for $q \in \{1, 2, \ldots, |\mathcal{S}|\}$ **do**

      $(C_q, i_q, n_q, r_q) \leftarrow$ the q-th component of $\mathcal{S}$;

      if $i_q - i_p < n_p$ *and* $i_p \neq i_q$ **then**

        $t \leftarrow$ true;

        for $x \in \{1, \ldots, l\}$ **do**

          if $C_p[n_p - l + x] \neq C_q[x]$ **then**

            $t \leftarrow$ false;

        if $t = true$ **then**

          $C_{p \oplus q} \leftarrow (C_p[1], C_p[2], \ldots, C_p[n_p], C_q[l + 1], C_q[l + 2], \ldots, C_q[n_q])$;

          $r_{p \oplus q} \leftarrow r_p$;

          for $i \in \{1, 2, \ldots, n_p + n_q - l\}$ **do**

            for $j \in \{l + 1, l + 2, \ldots, n_q\}$ **do**

              $r_{p \oplus q} \leftarrow r_{p \oplus q} + 0.1^{\frac{|i-j|-1}{n-1}} \cdot \text{relatedness}(C_{p \oplus q}[i], C_q[j])$;

          $\mathcal{S} \leftarrow \mathcal{S} \cup \{(C_{p \oplus q}, i_p, n_p + n_q - l, r_{p \oplus q})\}$;

for $j \in \{1, 2, \ldots, m\}$ **do**

  $\mathcal{Q}_j \leftarrow \{(C, i, d, r) \mid (C, i, d, r) \in \mathcal{S}, j \in \{i, i + 1, \ldots, i + d - 1\}\}$;

  $\mathcal{Q}_j \leftarrow$ the top k configurations obtained by sorting the configurations in $\mathcal{Q}_j$ by their length (descending);

  $ps_{w_j} \leftarrow$ the predominant sense obtained by using a majority voting scheme on $\mathcal{Q}_j$;

  $G[j] \leftarrow ps_{w_j}$;

Output:

$G = (ps_{w_1}, ps_{w_2}, \ldots, ps_{w_m})$, $ps_{w_i} \in S_{w_i}$, $\forall i \in \{1, 2, \ldots, m\}$ – the global configuration of senses returned by the algorithm.

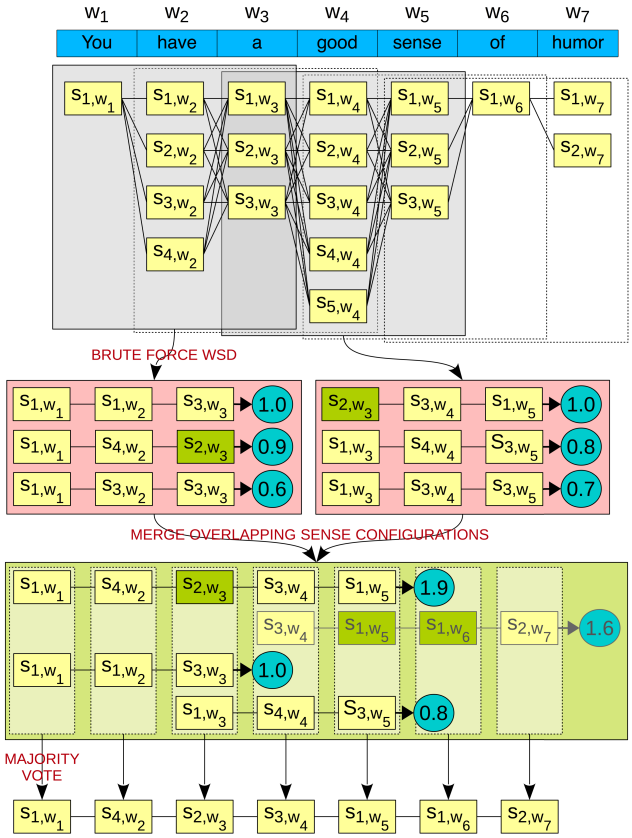

FIGURE 1. An example of building a global sense configuration with ShotgunWSD for a document of 7 words. The algorithm is based on three main phases: building local sense configurations using a brute-force approach, assembling shorter configurations into longer configurations by prefix-suffix matching and majority voting. Best viewed in color.

that the BF algorithm used for generating sense configurations for short windows does not produce output errors (as in genome sequencing), hence it is not necessary to use a distance measure in order to find overlaps for merging configurations. We simply verify if the suffix of a former configuration coincides (matches exactly) with the prefix of a latter configuration in order to join them together, according to steps 29-33 of Algorithm 1. The length l of the suffix and the prefix that get overlapped needs to be strictly greater then zero, so at least one sense choice needs to coincide. We gradually consider shorter and shorter suffix and prefix lengths starting with $l = min\{5, n-1\}$, according to step 23 of Algorithm 1. Sense configurations are assembled in order to obtain longer configurations (step 34), until none of the resulted configurations can be further merged together. When merging, the relatedness score of the resulting configuration needs to be recomputed (steps 36-38), but we can take advantage of some of the previously computed scores (step 35). Longer configurations indicate that there is an agreement (regarding the chosen senses) that spans across a longer phrase. In other words, longer configurations are more likely to provide correct sense choices, since they inherently embed a higher degree of agreement among senses. After the configuration assembly phase, we start assigning the sense to each word in the document, according to step 40 of Algorithm 1. According to step 42, we build a ranked list of sense configurations for each word in the document, based on the assumption that longer configurations provide better information about correct word senses. Naturally, for a given word, we only consider the configurations that contain the respective word, according to step 41. Finally, the sense of each word is given by a majority vote on the top k configurations from its ranked list, according to steps 43-44 of Algorithm 1. The number of sense configurations k is an external parameter of our approach, and it can be tuned for optimal results.

A. SEMANTIC RELATEDNESS

For a sense configuration assigned to a context window of n words, we compute a semantic relatedness score (numeric value) between each pair of selected senses. In steps 19 and 38 of Algorithm 1, the score is computed by the *relatedness* function, which takes two word senses as input and provides their semantic relatedness score as output. Butnaru *et al.* [25] used two alternative approaches for computing the relatedness score. In this paper, we propose a third approach. We note that all three approaches are built on top of WordNet semantic relations. Each of the three approaches can be regarded as a different way of estimating the semantic relatedness of two WordNet synsets. For each synset, we first build a *disambiguation vocabulary* (also referred to as *sense bag*) by extracting words from the WordNet [28], [29] lexical knowledge base, as described next. Starting from the synset itself, we first include all the synonyms that form the respective synset along with the content words of the gloss (examples included). We also include into the disambiguation vocabulary words indicated by specific WordNet semantic relations that are chosen according to the part-of-speech of the ambiguous word. More precisely, we have considered hyponyms and meronyms for nouns. For adjectives, we have considered similar synsets, antonyms, attributes, pertainyms and related (see also) synsets. For verbs, we have considered troponyms, hypernyms, entailments and outcomes. Finally, for adverbs, we have considered antonyms, pertainyms and topics. These choices have been made because previous studies [9], [32] have reached the conclusion that using these specific relations for each part-of-speech provides better empirical results for the WSD task. For the sake of completeness, we next describe the two approaches for computing the relatedness score proposed in [25], as well as our novel approach based on removing outlier words using k-means clustering.

1) EXTENDED LESK MEASURE

The original Lesk algorithm [30] takes into account one word overlaps among the glosses of a target word and those that surround it in a given context. Banerjee and Pedersen [31] consider that this is a significant limitation of the original Lesk algorithm, since dictionary glosses tend to be fairly short and they fail to provide sufficient information to make fine grained distinctions between word senses. To this end,

Banerjee and Pedersen [32] extend the original Lesk algorithm with a measure that takes two WordNet synsets as input and returns a numeric value that quantifies their degree of semantic relatedness by taking into consideration the glosses of related WordNet synsets as well. Moreover, when comparing two glosses, the extended Lesk measure considers overlaps of multiple consecutive words, based on the assumption that a longer phrase is more representative for the relatedness of the two synsets. Before applying the extended Lesk algorithm, we eliminate the stopwords. The remaining words are stemmed using the Porter stemmer algorithm [57]. The stemming process reduces a word to its root form by removing the most common morphological and inflexional endings from words in English. The resulting stems represent the final set of features that we use for computing the relatedness score between two synsets. Given two input glosses, the longest overlap between them is detected and then replaced with a unique marker (symbol) in each of the two glosses. The resulting glosses are then again checked for overlaps, and this process continues until there are no more overlaps. The lengths of the detected overlaps are squared and added together to obtain the score for the given pair of glosses. Depending on the WordNet relations used for each part-of-speech, several pairs of glosses are compared and summed up together to obtain the final relatedness score. However, WordNet does not define semantic relations between synsets if they do not belong to the same part-of-speech. For this reason, we compute the semantic relatedness using only the WordNet glosses and examples when two words are of different parts-of-speech. Further details regarding the extended Lesk measure are provided by Banerjee and Pedersen [32].

2) SENSE EMBEDDINGS

In this section, we describe a simple approach based on word embeddings to measure the semantic relatedness of two synsets. Approaches based on *word embeddings* [34]–[36] represent words as low-dimensional real-valued vectors, such that semantically related words reside in close vicinity in the generated space. In our algorithm, we employ the pre-trained word embeddings computed by the *word2vec* framework [36] on the Google News data set using the Skip-gram model. This pre-trained model contains 300-dimensional vectors for nearly 3 million words and phrases.

We compute the relatedness score between two synsets as described next. We eliminate the stopwords and embed each remaining word in the disambiguation vocabulary of a synset in order to obtain the corresponding word vector. We thus obtain a set of word embedding vectors for each given synset. We derive the sense embedding for a synset simply by computing the median of all the word embeddings in the corresponding set. We can naturally assume that some of the word vectors in the set correspond to words that do not help the disambiguation process. From this point of view, these words can be regarded as outliers. In this context, we consider that using the (geometric) median instead of the mean is more appropriate, as the mean is largely influenced by outliers. It is important to note that our third approach (presented next) for computing the semantic relatedness aims to properly address the outlier removal issue. According to our second approach, the semantic relatedness of two synsets is simply given by the cosine similarity between their median vectors:

$$relatedness(A, B) = \frac{\sum_{i=1}^{m} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{m} a_i^2}\sqrt{\sum_{i=1}^{m} b_i^2}},$$

where A and B are m-dimensional median vectors corresponding to two WordNet synsets. For the employed *word2vec* model, the vectors have $m = 300$ components.

An important remark is that Bhingardive *et al.* [21] proposed an approach based on the mean (instead of the median) of word vectors to construct sense embeddings, but with a slightly different purpose than ours, namely to determine which synset is most similar to the target word, assuming that the respective synset should correspond to the most common sense of the target word. As such, they completely disregard the context of the target word. Different from their approach, we are trying to measure the semantic relatedness between two synsets of distinct words that appear in the same context window. Furthermore, the empirical results presented in Section IV prove that our approach provides better performance than the MCS estimation approach of Bhingardive *et al.* [21], thus putting an even greater gap between their method and ours.

3) SENSE EMBEDDINGS AFTER OUTLIER REMOVAL

We start by gathering the words in the document that we aim to disambiguate into a set. Along with the words in the document, we also add all the words from the disambiguation vocabularies of each sense of each ambiguous word in the document. In the whole process, stopwords are disregarded. Each word is then embedded into a vector space using the *word2vec* [36] framework. Based on the fact that word embeddings carry semantic information by projecting semantically related words in the same region of the embedding space, the next step is to cluster the word vectors in order to obtain relevant semantic clusters of words. The words are clustered using k-means clustering with k-means++ [58] initialization.

Next, we eliminate the clusters with fewer samples, based on the assumption that these smaller clusters contain mostly outlier samples. We motivate our assumption through the following toy example. We generate 400 data points sampled from two normal distributions of different means. We group the points into $k = 30$ clusters using k-means and we illustrate the result in Figure 2. We then count the number of points in each cluster and obtain the histogram depicted in Figure 3. In this example, we consider that the clusters with less than 10 data points contain mostly outliers. The centroids of these smaller clusters are marked with a large blue square in Figure 2. We can clearly see that the marked clusters are farthest from both normal distribution means,

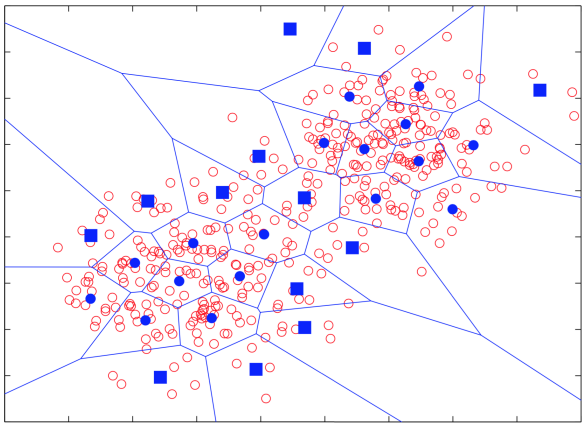

FIGURE 2. A set of 400 data points sampled from two normal distributions of different means. The points are clustered into 30 clusters using k-means. The centroids of clusters with less than 10 samples are represented with a large blue square. Best viewed in color.

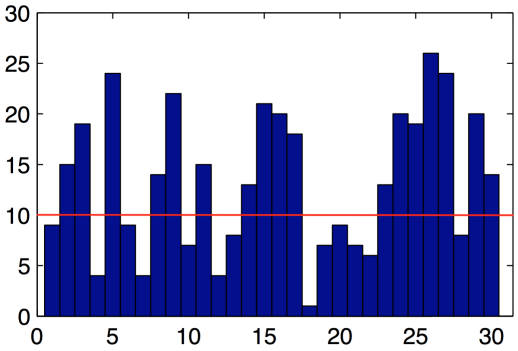

FIGURE 3. A histogram representing the number of data points in each cluster. The histogram corresponds to the k-means clustering applied over the 400 data points illustrated in Figure 2. A threshold of 10 is used to detect clusters of outliers. Best viewed in color.

indicating that the containing points are indeed outliers. This assumption is also supported by the results obtained by Ionescu *et al.* [59] in abnormal event detection in video, as they employ the same approach (based on k-means) to remove clusters of outlier motion and appearance samples. Nevertheless, our aim is to test out this assumption by quantifying the performance improvement of ShotgunWSD 2.0 over its previous version. To this end, we remove the words that belong to the eliminated clusters from each and every sense bag. We next compute the median of all the remaining word embeddings in a given sense bag to obtain a sense embedding for the corresponding word sense. The semantic relatedness of two synsets is given by the cosine similarity between the corresponding medians.

IV. EXPERIMENTS AND RESULTS

The goal of our experiments is to compare ShotgunWSD 2.0 with recent and well-known unsupervised or knowledge-based methods from the literature, including ShotgunWSD 1.0. On each data set, we include the MCS baseline, which is relevant for unsupervised or knowledge-based methods such as ShotgunWSD 2.0. As in common practice, we evaluate our approach on data sets that are annotated with corresponding WordNet senses. Similar to other methods from the literature, our algorithm has to choose the correct sense from the WordNet sense inventory. In practical applications, the correct sense of a word might not be present in the WordNet sense inventory. In this case, our algorithm will not be able to select the right sense.

The rest of this section is organized as follows. In Section IV-A, we present the data sets considered in the evaluation. We provide details about parameter tuning in Section IV-B. We hereby note that we used the same parameters for ShotgunWSD 2.0 in all our experiments, irrespective of the data set. In Section IV-C, we introduce the considered baseline methods. In the following six sections, we present the results on each of the six data sets. Finally, we provide a discussion that summarizes the experiments in Section IV-J.

A. DATA SETS

We evaluate the first and the second versions of our global WSD algorithm on five individual and unified data sets. The data sets considered in our evaluation are SemEval 2007 [37], Senseval-2 [38], Senseval-3 [39], SemEval 2013 [40] and SemEval 2015 [41]. The SemEval 2007 coarse-grained English all-words data set[3] consists of 5 documents that contain 2269 ambiguous words altogether. The Senseval-2 data set[4] consists of 3 documents that contain 2473 ambiguous words, while the Senseval-3 data set[4] consists of 3 documents that contain 2081 ambiguous words. The SemEval 2013 data set [40] is formed of 13 documents that contain 1644 ambiguous words (nouns) in the WordNet sense inventory. The SemEval 2015 data set [41] is composed of 4 documents that contain 1175 ambiguous words. A summary of the distribution of ambiguous words per part-of-speech in each data set is presented in Table 1. The unified set containing all these five benchmarks is described in [11].

B. PARAMETER TUNING

For ShotgunWSD 1.0, we use the same parameters as Butnaru *et al.* [25]. Thus, the internal parameter c, i.e. the number of sense configurations retained per context window, is set to 20 and the length of the context windows is set to $n = 8$. With these settings, ShotgunWSD 1.0 runs in 187 seconds on the first document of SemEval 2007. The reported time is measured on a computer with Intel Core i7 3.4 GHz processor and 16 GB of RAM using a single Core. The final sense for each word is assigned using a majority vote based on the top $k = 15$ configurations. For ShotgunWSD 2.0, we use the same values for the parameters c and k. However, the k-means clustering step requires additional processing time, so we need to reduce the window length from $n = 8$ to $n = 6$ in order to reach a processing time of the same order of magnitude as the processing time required by ShotgunWSD 1.0. On the same machine, ShotgunWSD 2.0 with $n = 6$

[3]http://lcl.uniroma1.it/coarse-grained-aw
[4]http://web.eecs.umich.edu/∼mihalcea/downloads.html

TABLE 1. A summary of the number of ambiguous words along with the distribution of ambiguous words per part-of-speech in the five data sets considered in our evaluation.

Data Set	Words	Nouns	Adjectives	Verbs	Adverbs
SemEval 2007	2269	1108	362	591	208
Senseval-2	2473	1136	457	581	299
Senseval-3	2081	951	364	751	15
SemEval 2013	1644	1644	0	0	0
SemEval 2015	1175	671	166	254	84

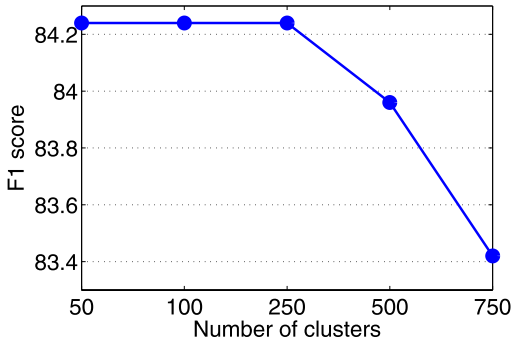

FIGURE 4. The F_1 scores of ShotgunWSD 2.0 on the first document of SemEval 2007, using different numbers of clusters for k-means.

runs in 105 seconds on the first document of SemEval 2007, which is slightly faster than ShotgunWSD 1.0 with $n = 8$. Butnaru *et al.* [25] showed that the processing time grows exponentially with the window length, hence $n = 8$ would not be a reasonable choice for ShotgunWSD 2.0.

There are two additional parameters for ShotgunWSD 2.0: the number of k-means clusters and the threshold used for outlier cluster elimination. As Schwab *et al.* [15] and Butnaru *et al.* [25], we tune our parameters on the first document of SemEval 2007. By tuning the parameters on just one document from SemEval 2007, we avoid the overfitting to a particular data set. We tested our algorithm using 50, 100, 250, 500 and 750 clusters and we found out that the performance starts to slightly drop after 250 clusters, as illustrated in Figure 4. In the end, we opted to use 250 clusters in all the experiments. We also tried to find out if eliminating 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45% or 50% of the smaller (outlier) clusters would have a different effect on performance. As shown in Figure 5, it seems that eliminating 25% of the clusters is the optimal choice. It is important to note that we use the same parameters throughout the subsequent experiments on all six data sets.

C. BASELINES

We compare both versions of ShotgunWSD with several state-of-the-art unsupervised and knowledge-based WSD methods, as long as the works presenting these methods [15], [18]–[23], [42]–[45] report results on the data sets considered in our evaluation. We hereby note that some of these works do not report results on all six data sets, hence the baselines on each data set may be different.

We first compare ShotgunWSD with the MCS baseline and four state-of-the-art approaches [15], [19], [22], [23] on the SemEval 2007 coarse-grained English all-words task [37]. While Schwab *et al.* [15] and Chen *et al.* [19] report results on SemEval 2007, Bhingardive *et al.* [21] and Torres and Gelbukh [42] report results on Senseval-2 and Senseval-3. Hence, we also compare our approach with the MCS baseline, the MCS estimation method of Bhingardive *et al.* [21], the extended Lesk algorithm [42] and the unsupervised approach based on game theory [22] on the Senseval-2 English all-words [38] and the Senseval-3 English all-words [39] data sets. On the SemEval 2013 English nouns WSD task [40], the baselines are Babelfy [20], the winner [43] and the first runner up [15] of the SemEval 2013 WSD task, which submitted results for the WordNet sense inventory. In a similar manner, we compare our approach with the winner [45], the first runner up [44] and a more recent approach [23] on the SemEval 2015 English all-words WSD task [41]. On the unified benchmark [11], we consider as baselines the WordNet MCS [11], the extended Lesk algorithm [11], an enhanced Lesk algorithm based on word embeddings [18], and Babelfy [20]. We next provide a brief description of each unsupervised or knowledge-based baseline:

- Extended Lesk [42] – The extended Lesk algorithm takes two WordNet synsets as input and returns a numeric value that quantifies their degree of semantic relatedness by taking into consideration the glosses of the two synsets as well as the glosses of the related synsets. Torres and Gelbukh [42] applied the extended Lesk algorithm to disambiguate every sentence using brute-force.
- Genetic Algorithms [15] – The Genetic Algorithm is a global WSD approach that can be divided into five phases: initialization, selection, crossover, mutation and evaluation. The algorithm starts with a randomly initialized population of sense configurations which are modified through genetic operations and sorted according to a version of the extended Lesk measure.
- Simulated Annealing [15] – The Simulated Annealing approach is a global WSD approach that is inspired from the physical phenomenon of metal cooling. The algorithm uses a single randomly initialized configuration and it performs a random change at every iteration. The algorithm accepts configurations with lower extended Lesk scores in order to prevent the algorithm from converging to local maxima.
- Ant Colony [15] – The Ant Colony Algorithm is a global WSD approach that is inspired by the social behavior of ants. The environment of the Ant Colony Algorithm is

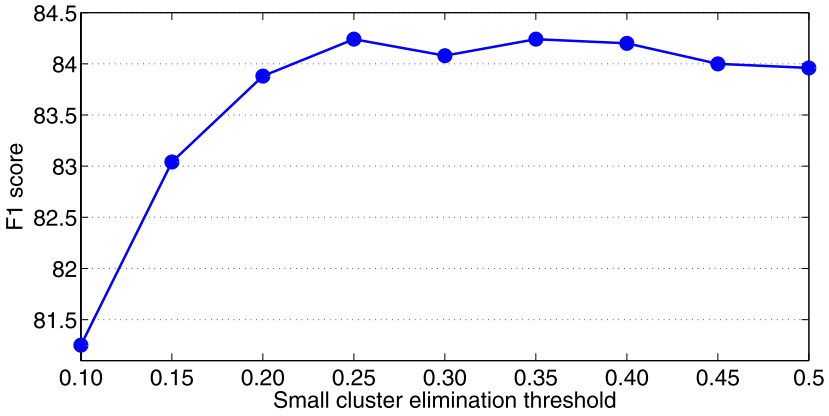

FIGURE 5. The F_1 scores of ShotgunWSD 2.0 on the first document of SemEval 2007, using different thresholds for eliminating the smaller k-means clusters.

a graph containing two types of nodes: nests and plain nodes. Nests produce ants that move in the graph in order to find energy and bring it back to their mother nest. Depending on the lexical information present and the structure of the graph, ants tend to follow links (edges) between more closely related word senses.

- UMCC-DLSI [43] – The UMCC-DLSI approach is based on the ISR-WN resource, which enriches the WordNet semantic network using edges from multiple lexical resources. WSD is performed using the ISR-WN network in conjunction with an extension of the Personalized PageRank algorithm [47], which includes sense frequencies. The Personalized PageRank algorithm requires initializing the PageRank algorithm with a set of seed synsets (vertices).
- Extended Lesk + Embeddings [18] – The algorithm proposed by Basile *et al.* [18] is an enhanced version of the Lesk algorithm, in which word embeddings are leveraged to compute the similarity between BabelNet definitions and the target context. The method relies on the use of a word similarity function defined on a distributional semantic space to compute the gloss-context overlap.
- S2C Unsupervised [19] – The S2C (simple to complex) approach is based on a unified model for joint word sense representation and disambiguation. S2C disambiguates the words with fewer senses first, since these are easier to disambiguate. The Skip-gram model is employed to learn sense vectors, and disambiguation is performed by computing the cosine similarities between the context vector and the sense vectors of an ambiguous word.
- Babelfy [20] – The Babelfy system is based on the BabelNet multilingual semantic network and performs disambiguation and entity linking. Given an input text, Babelfy extracts all the linkable fragments from the text. Then, for each fragment, it lists the possible meanings according to BabelNet. Using pre-computed semantic signatures from BabelNet, the algorithm constructs a graph-based semantic representation of the input text by linking the candidate meanings of the extracted fragments. Finally, it finds a dense subgraph of the constructed representation, selecting the best candidate meaning for each fragment.
- MCS Estimation [21] – In order to estimate the MCS, Bhingardive *et al.* [21] proposed an approach based on the mean of word vectors in a synset, for deriving a sense embedding for the respective synset. The algorithm uses the cosine similarity in order to determine which sense embedding is most similar to the target word, assuming that the respective sense embedding should correspond to the most common sense of the target word. The approach completely disregards the context of the target word.
- LIMSI [45] – The LIMSI system performs WSD by taking advantage of the parallelism of the multilingual test data. The texts are first aligned by sentence and by word. Next, content words are tagged by their translations in other languages. The alignments serve to retrieve the BabelNet synsets that are relevant for each instance of a word, i.e. the synsets that contain both the disambiguation target and its aligned translation.
- Sudoku [44] – The Sudoku approach employs Personalised PageRank [47] to select the best candidate. The algorithm starts with a surfing vector biased towards monosemous words. Words are disambiguated in the increasing order of polysemy, using a semantic subgraph constructed from BabelNet.
- Unsupervised Game Theory [22] – The algorithm proposed by Tripodi and Pelillo [22] is based on evolutionary game theory. Each word to be disambiguated is represented as a node in a graph with edges representing word relations, while senses are represented as classes. The words simultaneously update their class membership preferences according to the senses that neighboring words are likely to choose.

TABLE 2. The F_1 scores of ShotgunWSD 1.0 and ShotgunWSD 2.0 versus the F_1 scores of the MCS baseline and various state-of-the-art WSD approaches, on the SemEval 2007 coarse-grained English all-words task. The results reported for ShotgunWSD 1.0 are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations. The results reported for ShotgunWSD 2.0 are obtained for windows of $n = 6$ words, a majority vote on the top $k = 15$ configurations and k-means clustering with 250 clusters. The best F_1 score is highlighted in bold.

Method	F_1 Score
Most Common Sense	78.89%
Genetic Algorithms [15]	74.53%
Simulated Annealing [15]	75.18%
Ant Colony [15]	79.03%
S2C Unsupervised [19]	75.80%
Unsupervised Game Theory [22]	80.40%
VecLesk [23]	75.29%
ShotgunWSD 1.0 + Extended Lesk [25]	79.15%
ShotgunWSD 1.0 + Sense Embeddings [25]	79.68%
ShotgunWSD 2.0	**81.22%**

- VecLesk [23] – The VecLesk approach is a global knowledge-based WSD system based on the Cuckoo Search Algorithm. The approach is based on a new way of creating sense vectors for any dictionary using an existing word embeddings model, by summing the vectors of the terms inside a sense definition. The sum of word vectors is weighted according to the inverse document frequency of each word and to the part-of-speech.

D. RESULTS ON SEMEVAL 2007

First, we conduct an empirical study on the SemEval 2007 coarse-grained English all-words task in order to evaluate the performance of ShotgunWSD 1.0 and 2.0 as well as other WSD methods. As described in Section III-A, we use two alternative approaches for computing the semantic relatedness scores in ShotgunWSD 1.0, namely extended Lesk and sense embeddings. ShotgunWSD 2.0 is based on a different approach which eliminates smaller clusters of word embeddings. We compare our two versions of ShotgunWSD with several bio-inspired algorithms described in [14], [15], namely a Genetic Algorithm, Simulated Annealing, and Ant Colony Optimization. Along with the bio-inspired algorithms, we include two approaches based on sense embeddings [19], [23] and one based on game theory [22] in our comparative study. All the approaches considered in the evaluation are unsupervised or knowledge-based. We compare them with the MCS baseline that is based on human annotations. The F_1 scores of the enumerated methods are all presented in Table 2. Among all state-of-the-art methods, two approaches [15], [22] are capable of surpassing the MCS baseline. The unsupervised S2C approach yields roughly 3% lower results than the MCS baseline, but Chen *et al.* [19] report better results in a semi-supervised setting. The VecLesk [23] knowledge-based algorithm provides a performance level similar to the unsupervised S2C approach. All versions of ShotgunWSD attain better results than the MCS baseline (78.89%) and the best bio-inspired method, namely the Ant Colony Optimization algorithm (79.03%). Indeed, ShotgunWSD 1.0 obtains an F_1 score of 79.15% when using the extended Lesk measure and an F_1 score of 79.68% when using sense embeddings. Interestingly, we observe that ShotgunWSD 1.0 gives slightly better results when sense embeddings are used instead of the extended Lesk method. Nevertheless, ShotgunWSD 2.0 obtains even better results. It surpasses ShotgunWSD 1.0 by 1.54% and the second best approach [22] by 0.82%. To our knowledge, the F_1 score of ShotgunWSD 2.0 (81.22%) is the best among all unsupervised methods evaluated on the SemEval 2007 coarse-grained English all-words task.

TABLE 3. The F_1 scores of ShotgunWSD 1.0 and ShotgunWSD 2.0 versus the F_1 scores of the MCS baseline, two unsupervised WSD approaches and the extended Lesk measure, on the Senseval-2 English all-words data set. The results reported for ShotgunWSD 1.0 are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations. The results reported for ShotgunWSD 2.0 are obtained for windows of $n = 6$ words, a majority vote on the top $k = 15$ configurations and k-means clustering with 250 clusters. The best F_1 score is highlighted in bold.

Method	F_1 Score
Most Common Sense	60.10%
MCS Estimation [21]	52.34%
Extended Lesk [42]	54.60%
Unsupervised Game Theory [22]	**61.20%**
ShotgunWSD 1.0 + Extended Lesk [25]	55.78%
ShotgunWSD 1.0 + Sense Embeddings [25]	57.55%
ShotgunWSD 2.0	58.24%

E. RESULTS ON SENSEVAL-2

In Table 3, we present the F_1 scores of the two versions of ShotgunWSD against the MCS baseline, the MCS estimation approach [21], the extended Lesk measure [42] and the unsupervised approach based on game theory [22] on the Senseval-2 English all-words data set. The empirical results presented in Table 3 indicate that ShotgunWSD 1.0 based on sense embeddings obtains an F_1 score that is almost 5% better than the F_1 score of Bhingardive *et al.* [21]. In the same time, ShotgunWSD 1.0 based on the extended Lesk method gives an F_1 score that is around 1% better than the F_1 score reported by Torres and Gelbukh [42]. It is important to note that Torres and Gelbukh [42] apply the extended Lesk measure by performing the brute-force search at the sentence level (not on the whole document), hence it is not surprising that ShotgunWSD 1.0 is able obtain better results.

Despite of using windows of shorter length (6 instead of 8), ShotgunWSD 2.0 is able to yield better performance than both variants of ShotgunWSD 1.0. The improvement with respect to the better variant of ShotgunWSD 1.0, the one based on sense embeddings, is 0.69%. However, ShotgunWSD 2.0 (58.24%) is still under the MCS baseline (60.10%) on this data set. Remarkably, the unsupervised approach based on game theory [22] is able to surpass the MCS baseline on Senseval-2.

F. RESULTS ON SENSEVAL-3

We also compare ShotgunWSD 1.0 and 2.0 with the MCS baseline, the MCS estimation approach of Bhingardive *et al.* [21], the extended Lesk measure [42] and the unsupervised

TABLE 4. The F_1 scores of ShotgunWSD 1.0 and ShotgunWSD 2.0 versus the F_1 scores of the MCS baseline, two unsupervised WSD approaches and the extended Lesk measure, on the Senseval-3 English all-words data set. The results reported for ShotgunWSD 1.0 are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations. The results reported for ShotgunWSD 2.0 are obtained for windows of $n = 6$ words, a majority vote on the top $k = 15$ configurations and k-means clustering with 250 clusters. The best F_1 score is highlighted in bold.

Method	F_1 Score
Most Common Sense	**62.30%**
MCS Estimation [21]	43.28%
Extended Lesk [42]	49.60%
Unsupervised Game Theory [22]	59.10%
ShotgunWSD 1.0 + Extended Lesk [25]	57.89%
ShotgunWSD 1.0 + Sense Embeddings [25]	59.82%
ShotgunWSD 2.0	59.92%

TABLE 5. The F_1 scores of ShotgunWSD 1.0 and ShotgunWSD 2.0 versus the F_1 scores of the MCS baseline and three state-of-the-art WSD approaches, on the SemEval 2013 English all-words task. The results reported for ShotgunWSD 1.0 are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations. The results reported for ShotgunWSD 2.0 are obtained for windows of $n = 6$ words, a majority vote on the top $k = 15$ configurations and k-means clustering with 250 clusters. The best F_1 score is highlighted in bold.

Method	F_1 Score
Most Common Sense	63.00%
UMCC-DLSI [43]	64.70%
Ant Colony [15]	51.40%
Babelfy [20]	65.90%
ShotgunWSD 1.0 + Extended Lesk [25]	57.36%
ShotgunWSD 1.0 + Sense Embeddings [25]	62.89%
ShotgunWSD 2.0	**63.05%**

approach based on game theory [22] on the Senseval-3 English all-words data set. The corresponding F_1 scores are presented in Table 4. With an F_1 score of 57.89%, ShotgunWSD 1.0 based on the extend Lesk measure brings a remarkable improvement of 8% over the extended Lesk algorithm applied at the sentence level [42]. Moreover, the empirical results indicate that all versions of ShotgunWSD reach considerably better F_1 scores compared to the MCS estimation approach [21]. By using sense embeddings in a completely different way than Bhingardive *et al.* [21], ShotgunWSD 1.0 attains an F_1 score of 59.82%, which is 16.54% above the MCS estimation approach [21]. On this data set, the improvement of ShotgunWSD 2.0 over ShotgunWSD 1.0 is very small (from 59.82% to 59.92%). However, ShotgunWSD 2.0 surpasses all baselines, including the unsupervised approach based on game theory [22].

G. RESULTS ON SEMEVAL 2013

For the SemEval 2013 English all-words task, we report the F_1 scores of ShotgunWSD 1.0 and ShotgunWSD 2.0 using WordNet senses in Table 5. The table includes several state-of-the-art approaches [15], [20], [43] for reference. When we use the extended Lesk as local WSD algorithm, the performance of ShotgunWSD 1.0 (57.36%) is roughly 6% under the MCS baseline (63.00%). A similar difference is recorder between the ShotgunWSD variant based on extended Lesk and the other two variants

TABLE 6. The F_1 scores of ShotgunWSD 1.0 and ShotgunWSD 2.0 versus the F_1 scores of the BabelNet First Sense and three state-of-the-art WSD approaches, on the SemEval 2015 English all-words task. The results reported for ShotgunWSD 1.0 are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations. The results reported for ShotgunWSD 2.0 are obtained for windows of $n = 6$ words, a majority vote on the top $k = 15$ configurations and k-means clustering with 250 clusters. The best F_1 score is highlighted in bold.

Method	F_1 Score
BabelNet First Sense [41]	**66.30%**
LIMSI [45]	64.70%
Sudoku [44]	59.90%
VecLesk [23]	59.01%
ShotgunWSD 1.0 + Extended Lesk [25]	45.66%
ShotgunWSD 1.0 + Sense Embeddings [25]	58.44%
ShotgunWSD 2.0	61.30%

of ShotgunWSD based on sense embeddings. All variants of ShotgunWSD achieve better performance than the Ant Colony approach [15]. With an F_1 score of 63.05%, ShotgunWSD 2.0 is able to marginally surpass the MCS baseline on SemEval 2013, but it stays behind the UMCC-DLSI [43] and the Babelfy [20] approaches.

H. RESULTS ON SEMEVAL 2015

Table 6 shows the results of ShotgunWSD 1.0 and 2.0 against a recent knowledge-based approach [23] and the top two methods [44], [45] from the SemEval 2015 English all-words WSD task [41]. The table also includes the BabelNet first sense (BFS) [41] as reference. We first note that ShotgunWSD 1.0 based on the extended Lesk gives considerably worse results (45.66%) than ShotgunWSD 1.0 based on sense embeddings (58.44%). In the same time, ShotgunWSD 2.0 attains better performance than both variants of ShotgunWSD 1.0. The F_1 score of ShotgunWSD 2.0 (61.30%) is also 1.4% better than Sudoku [44] and 2.3% better than VecLesk [23]. On the other hand, the performance of ShotgunWSD 2.0 is 3.4% under the performance of the winners [45] of the SemEval 2015 English all-words WSD task. However, it is important to remark that Apidianaki and Gong [45] exploit the parallelism of the multilingual SemEval 2015 test data by using translations as source of indirect supervision for sense selection. As Manion [44] and the rest of the participants [41], we do not use this kind of information in our algorithm.

I. RESULTS ON UNIFIED DATA SETS

We report results for the recent evaluation setting proposed by Raganato *et al.* [11] in Table 7. The results of ShotgunWSD are compared against the same baselines considered in [11]. We note that ShotgunWSD 1.0 based on the extended Lesk (59.09%) surpasses the baseline extended Lesk (48.70%) by roughly 10%. When ShotgunWSD 1.0 employs sense embeddings (62.74%), the results improve by more than 3%. However, both variants of ShotgunWSD 1.0 reach performance levels below the state-of-the-art approaches [18], [20]. ShotgunWSD 2.0 (63.84%) manages to surpass the approach of Basile *et al.* [18], but it is still under the MCS

TABLE 7. The F_1 scores of ShotgunWSD 1.0 and ShotgunWSD 2.0 versus the F_1 scores of two state-of-the-art WSD approaches, on the unified data sets. The results reported for ShotgunWSD 1.0 are obtained for windows of $n = 8$ words and a majority vote on the top $k = 15$ configurations. The results reported for ShotgunWSD 2.0 are obtained for windows of $n = 6$ words, a majority vote on the top $k = 15$ configurations and k-means clustering with 250 clusters. The best F_1 score is highlighted in bold.

Method	F_1 Score
Most Common Sense	**65.20%**
Extended Lesk	48.70%
Extended Lesk + Embeddings [18]	63.70%
Babelfy [20]	65.50%
ShotgunWSD 1.0 + Extended Lesk [25]	59.09%
ShotgunWSD 1.0 + Sense Embeddings [25]	62.74%
ShotgunWSD 2.0	63.84%

baseline (65.20%). The only approach that surpasses the MCS baseline, yet only by a slight margin, is Babelfy [20].

J. DISCUSSION

Considering the overall results, we can conclude that ShotgunWSD 2.0 attains generally better results (sometimes up to $2 - 3\%$) than ShotgunWSD 1.0 based on sense embeddings, which in turn, is better than ShotgunWSD 1.0 based on the extended Lesk measure. On one of the data sets (SemEval 2007), all versions of ShotgunWSD yield better performance than the MCS baseline. Since a the dominant sense of a word will vary across domains and text genres, it is not a trivial task to develop an NLP approach that determines the most common sense in a given domain. The domain-independent MCS estimation approach proposed by Bhingardive *et al.* [21] provides a considerable performance gap compared to the MCS baseline, but more recent approaches [50] are able to close the gap or even surpass the MCS baseline. Moreover, Bennett *et al.* [60] show that it is possible to produce very accurate domain-neutral sense distributions reflecting usage in modern English, which allows to build a reliable domain-neutral MCS estimation. No matter how accurate MCS estimation approaches are, MCS is still a poor approach for assigning word senses, i.e. choosing the most frequent sense is not equivalent to performing actual disambiguaton. Clearly, WSD methods should outperform the basic MCS baseline to demonstrate their usefulness, but this is not easy. For instance, in the recent SemEval 2015 WSD task, the BabelNet First Sense [41], surpassed all unsupervised [44], [45] and knowledge-based [23] approaches. Therefore, some researchers in the WSD community [7] consider important even slightly outperforming the MCS baseline with an unsupervised method. In light of these comments, we consider important the fact that ShotgunWSD 2.0 surpasses the MCS baseline on SemEval 2007 and SemEval 2013. Furthermore, our algorithm compares favorably to other state-of-the-art unsupervised [15], [19], [21], [22], [43], [44] and knowledge-based [18], [23], [31], [42] WSD methods.

Regarding the performance of our algorithm, an interesting question that arises is how much does the assembly phase help. We carried out a small experiment to provide an answer to this question. We considered the ShotgunWSD 1.0 variant based on sense embeddings without changing its parameters, and we removed the assembly phase completely. Thus, the algorithm did no longer generate configurations of length greater than 8, as the parameter n is set to 8. We have evaluated this stub algorithm on SemEval 2007 and we have obtained a lower F_1 score (77.61%). We also performed a similar experiment with a stub version of ShotgunWSD 2.0, obtaining an F_1 score of 78.27%. These results indicate that the assembly phase in Algorithm 1 boosts the performance by nearly $2 - 3\%$.

It is perhaps interesting to note that we have considered an approach to combine the two semantic relatedness approaches independently used by ShotgunWSD 1.0, namely the extended Lesk measure and sense embeddings, with the goal of improving the accuracy. However, we did not observe any improvements when fusing these two measures. For this reason, we did not report any results of the combination in the paper. We also tried *GloVe* word embeddings [61] instead of *word2vec* [36], but the results were mixed (sometimes better, sometimes worse) and without any significant differences.

We performed paired McNemar's statistical significance tests [62], in order to find out if ShotgunWSD 2.0 is significantly better than the two variants of ShotgunWSD 1.0. The tests indicate that ShotgunWSD 2.0 attains significantly better results on all data sets, with respect to ShotgunWSD 1.0 based on the extended Lesk. ShotgunWSD 2.0 also attains significantly better results on three data sets (SemEval 2007, Senseval-2 and SemEval 2015), compated to the ShotgunWSD 1.0 based on sense embeddings. The tests were performed at a confidence level of 0.01. We therefore conclude that the performance improvements brought by ShotgunWSD 2.0 over its previous version are significant in most cases.

V. CONCLUSION

In this paper, we have presented a new version (2.0) of a recently introduced global WSD algorithm [25] inspired by the Shotgun genome sequencing technique [26]. Compared to other bio-inspired WSD methods [14], [15], [23], our algorithm has less parameters. Furthermore, these parameters can be intuitively tuned with respect to the WSD task. The empirical results on six evaluation benchmarks indicate that ShotgunWSD 2.0 can obtain better performance than ShotgunWSD 1.0 and other state-of-the-art unsupervised [15], [19], [21], [22], [44] and knowledge-based [18], [23], [42] WSD methods. Furthermore, our algorithm outperformed the strong Most Common Sense (MCS) baseline on two data sets.

In future work, we would like to apply our unsupervised algorithm for less studied languages, in which sense-annotated corpora are not available.

REFERENCES

[1] M. Carpuat and D. Wu, "Improving statistical machine translation using word sense disambiguation," in *Proc. EMNLP*, Jun. 2007, pp. 61–72.

[2] S. Neale, L. Gomes, E. Agirre, O. L. de Lacalle, and A. Branco, "Word sense-aware machine translation: Including senses as contextual features for improved translation models," in *Proc. LREC*, May 2016, pp. 2777–2783.
[3] X. Pu, N. Pappas, J. Henderson, and A. Popescu-Belis, "Integrating weakly supervised word sense disambiguation into neural machine translation," *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 635–649, Dec. 2018.
[4] L. Plaza, A. J. Jimeno-Yepes, A. Díaz, and A. R. Aronson, "Studying the correlation between different word sense disambiguation methods and summarization effectiveness in biomedical texts," *BMC Bioinf.*, vol. 12, Aug. 2011, Art. no. 355.
[5] A.-G. Chifu and R. T. Ionescu, "Word sense disambiguation to improve precision for ambiguous queries," *Central Eur. J. Comput. Sci.*, vol. 2, no. 4, pp. 398–411, Dec. 2012.
[6] C. Sumanth and D. Inkpen, "How much does word sense disambiguation help in sentiment analysis of micropost data?" in *Proc. WASSA*, Sep. 2015, pp. 115–121.
[7] E. Agirre and P. G. Edmonds, *Word Sense Disambiguation: Algorithms and Applications*. Amsterdam, The Netherlands: Springer, 2006.
[8] R. Navigli, "Word sense disambiguation: A survey," *ACM Comput. Surv.*, vol. 41, no. 2, Feb. 2009, Art. no. 10.
[9] F. Hristea, M. Popescu, and M. Dumitrescu, "Performing word sense disambiguation at the border between unsupervised and knowledge-based techniques," *Artif. Intell. Rev.*, vol. 30, nos. 1–4, pp. 67–86, Dec. 2008.
[10] I. Iacobacci, M. T. Pilehvar, and R. Navigli, "Embeddings for word sense disambiguation: An evaluation study," in *Proc. ACL*, Aug. 2016, pp. 897–907.
[11] A. Raganato, J. Camacho-Collados, and R. Navigli, "Word sense disambiguation: A unified evaluation framework and empirical comparison," in *Proc. EACL*, Apr. 2017, pp. 99–110.
[12] K. Taghipour and H. T. Ng, "One million sense-tagged instances for word sense disambiguation and induction," in *Proc. CoNLL*, Jul. 2015, pp. 338–344.
[13] C. D. Bovi, J. Camacho-Collados, A. Raganato, and R. Navigli, "EUROSENSE: Automatic harvesting of multilingual sense annotations from parallel text," in *Proc. ACL*, Jul. 2017, pp. 594–600.
[14] D. Schwab, J. Goulian, A. Tchechmedjiev, and H. Blanchon, "Ant colony algorithm for the unsupervised word sense disambiguation of texts: Comparison and evaluation," in *Proc. COLING*, Mumbai, India, Dec. 2012, pp. 2389–2404.
[15] D. Schwab, J. Goulian, and A. Tchechmedjiev, "Worst-case complexity and empirical evaluation of artificial intelligence methods for unsupervised word sense disambiguation," *Int. J. Eng. Technol.*, vol. 8, no. 2, pp. 124–153, Aug. 2013.
[16] D. Schwab, A. Tchechmedjiev, J. Goulian, M. Nasiruddin, G. Sérasset, and H. Blanchon, "GETALP system: Propagation of a Lesk measure through an ant colony algorithm," in *Proc. SemEval*, vol. 2, Jun. 2013, pp. 232–240.
[17] E. Agirre, O. López de Lacalle, and A. Soroa, "Random walks for knowledge-based word sense disambiguation," *Comput. Linguistics*, vol. 40, no. 1, pp. 57–84, Mar. 2014.
[18] P. Basile, A. Caputo, and G. Semeraro, "An enhanced Lesk word sense disambiguation algorithm through a distributional semantic model," in *Proc. COLING*, Aug. 2014, pp. 1591–1600.
[19] X. Chen, Z. Liu, and M. Sun, "A unified model for word sense representation and disambiguation," in *Proc. EMNLP*, Oct. 2014, pp. 1025–1035.
[20] A. Moro, A. Raganato, and R. Navigli, "Entity linking meets word sense disambiguation: A unified approach," *Trans. Assoc. Comput. Linguistics*, vol. 2, pp. 231–244, Dec. 2014.
[21] S. Bhingardive, D. Singh, V. R. Murthy, H. Redkar, and P. Bhattacharyya, "Unsupervised most frequent sense detection using word embeddings," in *Proc. NAACL*, 2015, pp. 1238–1243.
[22] R. Tripodi and M. Pelillo, "A game-theoretic approach to word sense disambiguation," *Comput. Linguistics*, vol. 43, no. 1, pp. 31–70, 2017.
[23] L. Vial, B. Lecouteux, and D. Schwab, "Sense embeddings in knowledge-based word sense disambiguation," in *Proc. IWCS*, 2017, pp. 1–8.
[24] O. Dongsuk, S. Kwon, K. Kim, and Y. Ko, "Word sense disambiguation based on word similarity calculation using word vector representation from a knowledge-based graph," in *Proc. COLING*, Aug. 2018, pp. 2704–2714.
[25] A. Butnaru, R. T. Ionescu, and F. Hristea, "ShotgunWSD: An unsupervised algorithm for global word sense disambiguation inspired by DNA sequencing," in *Proc. EACL*, Apr. 2017, pp. 916–926.
[26] S. Anderson, "Shotgun DNA sequencing using cloned DNase I-generated fragments," *Nucleic Acids Res.*, vol. 9, no. 13, pp. 3015–3027, Jul. 1981.
[27] S. Istrail *et al.*, "Whole genome shotgun assembly and comparison of human genome assemblies," *Proc. Nat. Acad. Sci. USA*, vol. 101, no. 7, pp. 1916–1921, 2004.
[28] G. A. Miller, "WordNet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995.
[29] C. Fellbaum, Ed., *WordNet: An Electronic Lexical Database*. Cambridge, MA, USA: MIT Press, 1998.
[30] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," in *Proc. SIGDOC*. New York, NY, USA, Jun. 1986, pp. 24–26.
[31] S. Banerjee and T. Pedersen, "An adapted Lesk algorithm for word sense disambiguation using wordnet," in *Proc. CICLing*. London, U.K.: Springer-Verlag, 2002, pp. 136–145.
[32] S. Banerjee and T. Pedersen, "Extended gloss overlaps as a measure of semantic relatedness," in *Proc. IJCAI*, San Francisco, CA, USA: Morgan & Kaufmann, 2003, pp. 805–810.
[33] S. Patwardhan, S. Banerjee, and T. Pedersen, "Using measures of semantic relatedness for word sense disambiguation," in *Proc. CICLing*, Berlin, Germany: Springer-Verlag, 2003, pp. 241–257.
[34] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.
[35] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. ICML*, New York, NY, USA, 2008, pp. 160–167.
[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, 2013, pp. 3111–3119.
[37] R. Navigli, K. C. Litkowski, and O. Hargraves, "SemEval-2007 task 07: Coarse-grained english all-words task," in *Proc. SemEval*, Stroudsburg, PA, USA, Jun. 2007, pp. 30–35.
[38] P. Edmonds and S. Cotton, "SENSEVAL-2: Overview," in *Proc. SENSEVAL*, Stroudsburg, PA, USA, Jul. 2001, pp. 1–5.
[39] R. Mihalcea, T. Chklovski, and A. Kilgarriff, "The senseval–3 english lexical sample task," in *Proc. SENSEVAL*, Stroudsburg, PA, USA, Jul. 2004, pp. 25–28.
[40] R. Navigli, D. Jurgens, and D. Vannella, "SemEval-2013 task 12: Multilingual word sense disambiguation," in *Proc. SemEval*, Jun. 2013, pp. 222–231.
[41] A. Moro and R. Navigli, "Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking," in *Proc. SemEval*, Jun. 2015, pp. 288–297.
[42] S. Torres and A. Gelbukh, "Comparing similarity measures for original WSD Lesk algorithm," *Res. Comput. Sci.*, vol. 43, no. 1, pp. 155–166, 2009.
[43] Y. Gutiérrez, Y. Castaneda, A. González, R. Estrada, D. D. Piug, J. I. Abreu, R. Pérez, A. F. Orquín, A. Montoyo, R. Muñoz, and F. Camara, "UMCC_DLSI: Reinforcing a ranking algorithm with sense frequencies and multidimensional semantic resources to solve multilingual word sense disambiguation," in *Proc. SemEval*, Jun. 2013, pp. 241–249.
[44] S. L. Manion, "SUDOKU: Treating word sense disambiguation & entity linking as a deterministic problem—Via an unsupervised & iterative approach," in *Proc. SemEval*, Jun. 2015, pp. 365–369.
[45] M. Apidianaki and L. Gong, "LIMSI: Translations as source of indirect supervision for multilingual all-words sense disambiguation and entity linking," in *Proc. SemEval*, Jun. 2015, pp. 298–302.
[46] R. V. V. Bhala and S. Abirami, "Trends in word sense disambiguation," *Artif. Intell. Rev.*, vol. 42, no. 2, pp. 159–171, Aug. 2014.
[47] E. Agirre and A. Soroa, "Personalizing pagerank for word sense disambiguation," in *Proc. EACL*, 2009, pp. 33–41.
[48] K.-H. Nguyen and C.-Y. Ock, "Word sense disambiguation as a traveling salesman problem," *Artif. Intell. Rev.*, vol. 40, no. 4, pp. 405–427, Dec. 2013.
[49] A. Panchenko, E. Ruppert, S. Faralli, S. P. Ponzetto, and C. Biemann, "Unsupervised does not mean uninterpretable: The case for word sense induction and disambiguation," in *Proc. EACL*. Valencia, Spain, 2017, pp. 86–98.
[50] T. Pasini and R. Navigli, "Two knowledge-based methods for high-performance sense distribution learning," in *Proc. AAAI*, Apr. 2018, pp. 5374–5381.
[51] D. Yuan, J. Richardson, R. Doherty, C. Evans, and E. Altendorf, "Semi-supervised word sense disambiguation with neural models," in *Proc. COLING*, 2016, pps. 1374–1385.
[52] A. Raganato, C. D. Bovi, and R. Navigli, "Neural sequence learning models for word sense disambiguation," in *Proc. EMNLP*, Sep. 2017, pp. 1156–1167.

[53] K. V. Voelkerding, S. A. Dames, and J. D. Durtschi, "Next-generation sequencing: From basic research to diagnostics," *Clin. Chem.*, vol. 55, no. 4, pp. 41–47, Apr. 2009.
[54] S. Bennett, "Solexa Ltd," *Pharmacogenomics*, vol. 5, no. 4, pp. 433–438, Jun. 2004.
[55] R. K. Patel and M. Jain, "NGS QC toolkit: A toolkit for quality control of next generation sequencing data," *PLoS ONE*, vol. 7, no. 2, Feb. 2012, Art. no. e30619.
[56] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys. Doklady*, vol. 10, no. 8, pp. 707–710, 1966.
[57] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 40, no. 3, pp. 211–218, 2006.
[58] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. SODA*, Philadelphia, PA, USA, Jan. 2007, pp. 1027–1035.
[59] R. T. Ionescu, S. Smeureanu, M. Popescu, and B. Alexe, "Detecting abnormal events in video using narrowed normality clusters," in *Proc. WACV*, Jan. 2019, pp. 1951–1960.
[60] A. Bennett, T. Baldwin, J. H. Lau, D. McCarthy, and F. Bond, "LexSemTm: A semantic dataset based on all-words unsupervised sense distribution learning," in *Proc. ACL*, Aug. 2016, pp. 1513–1524.
[61] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. EMNLP*, Oct. 2014, pp. 1532–1543.
[62] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, 1998.

ANDREI M. BUTNARU graduated from the Faculty of Mathematics and Computer Science, University of Bucharest, Romania, in 2014. He received the M.Sc. degree in artificial intelligence from the University of Bucharest, in 2016, where he is currently pursuing the Ph.D. degree. He published several articles at international peer-reviewed conferences, including top-tier conferences, such as ACL, EMNLP, NAACL, and EACL. His main research interests include machine learning, text mining, and computational linguistics. Together with other coauthors, he participated at the Arabic Dialect Identification Shared Task of the VarDial Workshop of EACL 2017, ranking on the first place, and at the Arabic Dialect Identification Shared Task of the VarDial Workshop of COLING 2018, ranking once again on the first place.

RADU TUDOR IONESCU received the Ph.D. degree from the University of Bucharest, Romania, in 2013, where he is currently an Associate Professor. He has published over 60 articles at international peer-reviewed venues (e.g., CVPR, ICCV, ACL, and EMNLP) and a research monograph with Springer. His research interests include machine learning, computer vision, image processing, text mining, and computational biology. He received the Caianiello Best Young Paper Award at ICIAP 2013 for the article *Kernels for Visual Words Histograms*. He received the 2014 Award for Outstanding Doctoral Research in the field of computer science from the Romanian Ad Astra Association. He also received the Young Researchers in Science and Engineering Prize organized by Prof. Rada Mihalcea for young Romanian researchers in all scientific fields. He participated at several international competitions obtaining top ranks: the fourth place in the Facial Expression Recognition Challenge of the WREPL Workshop of ICML 2013, the third place in the Native Language Identification Shared Task of the BEA-8 Workshop of NAACL 2013, the second place in the Arabic Dialect Identification Shared Task of the VarDial Workshop of COLING 2016, the first place in the Arabic Dialect Identification Shared Task of the VarDial Workshop of EACL 2017, the first place in the Native Language Identification Shared Task of the BEA-12 Workshop of EMNLP 2017, and the first place in the Arabic Dialect Identification Shared Task of the VarDial Workshop of COLING 2018.