Artificial Intelligence 279 (2020) 103215

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Train-O-Matic: Supervised Word Sense Disambiguation with no (manual) effort

Tommaso Pasini *, Roberto Navigli

Department of Computer Science, Sapienza University of Rome, Italy

ARTICLE INFO

Article history:
Received 24 November 2018
Received in revised form 11 November 2019
Accepted 15 November 2019
Available online 26 November 2019

Keywords:
Word Sense Disambiguation
Corpus Generation
Word Sense Distribution learning
Multilinguality

ABSTRACT

Word Sense Disambiguation (WSD) is the task of associating the correct meaning with a word in a given context. WSD provides explicit semantic information that is beneficial to several downstream applications, such as question answering, semantic parsing and hypernym extraction. Unfortunately, WSD suffers from the well-known knowledge acquisition bottleneck problem: it is very expensive, in terms of both time and money, to acquire semantic annotations for a large number of sentences. To address this blocking issue we present Train-O-Matic, a knowledge-based and language-independent approach that is able to provide millions of training instances annotated automatically with word meanings. The approach is fully automatic, i.e., no human intervention is required, and the only type of human knowledge used is a task-independent WordNet-like resource. Moreover, as the sense distribution in the training set is pivotal to boosting the performance of WSD systems, we also present two unsupervised and language-independent methods that automatically induce a sense distribution when given a simple corpus of sentences. We show that, when the learned distributions are taken into account for generating the training sets, the performance of supervised methods is further enhanced. Experiments have proven that Train-O-Matic on its own, and also coupled with word sense distribution learning methods, lead a supervised system to achieve state-of-the-art performance consistently across gold standard datasets and languages. Importantly, we show how our sense distribution learning techniques aid Train-O-Matic to scale well over domains, without any extra human effort. To encourage future research, we release all the training sets in 5 different languages and the sense distributions for each domain of SemEval-13 and SemEval-15 at http://trainomatic.org.

1. Introduction

Word Sense Disambiguation is the task of associating a meaning, selected from a fixed set of concepts, with a word in a given context [1]. It is a key task in both computational lexical semantics and in artificial intelligence more broadly [2], inasmuch as it addresses the lexical ambiguity of text by making the meaning of words occurring in a given context explicit. Indeed, removing the ambiguity of words in a text can be beneficial to a variety of different NLP applications, such as: machine translation, where the meaning information for the target word can be helpful to choose the correct translation; hypernym extraction, which requires a disambiguation step in order to pick the right hypernym of a word which otherwise

* Corresponding author.
E-mail addresses: pasini@di.uniroma1.it (T. Pasini), navigli@di.uniroma1.it (R. Navigli).

https://doi.org/10.1016/j.artint.2019.103215

might have different hypernyms depending on the context; semantic search, which, in contrast to lexical search, takes advantage of synonymy and polysemy to provide more accurate results.

The NLP community conducts two main lines of research in WSD: knowledge-based and supervised WSD. Knowledge-based approaches exploit the structural information contained in a general-purpose lexical semantic knowledge resource like WordNet [3] to identify the correct sense of a target word in a context, i.e., a sentence or a text. Knowledge resources are usually structured as a graph, where the nodes represent the concepts and the edges the relations among those concepts. Taking advantage of the graph structure, different graph algorithms have been developed to disambiguate a target word in a sentence or document. For example, a PageRank-based algorithm can be used to determine the probability of reaching any sense in the graph starting from the senses of other words in the context. Recent approaches of this kind have been shown to obtain competitive results [4,5], which can be further improved when a large set of syntagmatic relations is added to the reference knowledge base [6].

Tripodi and Pelillo [7], instead, formulated the problem as a game where words are the players and they have to choose their best sense in terms of similarity with the senses selected by the other words in the context. This formulation still exploits a knowledge resource in order to identify the set of concepts for each word and to compute the similarity between senses. The most recent effort in terms of knowledge-based WSD is [8] which exploits topic modeling techniques, i.e., Latent Dirichlet Allocation, and the whole document in order to identify the sense for a target word. The main inherent limits of knowledge-based approaches arise from the fact that they tend to adopt a bag-of-word approach hence discarding the word ordering information and giving the same importance to all the tokens regardless of their distance from the target word; moreover, they do not include function words.

In contrast, supervised WSD frames the problem as a classical machine learning task [9–11], in which, first, a model for each word is trained from sentences annotated with word senses, and, second, these models are then used to disambiguate each target word in previously-unseen sentences separately. Raganato et al. [12], instead, used a single sequence-to-sequence model for all the words, which, given a sequence of words, outputs a sequence of words and senses.

Supervised WSD has an additional difficulty with respect to other classification problems as the classes to choose from (i.e., the senses of the target words) vary for each word, therefore requiring either a separate training process to be performed on a word by word basis, or, alternatively, the handling of a very large output vocabulary. As a result, hundreds of training instances are needed for each ambiguous word in a language's lexicon. This would necessitate a million-item training set to be manually created for each language of interest, an endeavour that is currently beyond reach even for resource-rich languages like English. Another issue that arises when a supervised approach is chosen is the balancing of the training data. It is common, in fact, for machine learning algorithms to overfit towards the most frequent class in the training set (i.e. the Most Frequent Sense (MFS) of a word in our case). This is even more exacerbated in WSD, as word senses follow a Zipfian distribution, i.e., the least frequent senses appear exponentially fewer times than the most frequent ones. Pilehvar and Navigli [13] and Postma et al. [14], indeed, proved that knowing the sense distribution on the test set would increase the performance of supervised systems by more than 10 points.

In this paper we investigate, firstly, how to put knowledge bases at the service of supervised systems in order to reduce the knowledge acquisition bottleneck, and, secondly, how to perform on-the-fly domain adaptation by learning the word sense distribution of the text to disambiguate. In this context we present Train-O-Matic [15,16], a language-independent method for generating huge high-quality training sets for all the words in a language's vocabulary. We provide an in-depth extrinsic evaluation of the method, testing a WSD system trained on our automatically generated data in multiple languages. We also extend our approach by proposing a semi-supervised neural variant for estimating the probability of a sense appearing in a sentence. Furthermore, given the high impact on performance of the number of examples assigned to each sense [14], we also present two knowledge-based methods for learning the distribution of word senses given a raw corpus of sentences [17]. Our experiments show that Train-O-Matic is able to generate high-quality data for all the languages we tested, and, when coupled with word sense distribution learning, can adapt the generated training set to the target domain autonomously. This paper extends [15] and [17] by providing the following novel contributions:

- A semi-supervised neural approach to compute the probability of a synset appearing in a given sentence.
- A method for the automatic domain adaptation of the training sets generated by Train-O-Matic.
- Extensive experiments on the influence of word sense distribution in the training set on supervised systems, where, thanks to the automatic balancing of training sets and ad hoc baselines for computing the most frequent sense of a word, we obtain state-of-the-art results on domain-specific experiments.
- In-depth extrinsic evaluation of all the proposed approaches on multiple languages that proves, for the first time, how effective the learned MFS baselines are and how they can be used to improve state-of-the-art performance on both multiple-language and domain-specific evaluation.

2. Train-O-Matic: building a training set from scratch

In this Section we present Train-O-Matic [15,16], a language-independent approach to the automatic construction of a sense-tagged training set. Train-O-Matic takes as input a corpus C (e.g., Wikipedia) and a semantic network $G = (V, E)$ where V is the set of concepts (i.e., synsets) and E the set of edges which represent the semantic relations between nodes.

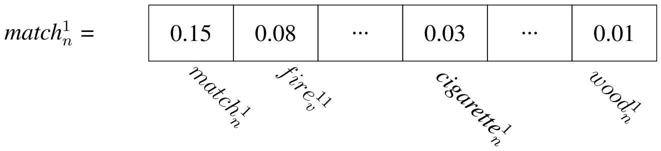

Fig. 1. Personalised PageRank vector for the lighter sense of *match*.

We also define the function $Senses(w)$ that, given a word w, returns the set of vertices in V that contain w among their senses, e.g., given the word *match*, $Senses(match)$ returns all the synsets in G that contain *match* among their senses.

Train-O-Matic consists of three steps:

- **Lexical profiling:** for each vertex (i.e., synset) in the semantic network, we compute its Personalised PageRank vector, which provides its lexical-semantic profile (Section 2.1), i.e., a vector where all vertices in the network appear with a score expressing their probability of being reached if we start surfing the graph from the starting node.
- **Sentence scoring:** given a target word and the list of sentences it appears in, we compute a distribution over its senses.
- **Sentence ranking and selection:** Train-O-Matic ranks each sentence σ for a sense s of a word $w \in$[1] σ according to the difference between the first and the second sense in the distribution we computed at the previous step (Section 2.3).

2.1. Lexical profiling

The first step aims at computing, for each node in the semantic network, a probability distribution over all the nodes in the graph conditioned to a single given synset. Such distribution can be interpreted as a relatedness measure between the starting synset s and all the nodes s' in the graph (including s itself). Thus, for each vertex v, we define its lexical profile over the graph $G = (V, E)$ as the output of the Personalised PageRank (PPR) algorithm. The personalised variant of PageRank [18] collapses the restart probability on one vertex only, which is equivalent to performing random walks restarting always on the same node thereby concentrating more probability mass in the surroundings of the starting vertex neighbours.

Formally, (P)PR is computed as follows:

$$v^{(t+1)} = (1-\alpha)v^{(0)} + \alpha M v^{(t)} \tag{1}$$

where M is the row-normalized adjacency matrix of the semantic network, the restart probability distribution is encoded by the column vector $v^{(0)}$, $v^{(t)}$ is the column vector with the probabilities of each node at time step t, and α is the well-known damping factor usually set to 0.85 [18]. If we set $v^{(0)}$ to a unit probability vector $(0, \dots, 0, 1, 0, \dots, 0)$, i.e., restart is always on a given vertex, PPR outputs the probability of reaching every vertex starting from the restart vertex after a random number of steps. This approach has been used in the literature to create semantic signatures (i.e., profiles) of individual concepts, i.e., vertices of the semantic network [19], and then to determine the semantic similarity of concepts. Instead we use the PPR vector as an estimate of the conditional probability of a word w' given the target sense[2] $s \in V$ of word w as was also done by Pilehvar and Collier [20]:

$$P(w'|s, w) = \frac{\max_{s' \in Senses(w')} v_s(s')}{Z} \tag{2}$$

where $Z = \sum_{w''} P(w''|s, w)$ is a normalization constant, v_s is the resulting PPR vector with restart probability on node s, and $v_s(s')$ is the vector component corresponding to sense s'. We prefer the *max* operator instead of *sum* as we are more interested in capturing the maximum possible relatedness between s and any of the senses of w', rather than the relatedness between s and the word w intended as the sum of its meanings. This will be further clarified in Section 2.2.2, where we explain how we compute the probability of a sense appearing in a sentence.

To explain this step with an example, given the vector v of the lighter sense of *match* s in Fig. 1,[3] we compute the probability of the word *cigarette* given the sense s and the target word *match* by picking the highest score of a synset in v with *cigarette* among its senses (i.e., cigarette_n^1). Thus $P(cigarette \,|\, \text{match}_n^1, match) = 0.03$. To fix the number of iterations needed to have a sufficiently accurate vector, we follow Lofgren et al. [21] and set the number of iterations to 100,000. As a result of this lexical profiling step we have a probability distribution over the knowledge base concepts for each given word sense of interest.

[1] Here and across the paper we make use of the $\in$ symbol to express the appearance of a word in a sentence even if σ is not a set but a list of words.

[2] Note that we use senses and concepts (synsets) interchangeably, because – given a word – a word sense unambiguously determines a concept (i.e., the synset it is contained in) and vice versa.

[3] We use the notation introduced by Navigli [1] where w_p^k indicates the k-th sense of the word w with POS p.

2.2. Sentence scoring

The objective of the second step is to score the importance of word senses for each of the corpus sentences which contain the word of interest. In other words, we want to assign, to each word of interest and for all the sentences in the corpus where it appears, a distribution over its senses.

We present two different ways for computing the probability of a sense s to appear in a given sentence σ containing the target word w. The first one is graph-based and relies on the lexical profile we described in the previous Section, the second one, instead, is semi-supervised and exploits word embeddings to represent synsets and sentences in the same vector space, and a sense-annotated corpus such as SemCor to compute the sense embeddings.

2.2.1. Knowledge-based scoring

The first method is graph-based and exploits a knowledge base in order to compute the probability of a synset s to appear in a sentence σ. Given a sentence $\sigma = w_1, w_2, \dots, w_n$, and a target word w in the sentence ($w \in \sigma$), for each of its senses $s \in Senses(w)$, we compute the probability $P(s|\sigma, w)$. Thanks to Bayes' theorem we can determine the probability of sense s of w given the sentence as follows:

$$\begin{aligned} P(s|\sigma, w) &= \frac{P(\sigma|s, w)P(s|w)}{P(\sigma|w)} && (3)\\ &= \frac{P(w_1, \dots, w_n|s, w)P(s|w)}{P(w_1, \dots, w_n|w)} \\ &\propto P(w_1, \dots, w_n|s, w)P(s|w) && (4)\\ &\approx P(w_1|s, w)\dots P(w_n|s, w)P(s|w) && (5)\end{aligned}$$

where Formula (4) is proportional to the original probability (due to removing the constant in the denominator) and is approximated by Formula (5) due to the assumption of independence of the words in the sentence. $P(w_i|s, w)$ is calculated as in Formula (2) and $P(s|w)$ is set to $1/|Senses(w)|$ (recall that s is a sense of w). Note that, since we used the *max* operator in Formula (2), we estimate the probability of a sense s according to the senses in the sentence that are most related to it, rather than to the words – which would have been the case if we had used the *sum* operator in Equation (2).

To give an example of how this procedure works, let the sentence $\sigma =$ "A match is a tool for starting a fire", the target word $w = match$ and its set of senses $S_{match} = \{s^1_{match}, s^2_{match}\}$ be the input where s^1_{match} is the *lighter* sense and s^2_{match} is the *game* sense of *match*. We want to calculate the probability of each $s^i_{match} \in S_{match}$ being the correct sense of *match* in the sentence σ. Therefore, following Formula (5) we have:

$$\begin{aligned} P(s^1_{match}|\sigma, match) &\approx P(\text{match}|s^1_{match}, match)P(\text{tool}|s^1_{match}, match) \cdot P(\text{start}|s^1_{match}, match)\\ &\cdot P(\text{fire}|s^1_{match}, match) \cdot P(s^1_{match}|match)\\ &= (1.0) \cdot (2.1 \cdot 10^{-4}) \cdot (2 \cdot 10^{-3}) \cdot (10^{-2}) \cdot (5 \cdot 10^{-1})\\ &= 2.1 \cdot 10^{-9}\\ P(s^2_{match}|\sigma, match) &\approx P(\text{match}|s^2_{match}, match) \cdot P(\text{tool}|s^2_{match}, match) \cdot P(\text{start}|s^2_{match}, match)\\ &\cdot P(\text{fire}|s^2_{match}, match) \cdot P(s^2_{match}|match)\\ &= (1.0) \cdot (10^{-5}) \cdot (2.9 \cdot 10^{-4}) \cdot (10^{-6}) \cdot (5 \cdot 10^{-1})\\ &= 1.45 \cdot 10^{-15}\end{aligned}$$

As can be seen, the first sense of *match* has a much higher probability, due to its stronger relatedness to the other words in the context (i.e., *start*, *fire* and *tool*). Note also that all the probabilities for the second sense are at least one magnitude less than the probabilities of the first one.

2.2.2. Sentence scoring with word embeddings

The second approach is semi-supervised and exploits the lexical context of synsets in an annotated corpus (i.e. SemCor) to compute a latent representation for each synset. This vector is computed as the average of the word vectors across all sentences in the corpus where the synset appears. Thus, for each sentence σ containing a given synset s, we first compute a single contextual vector of s by averaging the embeddings of the words surrounding it in σ. Then, we compute a single latent representation for the target synset s by averaging all its contextualised representations.

More formally, given a synset s and a set of sentences ξ where s appears as the sense of the word w, we compute the vector for s as follows:

$$v(s,w) = \frac{1}{|\xi|}\sum_{\sigma\in\xi} vec(\sigma, w, W) \tag{6}$$

$$vec(\sigma, w, W) = \frac{1}{2W+1}\sum_{\substack{i=w.index-W;\\ i\neq w.index}}^{w.index+W} word2vec(\sigma[i]) \tag{7}$$

where the function $vec(\sigma, w, W)$ is the average of the word vectors surrounding the word with sense s in a given window W, and $word2vec(w)$ returns the vector representation of the word w. Note that we can now compare word senses and sentences as they all lie in the same vector space defined by the word vectors. Thanks to this, given a new sentence and a target word we can compute the sentence vector in the same way as we did for synsets (i.e., by averaging the word vectors in the surroundings of the target word). Then, we compute the probability distribution over the senses of a target word in a sentence by assigning a score defined by the cosine similarity between the sentence and the senses' vectors and renormalising them to form a distribution. More formally we compute the probability of a sense s of a word w to appear in the sentence σ as follows:

$$P(s|\sigma, w) = \frac{e^{sim(vec(\sigma,w,W),v(s,w))}}{\sum\limits_{s'\in senses(w)} e^{sim(vec(\sigma,w,W),v(s',w))}}$$

i.e., the softmax over the possible senses s' of the word w where $vec(\sigma, w, W)$ and $v(s, w)$ are the two functions we introduced in Formula (6) and (7), and $sim(x, y)$ is the cosine similarity between the vectors x and y. We choose the *softmax* since it is the most obvious way for normalising and building a valid probability distribution from values that range in $[-1, 1]$.

2.3. Sentence ranking and selection

Finally, for a given word w and a given sense $s_1 \in Senses(w)$, we score each sentence σ in which w appears and s_1 is its most likely sense according to a formula that takes into account the difference between the first (i.e., s_1) and the second most likely sense of w in σ:

$$\Delta_{s_1}(\sigma) = P(s_1|\sigma, w) - P(s_2|\sigma, w) \tag{8}$$

where $s_1 = \arg\max_{s\in Senses(w)} P(s|\sigma, w)$, and $s_2 = \arg\max_{s\in Senses(w)\setminus\{s_1\}} P(s|\sigma, w)$. We then sort all sentences based on $\Delta_{s_1}(\cdot)$ and return a ranked list of sentences where word w is most likely to be sense-annotated with s_1.

3. Creating a denser and multilingual semantic network

The approach we have presented so far depends highly on the underlying semantic network to compute the probability of a sense given a sentence and a target word (see Formula (5)). Therefore, while in Section 2 we employed WordNet as the knowledge graph, we note that it lacks syntagmatic relations that connect verbs to their arguments (e.g. run_v and $program_n$) and pairs of nominal synsets that form multi-word expressions (e.g. bus_n and $driver_n$). Indeed, WordNet nodes are mainly connected via paradigmatic relations such as hypernymy (is-a) and meronymy (part-of). Thus, due to the lack of those relations between synsets, if WordNet is used, Train-O-Matic would not be able to find a good estimation of the probability of a synset given a word and a sentence, as it depends directly on the PPR – and thus on the relations within the graph – of the synset itself (see Formula (2) and Formula (5)). Moreover, our approach is not tied to any specific language, whereas WordNet, instead, covers only English. In order to overcome these limitations, we exploit BabelNet[4] [22], the largest multilingual resource that includes many lexical resources such as WordNet, Wikipedia, Wiktionary, etc. Thanks to its mapping between different resources, we are able to extract a subgraph of BabelNet induced by WordNet vertices that also includes all the syntagmatic relations comprised in BabelNet.

However, as most relations come from Wikipedia (e.g., the English Wikipedia *Bus* article[5] is linked to *Passenger*, *Tourism*, *Bus lane*, *Timetable*, and many more), but not all Wikipedia pages are connected with the same degree of relatedness (e.g. countries are often linked to many pages, but they are not necessarily closely related to them), we weight each edge $(s, s') \in E$ in our WordNet-induced subgraph of BabelNet $G = (V, E)$ as follows:

$$w(s,s') = \begin{cases} 1 & (s,s') \in E(\text{WordNet}) \\ WO(s,s') & \text{otherwise} \end{cases} \tag{9}$$

[4] http://babelnet.org.

[5] Retrieved on June 14th, 2019.

Table 1
Top-ranking synsets of the PPR vectors computed on WordNet (first and third columns) and WordNet$_{BN}$ (second and fourth columns) for *mouse* as animal (left) and as device (right).

mouse (animal)		mouse (device)	
WordNet	WordNet$_{BN}$	WordNet	WordNet$_{BN}$
mouse^1_n	mouse^1_n	mouse^4_n	mouse^4_n
tail^1_n	little^1_a	wheel^1_n	computer^1_n
hairless^1_a	rodent^1_n	electronic_device^1_n	pad^4_n
rodent^1_n	cheese^1_n	ball^3_n	cursor^1_n
trunk^3_n	cat^1_n	hand_operated^1_n	operating_system^1_n
elongate^2_a	rat^1_n	mouse_button^1_n	trackball^1_n
house_mouse^1_n	elephant^1_n	cursor^1_n	wheel^1_n
minuteness^1_n	pet^1_n	operate^3_v	joystick^1_n
nude_mouse^1_n	experiment^1_n	object^1_n	Windows^1_n

where $E(\text{WordNet})$ is the edge set of the original WordNet graph and $WO(s, s')$ is the weighted overlap measure which calculates the similarity between two synsets:

$$WO(s, s') = \frac{\sum_{i=1}^{|S|} (r_i + r'_i)^{-1}}{\sum_{i=1}^{|S|} (2i)^{-1}}$$

where r_i and r'_i are the ranks of the i-th synsets in the set S of the components in common between the vectors associated with s and s', respectively. At this stage we have still not computed the lexical profiles for each synset, thus, in order to apply the edge weighting strategy introduced above, we exploit the pre-computed NASARI vectors [23,24] to compute the Weighted Overlap. This latter is chosen as distance measure between two sparse vector representations, since we are more interested in comparing the synsets' ranks rather than their values therein. Therefore, the Weighted Overlap better suits our purpose than the cosine similarity, moreover, it has proved to achieve higher performance when vectors have explicit dimensions [19]. At this point, to apply Formula (1) and compute PPR according to the weights we have just computed, we replace the unweighted adjacency matrix M of G with the row-normalised one containing the weights computed in Formula (9). To qualitatively corroborate our hunch that syntagmatic relations are essential to compute more meaningful and useful probabilities (see Formula (2)) we report in Table 1 the highest-probability synsets in the lexical profiles calculated for two different senses of *mouse* (its animal and device senses) when WordNet (left) and our WordNet-induced subgraph of BabelNet (WordNet$_{BN}$, right) are used as the semantic network during the PPR computation. Note that, when the plain WordNet is used, the top-ranked synsets are connected to the starting synset via paradigmatic relations, while the WordNet$_{BN}$ PPR version is richer and also contains related synsets such as *computer*^1_n for *mouse (device)* or *cheese*^1_n for *mouse (animal)*.

4. Balancing the training set

How to balance the training set in order to avoid overfit on the most frequent class is a well-known problem in Machine Learning. Different techniques have been developed in order balance the examples in the training set (up- or down-sampling), or to force the trained model not to overfit (regularization, early stopping, dropout etc.). This problem becomes even more critical in WSD where word senses (and language in general) follow Zipf's distribution, hence, it is very common to have many examples for the most frequent sense of a word, but exponentially fewer examples for the other senses. The scenario is even worse when we consider disambiguating texts that lie in a specific domain where the predominant sense is no longer the one we might expect for that word given a more general domain setting. For example, consider the word *plane*, which can usually be assumed to denote the sense of airplane; if we consider a math textbook, we are much more likely to find the word *plane* in its geometrical sense. Thus, following Pilehvar and Navigli [13] and Postma et al. [14] we note that the balance of the training data plays a role of fundamental importance in leading a supervised model to generalize more effectively not only on the most frequent classes, but also on the least frequent ones. In fact, Postma et al. [14] estimate that providing a number of examples for a sense that is proportional to its frequency in the test set would dramatically increase the performance by more than 10 points.

Thus, while a lot of effort in the past years has been devoted to developing deep neural networks for improving the state of the art on the SemEval benchmarks, few works only have focused on injecting different types of knowledge, i.e., the word sense distribution, into the training data in order to reflect the sense distribution of the target text. Even if it is not always possible to know the sense distribution of the target word, it can still be estimated in various ways: Bennett et al. [25] propose a method based on topic modeling to predict the distribution of senses in a given corpus, while Pasini and Navigli [17] propose two language-independent methods to estimate such distributions. Given our interest in scaling over multiple languages, we are now going to present two methods that we previously introduced in [17]. EnDi (Entropy based Distribution Learning) and DaD (Domain aware Distribution learning) are two methods that learn the word sense distribution given an input corpus of raw sentences. In order to assign to each word of interest w a sentence-level

Table 2
A sense distribution computation example for the word *plane*.

Sentence	plane_n^1 (aircraft)	plane_n^2 (geometry)	plane_n^5 (carpentry)
Two people on the **plane** died.	0.92	0.01	0.07
The flight was delayed due to trouble with the **plane**.	0.82	0.07	0.11
Only one **plane** landed successfully.	0.73	0.10	0.17
The cabinetmaker used a **plane** for the finish work.	0.20	0.18	0.62
A catalog of special **plane** curves.	0.10	0.85	0.05
$\mathcal{D}_{plane}$	0.55	0.24	0.21

sense distribution γ_w^σ, both methods apply the Sentence Scoring phase of Train-O-Matic (which in what follows we call *Train-O-Matic-1*) to all the sentences in the input raw corpus.

Note that at this stage we have a different sense distribution for each target word and for each sentence it appears in that is computed as explained in Section 2.2.2; now, instead, our aim is to build a single sense distribution for each different lexeme (i.e. lemma and part of speech). To build such distributions EnDi applies an entropy-based filter on the sentence-level sense distributions in order to filter out all those that have a uniform shape, and it then averages the remaining ones. DaD, on the other hand, exploits a graph-based algorithm and a mapping between knowledge graph nodes and domain labels. It first estimates the distribution of the domains in the input corpus and later propagates their probabilities over the graph synsets.

4.1. Entropy-Based Distribution Learning (EnDi)

The first method takes as input a lexicon $\mathcal{L}$ containing all the words for which we need to compute the sense distributions, the output of *Train-O-Matic-1*, i.e., the sets of sense distributions at sentence level $\Gamma_w = \{\gamma_w^\sigma : w \in \sigma\}$ for each word $w \in \mathcal{L}$, and a threshold θ that is used to filter out the γ_w^σ that are too entropic. The objective of this method is to assign only one sense distribution to each word in the lexicon by averaging the most skewed sentence-level sense distributions that have been computed by *Train-O-Matic-1*. In order to assess how skewed a distribution is we apply the entropy measure and compare it with the threshold θ. We consider skewed all those distributions that have an entropy smaller than θ as follows:

$$\hat{\Gamma}_w = \{\gamma_w^\sigma \in \Gamma_w : \mathcal{H}(\gamma_w^\sigma) \leq \theta\} \tag{10}$$

where γ_w^σ is the distribution over w's senses in the sentence σ and $\mathcal{H}(\gamma)$ is the entropy of the input distribution γ defined as:

$$\mathcal{H}(\gamma) = -\sum_{s\in\gamma} \gamma(s)\, log_2\, \gamma(s)$$

As a result $\hat{\Gamma}_w$ is a filtered version of Γ_w where the least skewed distributions were pruned out, i.e., it contains only distributions that show a clear predominance of a sense. Note that, to make the entropies of all the distributions comparable with each other, so to allow us to define a single threshold θ that is valid for all the distributions, we add n new components with a value of 0 to all those distributions that have only j meanings, where $n = m - j$ and m is the maximum number of possible meanings that a word $w \in \mathcal{L}$ has. At this point all the distributions comprise the same number of elements and we smooth them by adding to each component a small smoothing factor $\lambda = 10^{-3}$ and by normalising them by the sum of their values.

Finally, the unified probability mass function $\mathcal{D}_w$ for a word w is computed so as to have, for each sense s of w, the following value:

$$\mathcal{D}_w(s) = \frac{1}{|\hat{\Gamma}_w|} \sum_{\gamma_w^\sigma \in \hat{\Gamma}_w} \gamma_w^\sigma(s) \tag{11}$$

For example, let us consider the word *plane* and 5 sentences that contain it (see Table 2, leftmost column): we compute its sense distribution by summing the probability of each sense across the sentences and then renormalising the results by their sum (last row of the Table).

4.2. Domain-Aware Distribution Learning (DaD)

The second method for sense distribution learning also takes as input the lexicon $\mathcal{L}$, the set of sentence-level sense distributions from *Train-O-Matic-1* $\Gamma_w = \{\gamma_w^\sigma : w \in \sigma\}$ for each word $w \in \mathcal{L}$ and a semantic network $\mathcal{G} = (V, E)$. In addition, for this approach we assume a mapping between synsets in the semantic network and domain labels from a fixed set

$\mathcal{S}$[6](we exploit the work by [26] that maps each synset in BabelNet to zero, one or more domains). For example, the synset airplane^1_n is mapped to the TRANSPORT & TRAVEL domain only, while the synset parameter^3_n (parameter of a function) is mapped to two different domains: MATHS and COMPUTING.

The approach conflates the sparse information of the sentence-level sense distributions (i.e., Γ_w) in the input corpus into a more informative distribution over the domains. This is to say that, knowing that $\gamma^\sigma_w \in \Gamma_w$ are not 100% accurate, we leverage the huge amount of data *Train-O-Matic-1* produces and the synset-domain mapping from the knowledge base to merge the many different sense distributions at sentence level into a single distribution over the domains of the knowledge base.

To summarise, we learn the sense distributions by applying the following steps:

1. **Domain distribution:** we compute the distribution of the domains in the sentence-level sense distributions (i.e., Γ_w).
2. **Graph augmentation:** we augment the semantic network with domain nodes and connect them to the synsets in the semantic network they are associated with.
3. **Domain propagation:** we run Personalised PageRank on the augmented semantic graph and obtain a sense distribution over all the synsets in the graph.

Domain distribution. Given the set Γ_w of the sentence-level sense distributions extracted by *Train-O-Matic-1*, we estimate the probability $P(X = d)$ of a domain d as follows:

$$P(X = d) = \frac{e^{\mathcal{C}[d]}}{\sum_{d' \in \mathcal{S}} e^{\mathcal{C}[d]}} \tag{12}$$

$$\mathcal{C}[d] = \sum_{\gamma \in \Gamma} \sum_{\substack{s \in \gamma \ | \\ d \in domains(s)}} \gamma(s) \tag{13}$$

where *domains*(s) is the set of domains in BabelNet s is associated with, Γ is the set of all sentence-level distributions γ extracted by *Train-O-Matic-1* and $\mathcal{S}$ is the set of domains in BabelNet. In other words, the probability of domain d is the normalised score of the sum of all the values of all the synsets that belong to the domain d and that appear in a sentence-level distribution γ. The idea behind this approach is that, if a document belongs to a domain, then more synsets in that domain will appear, thus increasing its probability mass. Moreover, using a coarser sense inventory (i.e. domains) we conflate multiple senses to one domain that will benefit from the statistical information coming from all of the conflated senses.

Graph augmentation. Once we have a domain distribution for the input corpus, for each domain we add a new node to the semantic network and draw an edge between the domain node and all the synsets it is mapped to (note that a synset might be connected to 0, 1 or more domains).

Domain propagation. Once we have built the augmented semantic network, we are ready to apply PageRank to the graph by modifying the restart probability in order to reflect the probability distribution over the domains that we have previously computed with Formula (12).

Using again the analogy with the random walker, every time the walker decides to restart its walk, it will choose a domain according to its probability. The result of this step is a distribution over all the nodes in the semantic graph that is conditioned on the domain distributions, namely, senses connected to domains with a high restart probability will receive a high probability as well. In order to build the sense distribution given a lexeme, we retrieve all its senses in the semantic network together with their associated probabilities and normalise these to form a distribution.

We take advantage of these methods (EnDi and DaD) for two main purposes: 1) to obtain an estimation for the most common sense baseline that also reflects the potential domain bias of the target text, and 2) to build the training set by leveraging the sense order in the distribution, i.e., assigning to each word sense a number of examples proportional to its position in the estimated word sense distribution.

4.3. Automatic domain adaptation

So far, we have introduced Train-O-Matic for the building of semantic annotated training data, and EnDi and DaD for the learning of word sense distribution given a corpus of sentences. We now explain how to couple these methods in order to automatically adapt the training data to the domain of application. Given an input text to disambiguate, we first run EnDi or DaD in order to build a distribution of senses of all the words we are interested in disambiguating, and then we follow the ordering given by their distributions and Zipf's law to select the number of training samples for each word sense s_i as follows:

[6] The list of 42 domains can be found at:http://babelnet.org/javadoc/it/uniroma1/lcl/babelnet/data/BabelDomain.html.

$$sample(s_i) = \frac{1}{i^z} * K \tag{14}$$

where K is the number of examples that are assigned to the first sense of a word (i.e., s_1), z is an exponential factor that we determine experimentally, as specified in Section 5 and i is the rank of the input sense s_i. In this way, if the input text is biased towards a certain domain (e.g., Sport, Medicine, etc.), we transfer this bias, thanks either to EnDi or to DaD, directly into the training set that is built following the same distribution as the target text. In this manner we are able to generate a training set based on information (i.e., sense distribution) gathered directly from the documents we want to disambiguate.

5. Experimental setup

Corpora for sense annotation. Train-O-Matic is independent of the corpora from which the sentences are drawn, thus we experimented on two different corpora: Wikipedia and the United Nations Parallel Corpus [27]. The former is the largest and most up-to-date encyclopedic resource, containing definitional information on general domains, the latter, instead, is a public collection of parliamentary documents of the United Nations, thus it is biased towards Politics domains. The application of Train-O-Matic to the two corpora produced two sense-annotated datasets, which we named T-O-M$_{Wiki}$ and T-O-M$_{UN}$, respectively.

Semantic Network. We created sense-annotated corpora with Train-O-Matic, both when using PPR vectors computed from vanilla WordNet, and when using WordNet$_{BN}$, our denser network obtained from the WordNet-induced subgraph of BabelNet (see Section 3).

Gold standard datasets. We performed our evaluations using the framework made available by Raganato et al. [28] on five different all-words datasets, namely: the Senseval-2 [29], Senseval-3 [30], SemEval-07 [31], SemEval-13 [32] and SemEval-15 [33] WSD datasets. In addition, we also report the results on ALL, i.e., the concatenation of all the above datasets. We note that the performance measured on ALL is not the arithmetic average of the performances on the various datasets (macro average), but rather the mean weighted by the number of tokens of each dataset (micro average). We focused on nouns only, given the fact that Wikipedia provides connections between nominal synsets only, and therefore contributes mainly to syntagmatic relations between nouns. Moreover, as SemEval-13 and SemEval-15 contain documents coming from different semantic domains (e.g., Finance, Sport, etc.), we manually classified each document with a BabelNet domain and used each document or set of documents belonging to the same domain as separate evaluation benchmarks.

Comparison sense-annotated corpora. To show the impact of our T-O-M corpora in WSD, we compared their performance on the above gold standard datasets, against training with:

- **SemCor**[7] [34], a corpus containing about 226,000 words annotated manually with WordNet senses.
- **One Million Sense-Tagged Instances**[8] [35, OMSTI], a sense-annotated dataset obtained via a semi-automatic approach based on the disambiguation of a parallel corpus, i.e., the United Nations Parallel Corpus, performed by exploiting manually translated word senses. Because OMSTI integrates SemCor to increase coverage, to keep a level playing field we excluded the latter from the corpus.
- **SenseDefs**[9] [36], a multilingual corpus built by jointly disambiguating the glosses of the same BabelNet synset in multiple languages. We used the refined version which contains almost 250 million sense annotations in 256 different languages.
- **EuroSense**[10] [37], similarly to SenseDefs, it exploits the parallel data of EuroParl [38] in order to augment the context and increase disambiguation precision. We used the high-precision version which contains roughly 122 million sense annotations in 21 languages.

We note that T-O-M, similarly to SenseDefs, is fully automatic and does not require any WSD-specific human intervention, or any aligned corpus, as in the case for OMSTI, or multilingual corpus, as in the case for EuroSense.

Training set size & shape. We varied the number of examples for each synset according to different types of distribution, as follows:

1. **Uniform:** assigns to every word sense the same amount of training samples. If the system is not able to find enough examples for that word sense we assign it all the samples that Train-O-Matic is able to find.

[7] http://lcl.uniroma1.it/wsdeval/.
[8] https://www.comp.nus.edu.sg/~nlp/sw/.
[9] http://lcl.uniroma1.it/sensedefs/.
[10] http://lcl.uniroma1.it/eurosense/.

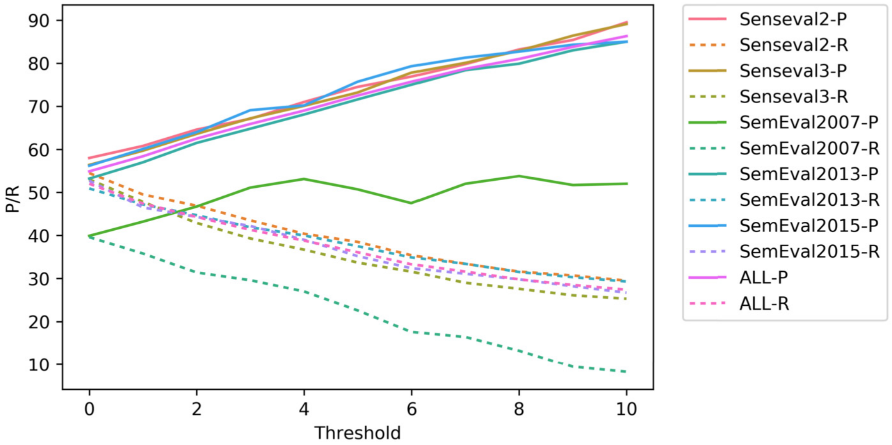

Fig. 2. Variation of Train-O-Matic step 2 performance on each test set when increasing the threshold on the disambiguation confidence.

2. **WordNet:** ranks the senses of each word according to WordNet ordering and then apply Zipf's law (see Formula (14)) to compute the number of sampled sentences for each synset according to its ranking, where i is the rank of the sense s in WordNet ordering and K is the number of sentences the first sense will receive.
3. **Automatic:** ranks the word senses of each word according to DaD or EnDi (see Section 4) and then apply Formula (14) as described above.
4. **Random:** assigns a random number of examples between 1 and K to each synset of a word where K is a parameter of the experiment. When the random number is higher than the available examples, it assigns the maximum number of examples available.
5. **Oracle:** ranks the word senses according to the sense frequency in the test set. Later, it assigns a number of examples to each sense following a Zipfian distribution as in Formula (14).
6. **Oracle 1-2:** acts as Oracle but swaps the first and the second sense in the synset ranking.
7. **Oracle 2-3:** acts in the same way as Oracle 1-2, but exchanges the second and the third sense in the synset ranking.
8. **Oracle**$_{prop}$**:** ranks the word senses according to their frequency in the test set. We then assign a number of examples to each meaning that is proportional to its frequency. Differently from Oracle, we use exactly the sense distribution that we find in the test data instead of approximating it by means of the in vitro Zipfian distribution (Formula (14)).

The Oracle variants are designed to prove the importance of identifying the predominant sense of a word, and, especially, to demonstrate how much mistaking the first sense for the second sense would affect the performance. For all the experiments, unless differently stated, we set the parameters K and z of Formula (14) by tuning the system (IMS) on a separate small in-house development dataset[11] as follows: $K = 500$ and $z = 2$.

Baseline. As a traditional baseline in WSD, we used the Most Frequent Sense (MFS) baseline given by the first sense in WordNet. The MFS is a very competitive baseline, due to the sense skewness phenomenon in language [1].

Reference system. In all our experiments, we used It Makes Sense [9, IMS], a WSD system based on linear Support Vector Machines, as our reference system for comparing its performance when trained on T-O-M to its performance when trained on other sense-annotated corpora (i.e., SemCor, OMSTI, etc.). Following the WSD literature, unless stated otherwise, we report performance in terms of F1, i.e., the harmonic mean of precision and recall.

Word Vectors. To test the semi-supervised variant of Train-O-Matic, we used Google word embedding vectors [39] extracted from Google News corpus.[12]

6. Results

6.1. Train-O-Matic disambiguation

As a first result, we report the performance of the shallow disambiguation that we carry out in the Sentence Scoring step of Train-O-Matic (Section 2.2.1). We note that Train-O-Matic does not rely on the second step alone to provide high-quality disambiguated sentences: in fact, the sentence ranking and selection step plays a fundamental role in ranking higher

[11] 50 word-sense pairs annotated manually.

[12] Available at https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit.

Table 3
F1 of IMS trained on T-O-M when PPR is obtained from the WordNet graph (WN) and from the WordNet-induced subgraph of BabelNet (BN).

System	Senseval-2	Senseval-3	SemEval-07	SemEval-13	SemEval-15	ALL
T-O-M$_{Wiki}$ WN	70.0	63.1	57.9	63.7	69.5	65.7
T-O-M$_{Wiki}$ BN	**70.5**	**67.4**	**59.8**	**65.5**	**68.6**	**67.3**

Table 4
F1 of IMS trained on Train-O-Matic, OMSTI and SemCor, and MFS for the Senseval-2, Senseval-3, SemEval-07, SemEval-13 and SemEval-15 WSD datasets.

System	Senseval-2	Senseval-3	SemEval-07	SemEval-13	SemEval-15	ALL
T-O-M$_{Wiki}$	70.5	67.4	59.8	**65.5**	**68.6**	67.3
T-O-M$_{UN}$	69.0	68.3	57.9	62.5	63.5	65.3
OMSTI	74.1	67.2	62.3	62.8	63.1	66.4
SemCor	**76.8**	**73.8**	**67.3**	**65.5**	66.1	**70.4**
MFS	72.1	72.0	65.4	63.0	66.3	67.6

those sentences that are most likely to be correctly annotated. Hence, to prove that the two steps are essential and complementary, and therefore that they enable Train-O-Matic to produce high-quality sense-annotated data, we directly applied Train-O-Matic on each Senseval and SemEval dataset and evaluated its disambiguation when increasing the threshold on the answers' scores. As can be seen from Fig. 2, when the threshold is 0 – meaning that all the disambiguated words are taken into account – the performance is modest and would not be good enough to provide sense-annotated data. However, when the threshold is increased up to 10, we can see that the precision increases remarkably on each dataset and reaches 86.3 points on the ALL dataset and a recall of 27.4. This confirms our hunch that a filtering step is necessary for retrieving high-quality data. On the other hand, the low recall that Train-O-Matic attains is not an issue as long as we have a large corpus of raw sentences at our disposal.

6.2. Impact of syntagmatic relations

With this experiment we aim at verifying our hunch, discussed in Section 3, that the edges coming from our WordNet-induced subgraph of BabelNet (WordNet$_{BN}$) add useful semantic relations that are exploited effectively by Train-O-Matic when generating the training set. Indeed, as can be seen from Table 3, when IMS is trained on Train-O-Matic using WordNet$_{BN}$, it performs from 0.5 to roughly 4 F1 points better than when using vanilla WordNet. This is then reflected in an overall increase in performance of 1.6 F1 points. These results confirm our intuition that excluding syntagmatic relations (such as those contained in Wikipedia and thus in BabelNet) limits the performance of Train-O-Matic. In Table 4 we show the results of IMS trained on Train-O-Matic when using Wikipedia as underlying corpus instead of the United Nations corpus. In this case, T-O-M$_{UN}$ performs consistently worse than T-O-M$_{Wiki}$ (except for Senseval-3) with an overall loss of 2 F1 points. Interestingly enough, OMSTI manages to beat Train-O-Matic when using the United Nations parallel corpus as sentence source, i.e., the same source as the one used by OMSTI itself. This result is also in part due to OMSTI benefiting from its semi-automatic approach that relies on human-tagged words in Chinese.

However, as one can see from the Table, Train-O-Matic, when left free to exploit a larger and more general-domain corpus such as Wikipedia, led IMS to generalise and perform better on most of the benchmarks than both T-O-M$_{UN}$ and OMSTI. Indeed, it is not coincidental that Train-O-Matic$_{UN}$ performed better than Train-O-Matic$_{Wiki}$ only on Senseval-3. This latter, in fact, contains sentences drawn from the Penn TreeBank corpus [40] which is based on The Wall Street Journal articles, and, thus, like the United Nations corpus, it is biased towards Politics and Finance. In Table 5 we show, indeed, that the first sense of *plane*, i.e., the airplane meaning, always appears with contexts related to war when the UN corpus is used. In contrast, when Wikipedia is used, Train-O-Matic is able to retrieve different contexts from different domains (e.g., video games, war, civil transportation, etc.) providing different scenarios in which the sense can be used. Due to the difference in the domain bias of the two corpora, comparing Train-O-Matic$_{Wiki}$ and OMSTI might seem unfairly in favour of Train-O-Matic, however, this is not the case. In fact, being able to choose the underlying corpus is one of the distinctive features that makes Train-O-Matic easily adaptable to different scenarios. OMSTI, on the other hand, was designed to be applied on the United Nations corpus, with some additional manual annotation in order to disambiguate the Chinese words. Therefore, our comparison places both Train-O-Matic$_{Wiki}$ and OMSTI in the same ballpark and helps investigate different features of both approaches.

6.3. Comparison against sense-annotated corpora

We now move to comparing the performance of Train-O-Matic, which is fully automatic, against corpora which are annotated manually (SemCor) and semi-automatically (OMSTI). In Table 4 we show the F1-score of IMS on each gold standard

Table 5
Excerpt of training data for the *airplane* sense of plane when the UN and Wikipedia corpora are used.

UN	[...] including transport and attack helicopters, cargo **planes** used as bombers [...]
	[...] using helicopter gun ships, Antonov **planes** and MIG fighters.
	[...] backed by attack helicopters, fighter **planes** and unmanned surveillance drones[...]
	[...] responded by destroying the **planes** used by FANCI during the air raids [...]
	[...] providing air support to [...], operating more than 100 **planes** and helicopters .
WIKI	[...] the main runway can be used by **planes** with up to around 180 passengers [...]
	cargo transportation services, it is suitable for **planes** like Boeing up to 757 [...]
	[...] a successful attack of air-to-air rockets on another **plane** took place on August 20 [...]
	Or even Rocket Boosts, which increases the **planes** speed dramatically [...]
	Sometimes a player [...] could fire a homing missile at a random opponent in front of the **plane**.

Table 6
Comparison between vanilla Train-O-Matic and its neural variants with word embeddings (WE).

Dataset	T-O-M$_{Vanilla}$	T-O-M$_{WE}$	OMSTI	SemCor	MFS
Senseval-2	70.5	**72.2**	74.1	76.8	72.1
Senseval-3	67.4	**72.2**	67.2	73.8	72.0
SemEval-07	59.8	**64.8**	62.3	67.3	65.4
SemEval-13	**65.5**	63.9	62.8	65.5	63.0
SemEval-15	**68.6**	65.5	63.1	66.1	66.3
ALL	67.3	67.9	66.4	70.4	67.6

Table 7
Number of unique and total tokens comprised in each dataset for which at least one training example is provided by OMSTI, EuroSense, SenseDefs and Train-O-Matic for each English dataset.

Dataset	OMSTI		EuroSense		SenseDefs		Train-O-Matic		ALL	
	Unique	Tot	Unique	Tot	Unique	Tot	Unique	Tot	Unique	Tot
Senseval-2	197	467	388	961	399	1024	400	1005	436	1066
Senseval-3	197	500	420	827	422	845	435	860	469	900
SemEval-07	68	89	126	156	123	153	127	159	127	159
SemEval-13	249	757	628	1503	625	1489	629	1429	751	1644
SemEval-15	102	246	218	493	224	493	226	495	253	531
ALL	456	2059	1311	3940	1334	4004	1350	3948	1557	4300

dataset in the evaluation framework and on all datasets merged together (last column), when it is trained with the various corpora described above.

As can be seen, Train-O-Matic$_{Wiki}$ obtains higher performance than OMSTI (up to 5.5 points above) on 3 out of 5 datasets, scoring 1 point above OMSTI overall. The MFS is in the same ballpark as T-O-M$_{Wiki}$, performing better on some datasets and worse on others. We note that IMS trained on T-O-M$_{Wiki}$ succeeds in surpassing or obtaining the same results as IMS trained on SemCor on SemEval-15 and SemEval-13. We view this as a significant achievement given that no manual annotations were involved in the creation of our corpus.

Because overall T-O-M$_{Wiki}$ outperforms T-O-M$_{UN}$, in what follows we report all the results with T-O-M$_{Wiki}$, except for the domain-oriented evaluation (see Section 6.7).

6.4. Comparison against neural methods

We now compare the vanilla Train-O-Matic against its neural variant in which pretrained embeddings (WE) are used instead of PPR to perform the sentence scoring step (see Section 2.2.2).

As shown in Table 6 T-O-M$_{WE}$ outperforms T-O-M$_{Vanilla}$ on 3 out of 5 datasets. This result was expected as, in fact, T-O-M$_{WE}$ is semi-supervised and exploits sense annotations in order to build a vector for each synset. Thus, it is interesting to see that the vanilla version is still able to outperform its alternative on SemEval-13 and SemEval-15 by 1.6 and 3 points, while scoring only 0.6 points lower on the concatenation of all datasets (ALL). We also note that T-O-M$_{WE}$ beats OMSTI – which is also semi-supervised – on all the datasets but Senseval-2, and gets very close to SemCor, especially on Senseval-3 and SemEval-07 where the vanilla version of T-O-M, instead, performs worse.

6.5. Performance without backoff strategy

IMS uses the MFS as a backoff strategy when no sense can be output for a target word in context [9]. Consequently, the performance of the MFS is mixed with that of the SVM classifier. As shown in Table 7, OMSTI is able to provide annotated

Table 8
Precision, Recall and F1 of IMS trained on OMSTI, Train-O-Matic, EuroSense, and SenseDefs corpora without MFS backoff strategy for Senseval-2, Senseval-3, SemEval-07, SemEval-13 and SemEval-15.

Dataset	OMSTI			Train-O-Matic			EuroSense			SenseDefs		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Senseval-2	64.8	28.5	39.6	69.5	65.5	**67.4**	63.1	55.2	58.9	67.4	67.4	**67.4**
Senseval-3	55.7	31.0	39.8	66.1	63.1	**64.6**	49.4	45.1	47.2	61.1	56.3	58.6
SemEval-07	64.1	35.9	46.0	59.8	59.8	**59.8**	40.4	39.6	40.0	51.7	48.3	49.9
SemEval-13	50.7	23.4	32.0	61.3	53.3	**57.0**	56.7	48.2	52.1	41.1	39.0	40.0
SemEval-15	57.0	26.7	36.4	67.0	62.3	**64.6**	56.7	51.2	53.8	50.4	42.2	46.0
ALL	56.5	27.0	36.5	65.1	59.7	**62.3**	56.0	49.3	52.5	55.7	49.5	52.4

Table 9
Evaluation on the ALL dataset of the same WSD system trained on training sets balanced in different ways.

Measure	WN	EnDi	DaD	Uniform	Random	Oracle	Oracle 1-2	Oracle 2-3	Oracle$_{prop}$
Precision	65.1	48.0	60.3	47.1	45.3	81.3	47.6	79.8	88.9
Recall	59.7	43.5	54.3	42.7	41.0	73.5	43.1	72.2	83.9
F1	62.3	45.6	57.3	44.6	43.0	77.2	45.3	75.8	86.3

sentences for roughly half of the tokens in the datasets. This means that in around 50% of cases OMSTI gives an answer based on the IMS backoff strategy. Train-O-Matic together with EuroSense and SenseDefs, instead, is able to cover almost all words in each dataset with at least one training sentence.

To determine the real impact of the different training data, we therefore decided to perform an additional analysis of the IMS performance when the MFS backoff strategy is disabled. Since we suspected the system would not always return a sense for each target word, in this experiment we measured precision, recall and their harmonic mean, i.e., F1. The results in Table 8 confirm our hunch, showing that OMSTI's recall drops heavily, thereby affecting F1 considerably. T-O-M performances, instead, remain high in terms of precision, recall and F1. This confirms that OMSTI relies heavily on the predominant sense information obtained from the MFS and from SemCor. EuroSense and SenseDefs, instead, managed to lead IMS to better results than OMSTI, but fail to beat Train-O-Matic on all datasets. This is because neither of the two approaches take advantage of any mechanism of sentence ranking, thus their datasets contain more noise than the one built with T-O-M.

6.6. Comparison of different methods for training balancing

Now that we have shown the ability of Train-O-Matic to generate training data, we investigate in more detail how changing the strategy for assigning the number of examples to a synset affects the performance of the trained model. In Table 9 we show the performance of IMS when trained on data balanced with the different methods described in Section 5. Note that the results are computed without the backoff strategy, thus there are actually words for which the classifier does not output an answer, and this is why precision and recall are different. The *Random* method, which assigns a random number of examples to each synset, is the one that performed worst, with 43 points in F1. Interestingly, we note that the *Uniform* strategy (which assigns the same number of examples to each sense) leads IMS to achieve only 1.6 points more than the *Random* approach. This means that the classifier needs to learn the bias given by the natural distribution of senses in order to achieve acceptable results. Another interesting result is the one achieved with the Oracle strategies. We can see from Table 9 that the Oracle and Oracle$_{prop}$ strategies lead IMS to 77.2 and 86.3 F1 points, respectively, on the ALL dataset. These results are from 10 to almost 20 points higher than the best performing system reported by [41] on the same dataset. As expected, Oracle$_{prop}$ is the best system across the board, as it is the one that most exploits the information available in the test set. In fact, it leverages the exact number of occurrences of the meanings – and not only their rankings – to determine the number of sentences to assign to each sense.

What is more interesting, on the other hand, is how poorly Oracle 1-2 performs. With 45.3 F1 points, it is able to beat only the Uniform strategy by 1 point. In contrast, Oracle 2-3 performs just 2 points lower than Oracle. This shows how sensitive the model is to the sense distribution in the training data and, more interestingly, how identifying the most common sense of a word is actually more important than correctly estimating the rest of the distribution. In fact, swapping the senses in the distribution from the second position and on, does not degrade the performance as happens when the first sense is swapped with the second one. Together with all these baselines, we also report the performance achieved by IMS when we use the WordNet sense distribution and the two predicted by EnDi and DaD.

As expected the WordNet ordering outperforms the two automatic baselines as it takes benefits from the manual annotations of SemCor. While we note that EnDi performs poorly, only slightly better than Oracle 1-2, we stress the fact that DaD, despite being fully unsupervised, enables IMS to achieve 57.3 F1 points, 5 less than when WordNet is used. These outcomes show the importance of finding better and more clever methods to induce the most suitable word sense distribution. Finally, as the word sense distribution has proven to play an important role for the classifier performance, we investigated whether,

Table 10
Evaluation of IMS with Oracle distribution when random senses are assigned to each word occurrences.

Method	Precision	Recall	F1
noisy-T-O-M 100	51.7	48.8	**50.2**
noisy-T-O-M 200	51.1	48.4	49.7
noisy-T-O-M 500	49.0	46.4	47.6

Table 11
Performance comparison over SemEval-13 domain-specific datasets.

Domain	Backoff	T-O-M$_{Wiki}$			T-O-M$_{UN}$			OMSTI			SemCor	MFS	Size
		P	R	F1	P	R	F1	P	R	F1	F1	F1	
Biology	MFS	63.0	63.0	63.0	65.9	65.9	**65.9**	65.9	65.9	**65.9**	66.3	64.4	135
	-	59.0	53.3	56.0	62.3	56.3	**59.2**	48.1	18.5	26.7			
Climate	MFS	68.1	68.1	**68.1**	63.4	63.4	63.4	68.0	68.0	68.0	70.1	67.5	194
	-	63.4	50.0	**55.9**	57.5	45.4	50.7	58.0	24.2	34.2			
Finance	MFS	68.0	68.0	**68.0**	56.6	56.6	56.6	64.4	64.4	64.4	63.7	56.2	219
	-	62.1	51.6	**56.4**	48.4	40.2	43.9	57.4	28.3	37.9			
Health Care	MFS	65.2	65.2	**65.2**	60.1	60.1	60.1	52.9	52.9	52.9	62.7	56.5	138
	-	61.3	55.1	**58.0**	55.6	50.0	52.6	34.6	18.4	24.0			
Politics	MFS	65.2	65.2	65.2	66.3	66.3	**66.3**	63.4	63.4	63.4	69.5	67.7	279
	-	62.5	54.8	58.4	63.9	55.9	**59.6**	54.1	21.5	30.8			
Social Issue	MFS	68.5	68.5	**68.5**	63.6	63.6	63.6	65.6	65.6	65.6	66.8	67.6	349
	-	63.1	53.0	**57.6**	57.2	47.9	52.1	54.7	25.2	34.5			
Sport	MFS	60.3	60.3	60.3	60.9	60.9	**60.9**	58.8	58.8	58.8	60.4	57.6	330
	-	58.3	54.6	**56.4**	58.1	53.3	55.5	45.0	23.0	30.4			

Table 12
Performance comparison over the Biomedicine and Maths & Computer domains in SemEval-15.

Domain	Backoff	T-O-M$_{Wiki}$			T-O-M$_{UN}$			OMSTI			SemCor	MFS	Size
		P	R	F1	P	R	F1	P	R	F1	F1	F1	
Biomedicine	MFS	76.3	76.3	**76.3**	66.0	66.0	66.0	64.9	64.9	64.9	70.3	71.1	100
	-	76.1	72.2	**74.1**	64.4	59.8	62.0	60.5	26.8	37.2			
Maths & Computer	MFS	50.0	50.0	**50.0**	48.0	48.0	48.0	36.0	36.0	36.0	40.6	40.9	97
	-	50.0	47.0	**48.5**	47.8	44.0	45.8	21.2	11.0	14.5			

solely by knowing the sense distribution, it is possible for a classifier to achieve good performance. We therefore studied the performance of IMS when trained on noisy data (noisy-T-O-M), i.e., where random senses are assigned to occurrences of the target word, while following the Oracle distribution of senses (see Section 5). Table 10 shows that contexts do, indeed, matter and that merely mimicking the test set distribution of senses is not enough to build a well-performing classifier. This is further confirmed by the fact that, increasing the number of examples each synset can receive (from 100 to 500), leads to a performance drop. Intuitively this happens because more noise is added to the dataset and thus the classifier gets increasingly confused.

6.7. Domain-oriented WSD

To further inspect the ability of Train-O-Matic to enable disambiguation in different domains, we evaluated it on specific documents from the various gold standard datasets that could be clearly associated with a domain label. Specifically, we tested on 13 documents belonging to different domains[13] from SemEval-13 and two documents belonging to two different domains[14] from SemEval-15 and carried out two separate tests and evaluations of Train-O-Matic on each domain: one using the MFS backoff strategy from WordNet, and one not using it.

In Tables 11 and 12 we report the results separately for each domain of IMS trained on both T-O-M$_{Wiki}$ and T-O-M$_{UN}$ so to measure the impact of each corpus, i.e., Wikipedia and the United Nations corpus, on each specific domain. As can

[13] Namely, Biology, Climate, Finance, Health Care, Politics, Social Issue and Sport.

[14] Maths & Computer and Biomedicine.

Table 13
Performance comparison over SemEval-13 domain-specific datasets when EnDi and Dad are used to compute MFS (+MFS) and sense distribution (+SD).

Domain	Backoff	T-O-M$_{DaD}$			T-O-M$_{EnDi}$			SemCor	WN MFS	Size
		P	R	F1	P	R	F1	F1	F1	
Biology	+MFS	63.7	63.7	63.7	63.7	63.7	63.7			
	+SD	69.8	65.2	67.4	54.8	51.1	52.9	66.3	64.4	135
	+MFS+SD	71.1	71.1	**71.1**	57.0	57.0	57.0			
Climate	+MFS	66.0	66.0	66.0	65.5	65.5	65.5			
	+SD	64.5	56.2	60.1	52.7	45.9	49.0	**70.1**	67.5	194
	+MFS+SD	66.5	66.5	66.5	55.7	55.7	55.7			
Finance	+MFS	58.9	58.9	58.9	58.9	58.9	58.9			
	+SD	62.0	58.0	59.9	52.2	48.9	50.5	63.7	56.2	219
	+MFS+SD	63.9	63.9	**63.9**	54.8	54.8	54.8			
Health Care	+MFS	58.0	58.0	58.0	58.7	58.7	58.7			
	+SD	66.4	60.1	63.1	50.4	45.7	47.9	62.7	56.5	138
	+MFS+SD	68.1	68.1	**68.1**	54.3	54.3	54.3			
Politics	+MFS	72.4	72.4	72.4	71.3	71.3	71.3			
	+SD	70.2	65.9	68.0	59.9	55.9	57.7	69.5	67.7	279
	+MFS+SD	72.0	72.0	**72.0**	60.9	60.9	60.9			
Social Issue	+MFS	67.6	67.6	67.6	67.9	67.9	67.9			
	+SD	72.2	63.9	67.8	59.5	52.7	55.9	66.8	67.6	349
	+MFS+SD	73.9	73.9	**73.9**	63.0	63.0	63.0			
Sport	+MFS	55.5	55.5	55.5	55.5	55.5	55.5			
	+SD	53.4	50.6	51.9	46.3	43.9	45.1	**60.4**	57.6	330
	+MFS+SD	53.6	53.6	53.6	47.0	47.0	47.0			

Table 14
Performance comparison over the Biomedicine and Maths & Computer domains in SemEval-15 when DaD and EnDi are used to predict the sense distribution of the test set.

Domain	Mode	T-O-M$_{DaD}$			T-O-M$_{EnDi}$			SemCor	MFS	Size
		P	R	F1	P	R	F1	F1	F1	
Biomedicine	+MFS	68.0	68.0	68.0	68.0	68.0	68.0			
	+SD	75.0	71.1	**73.0**	48.9	45.4	47.1	70.3	**71.1**	100
	+MFS+SD	75.3	75.3	**75.3**	51.5	51.5	51.5			
Maths & Computer	+MFS	51.0	51.0	**51.0**	50.3	50.3	50.3			
	+SD	62.1	59.0	**60.5**	40.9	38.0	39.4	40.6	40.9	97
	+MFS+SD	62.0	62.0	**62.0**	41.4	41.4	41.4			

be seen from Table 11, T-O-M$_{Wiki}$ systematically attains higher scores than OMSTI (except for the biology domain), and, in most cases, attains higher scores than MFS when the backoff is used, with a systematic increase over OMSTI on both recall and F1 when the backoff strategy is disabled. This demonstrates the usefulness of the corpora annotated by Train-O-Matic not only on open text, but also on specific domains. We note that T-O-M$_{UN}$ obtains the best results in the Politics domain, which is the closest domain to the UN corpus from which its training sentences are obtained. In Table 12 we can observe a similar behaviour with Train-O-Matic attaining the highest performance across the board.

We now compare the results attained by IMS trained on Train-O-Matic$_{Wiki}$ when standard WordNet MFS and sense distribution were used (see Tables 11 and 12), against those in Tables 13 and 14 where IMS is trained on the following three different datasets:

1. **+MFS:** We used Vanilla T-O-M to build the training corpus and set the MFS learnt by DaD or EnDi as backoff strategy.
2. **+SD:** We used the learnt sense distribution to order the word senses and decide the amount of training examples to assign to each one (see Section 4.3).
3. **+MFS+SD:** We used both: the learnt MFS as backoff strategy and the learnt sense distribution to shape the training set.

Results in Tables 13 and 14, not only demonstrate that just recomputing the MFS according to the text we want to disambiguate is an effective strategy, but also that it is possible to automatically adapt the training data to the target document we are about to disambiguate. In fact, just by shaping the training data with the learned distribution (+SD) of DaD, IMS gains from 3.5 to 10.2 points in F1 (Table 13) when compared to IMS trained on T-O-M, without MFS (see Table 11, second row of each domain). This is the case for all the domains except Sport, which is the only domain where Train-O-Matic fails to improve the IMS performance, registering a loss of almost 5 points. Similarly, on the SemEval-15

Table 15
Kullback-Leibler divergence between the distribution induced by EnDi and DaD and the Oracle distribution computed on each domain.

Dataset	Domain	EnDi	DaD
SemEval-13	Biology	0.173	**0.136**
	Climate	0.153	**0.128**
	Finance	0.166	**0.157**
	Health Care	**0.116**	0.141
	Politics	0.155	**0.114**
	Social Issue	0.170	**0.124**
	Sport	0.213	**0.177**
SemEval-15	Biomedicine	0.155	**0.119**
	Maths & Computer	0.169	**0.142**

(Table 14), IMS trained on Train-O-Matic +SD attains 10.5 higher results on the Maths & Computer domain at the small cost of 1.0 F1 points less on the Biomedicine domain (Table 12, first row). Even better results are achieved when we couple the learned sense distribution and the learned most frequent sense (+MFS+SD). In this configuration IMS trained on Train-O-Matic +MFS+SD achieves better scores (2.9-8.1 points more in Table 13) on Biology, Health Care, Politics, Social Issue and Biomedicine domains, than when WordNet MFS and sense distribution are used (see Tables 11 and 12 first row of each domain), while performing between 1 and 6.7 F1 points worse on the 3 remaining domains. We also note that Sport is the only domain where following the sense distribution learned by DaD leads to a significant decrease in performance (6.7 points). We further investigated this case and we noticed that the most misclassified word is *game*. This happened for two reasons: 1) The domain distribution built by DaD in its first phase is skewed towards Sport, which received the highest probability (0.19), and Philosophy and Psychology with a probability of 0.18. Thus there is no clear domain for the document and as a consequence synsets close to both domains will receive similar probabilities; 2) the sport sense of game in BabelNet is not associated with the Sport domain, and therefore it did not benefit from the direct connection with the domain during the domain propagation step of DaD (see Section 4.2). As a consequence, it does not result as one of the most frequent senses of game in the document, even if it is! We also note that, when compared against IMS trained on SemCor, Train-O-Matic coupled with both the learnt MFS and distribution is able to beat it by 0.2 to 7.1 points on all domains except two (Climate and Sport).

A similar behaviour is observed in Table 14 where Train-O-Matic +MFS +SD always beats IMS trained on SemCor and outperforms IMS when WordNet MFS is used on both domains.

EnDi and DaD Intrinsic Evaluation. To further investigate the impact of EnDi and DaD on the domain-specific performance of IMS, we now measure the Kullback-Leibler divergence (KLD) between their induced distributions and the Oracle distribution which we extract from each test set separately, as explained in Section 5. KLD computes the logarithmic difference between two probability distributions P and Q as follows:

$$D_{KL} = -\sum_{x \in \mathcal{X}} P(x) log\left(\frac{Q(x)}{P(x)}\right) \tag{15}$$

$D_{KL} = 0$ indicates perfect match between the two distributions. Since the KL divergence considers the probabilities of each distribution component, we have to make sure that the Oracle distribution takes into account all the possible meanings of a word and not only those appearing in the test set. Hence, to this end, whenever it happens that some meanings are missing from the Oracle distribution, we add their components with an associated probability of 0. We then compute the final Oracle distribution for a given word by adding a small smoothing factor to each score and normalising the values by their sum. For example, if the word bank$_n$ only appears in its financial institution meaning (i.e., bank$_n^2$) within the test set under analysis, we assign it a probability of 1 while assigning 0 to all the other meanings. Then, by applying the smoothing with a small value $\lambda = 10^{-3}$ we end up with a distribution where most of the mass is concentrated on bank$_n^2$ and the rest is spread across the other components. EnDi and DaD, instead, are designed to take into account all the possible word synsets contained by the sense inventory.

In Table 15 we report the divergence values of DaD and EnDi in comparison to the Oracle distribution. DaD always attains a lower KLD than EnDi compared to the Oracle distribution across all the domains except Health and Medicine, hence confirming the empirical results reported in the previous sections. In Fig. 3 (top) we show the distributions computed by Oracle, DaD and EnDi on the noun world$_n$ in the Climate domain. As one can see, DaD produces a distribution that is more similar to the gold one than the distribution induced by EnDi, and correctly identifies the planet meaning of *world* as the most likely one. In fact, DaD benefits from the association between world$_n^4$ and the domain Physics and Astronomy in BabelNet, which is highly correlated with the domain Climate. EnDi, instead, fails to predict the two most frequent senses for world$_n$, attributing highest scores to world$_n^2$ (People in general) and world$_n^5$ (People in general considered as a whole). A similar case is shown in Fig. 3 (bottom), where the distribution of treasury$_n$ senses is correctly induced by DaD

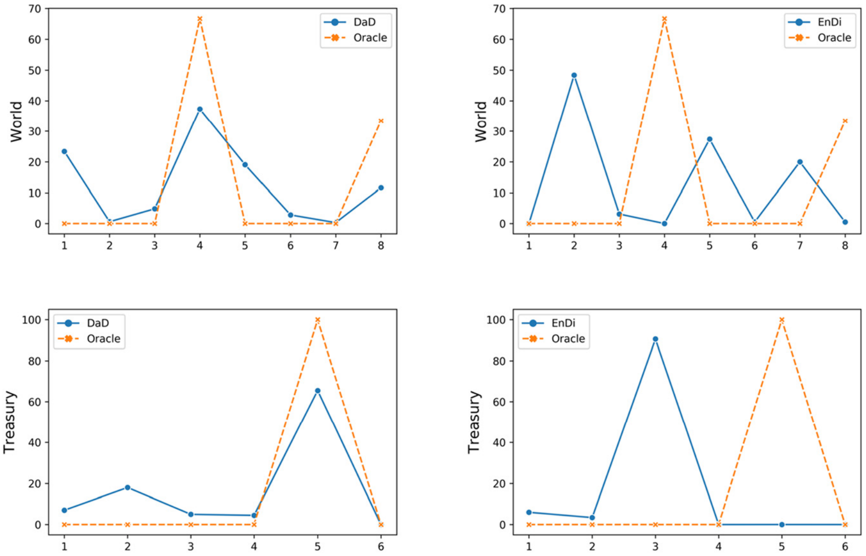

Fig. 3. Distribution induced by DaD (left) and EnDi (right) compared with the one extracted from the test set for the word *world*$_n$ (top), and *treasury*$_n$ (bottom). Each item on the X axis is the i-th meaning of the word under analysis.

Table 16
Performance comparison between T-O-M and SemEval-15's best run of SUDOKU on general domain (ALL) and domain specific (Maths & Computer and Biomedicine), when no word sense distribution learning took place and when EnDi and DaD were used.

Language	Dataset	Best System	T-O-M$_{Vanilla}$		
		F1	P	R	F1
Italian	ALL	56.6	65.1	55.6	**59.9**
	Maths & Computer	46.6	52.7	43.3	**47.6**
	Biomedicine	65.9	76.6	67.6	**71.8**
Spanish	ALL	56.3	61.3	54.8	**57.9**
	Maths & Computer	42.4	53.3	44.4	**48.5**
	Biomedicine	65.5	71.8	65.5	**68.5**

thanks to the direct link between the domain of Finance and treasury^5_n (the federal department that collects revenue and administers federal finances) and the higher number of connections between treasury^5_n and the other nodes of the graph. EnDi, in contrast, skews the distribution towards the wrong meaning, i.e., treasury^3_n (Negotiable debt obligations [...]) which, while being related to the Finance domain, is not the correct one.

In summary, the extrinsic domain-specific evaluation confirmed the intuition that WSD is more effective when domain information is provided. Specifically, DaD proved to be particularly effective, leading IMS to results that are better than those attained when manually-annotated data are used to induce the distributions. Moreover, it showed itself to be capable of inducing sense probabilities that are similar, to some extent, to those extracted by Oracle. While we note that there might be many different ways to provide distributional and domain information, we prove that, balancing the training set according to the learned distribution of word senses, which in DaD depends directly on the domain distribution of the document, is an effective way of providing domain information to the classifier and increasing performance by several F1 points without any human intervention.

7. Scaling up to multiple languages

Experimental Setup. In this Section we investigate the ability of Train-O-Matic to scale to other languages supported by BabelNet, such as Italian, Spanish, German and French, for which training data for WSD is not available.

Thanks to BabelNet, in fact, Train-O-Matic, as well as DaD and EnDi, can be used to generate sense-annotated data and sense distributions for any language supported by the knowledge base. So, in order to build new training datasets for the 4 languages, we ran Train-O-Matic on their corresponding versions of Wikipedia, then we tuned the two parameters K and z

Table 17
Performance comparison between T-O-M and SemEval-13's best SUDOKU Run.

Dataset	Language	Method	Best System	T-O-M		
			F1	P	R	F1
SemEval-13	Italian	Vanilla	65.8	69.6	65.7	**67.6**
		+MFS$_{DaD}$	65.8	71.0	71.0	**71.0**
		+MFS$_{EnDi}$	65.8	70.9	70.9	**70.9**
	Spanish	Vanilla	**71.0**	68.0	65.6	66.8
		+MFS$_{DaD}$	**71.0**	69.0	69.0	69.0
		+MFS$_{EnDi}$	**71.0**	68.6	68.6	68.6
	French	Vanilla	**60.5**	61.1	59.9	60.5
		+MFS$_{DaD}$	**60.5**	61.4	61.4	**61.4**
		+MFS$_{EnDi}$	**60.5**	61.1	61.1	**61.1**
	German	Vanilla	62.1	65.9	60.8	**63.2**
		+MFS$_{DaD}$	62.1	67.5	67.5	**67.5**
		+MFS$_{EnDi}$	62.1	67.2	67.2	**67.2**
SemEval-15	Italian	Vanilla	56.6	65.1	55.6	**59.9**
		+MFS$_{DaD}$	56.6	65.9	65.9	**65.9**
		+MFS$_{EnDi}$	56.6	64.7	64.7	**64.7**
	Spanish	Vanilla	56.6	61.3	54.8	**57.9**
		+MFS$_{DaD}$	56.3	62.6	62.6	**62.6**
		+MFS$_{EnDi}$	56.3	61.0	61.0	**61.0**

on an in-house development dataset.[15] In contrast to the English setting, in order to calculate Formula (14) we sorted the senses of each word according to BabelNet ordering, or by using one of the distributions learnt by EnDi or DaD. Finally, we used the output data to train IMS.

Results. To perform our evaluation we chose both the multilingual SemEval tasks from past years (SemEval-2013 task 12 and SemEval-2015 task 13), which include gold data for Italian, Spanish, French and German. As can be seen from Table 16, Train-O-Matic enabled IMS to perform better than the best participating system [42, SUDOKU] in all three settings (All domains, Maths & Computer and Biomedicine). Its performance was, in fact, 1 to 3 points higher, with a 6-point peak on Maths & Computer in Spanish and on Biomedicine in Italian. This demonstrates the ability of Train-O-Matic to enable supervised WSD systems to surpass state-of-the-art knowledge-based WSD approaches in different languages where manually-annotated data are not available and without relying on manually curated data for training. In Table 17 we show results in all the languages of SemEval-13 and SemEval-15 achieved by IMS trained on vanilla Train-O-Matic and when our MFS is used as backoff strategy. As can be seen our MFS is also beneficial in multilingual settings where no manual MFS is available. While it always improves results over the vanilla Train-O-Matic, we note that the learnt MFS also enabled IMS to surpass the state of the art on those datasets where vanilla T-O-M tended to perform worse than the comparing system; namely the Spanish part of SemEval-15 and the French part of SemEval-13.

8. Related work

8.1. Word Sense Disambiguation

There are two mainstream approaches to Word Sense Disambiguation: supervised and knowledge-based approaches. Both suffer in different ways from the so-called knowledge acquisition bottleneck, that is, the difficulty in obtaining an adequate amount of lexical-semantic data: for training in the case of supervised systems, and for enriching semantic networks in the case of knowledge-based ones [13,1].

State-of-the-art supervised systems include Support Vector Machines, such as IMS [9] and, more recently, LSTM neural networks with attention and multitask learning [12], as well as LSTMs paired with nearest neighbours classification [43,11]. Yuan et al. [11] also integrates a label propagation algorithm in order to enrich the sense annotated dataset. The main difference from our approach lies in its need for a manually annotated dataset to start the label propagation algorithm. Such datasets are usually unavailable for languages other than English, whereas Train-O-Matic is fully automatic. An evaluation against Yuan et al. [11] system would have been interesting, but neither the proprietary training data nor the code are available at the time of writing. Other interesting approaches which also aim at augmenting the information available in the network are the one introduced by Luo et al. [44] and Kumar et al. [45]. Differently from the work of Yuan et al. [11], they leverage the synsets' glosses contained in a knowledge base to provide additional information to the network for

[15] We set $K = 100$ and $z = 2.3$ for Spanish and $K = 100$ and $z = 2.5$ for Italian, $k = 100$ and $z = 2.0$ for German and $k = 100$ and $z = 2.3$ for French.

disambiguating the input sentence. In order to generalize effectively, these supervised systems require large numbers of training instances annotated with senses for each target word occurrence. Overall, this amounts to millions of training examples for each language of interest, a number that is not within reach for any language. In fact, no supervised system has been submitted in major multilingual WSD competitions for languages other than English [32,33]. To get rid of these limitations and to aid knowledge-based models to perform in the same ballpark of supervised approaches, Maru et al. [6] introduced SyntagNet,[16] a resource of semantic collocations which, when integrated in a knowledge base, proved to be effective in increasing the performance of a knowledge-based model such as UKB [4] by several points. Other approaches, instead, focused on automatically creating sense-tagged corpora, hence enabling supervised models to perform multilingual WSD. Raganato et al. [41] developed 7 heuristics to grow the number of hyperlinks in Wikipedia pages. Otegi et al. [46] applied a different disambiguation pipeline for each language to parallel text in Europarl [47] and QTLeap [48], in order to enrich them with semantic annotations. Taghipour and Ng [35], the work closest to ours, exploits the alignment from English to Chinese sentences of the United Nations Parallel Corpus [27], in order to reduce the ambiguity of English words. Another approach that is similar to ours is the one proposed by Scarlini et al. [49, OneSeC]. This latter exploits the Wikipedia Categories and the Wikipedia pages in order to generate annotated instances for the nominal tokens of a language lexicon in potentially all the languages supported by both BabelNet and Wikipedia.

Train-O-Matic stands out from the aforementioned approaches inasmuch, in contrast to OMSTI, it is fully automatic and can be applied to any kind of corpus (and language) depending on the specific need, whereas, in contrast to OneSeC, it is more flexible as it does not rely on any kind of structure of the input corpus.

Earlier attempts at the automatic extraction of training samples were made by Agirre and de Lacalle [50] and Fernández et al. [51]. Both exploited the monosemous relatives method [52] in order to retrieve sentences from the Web which contained a given monosemous noun or a relative monosemous word (e.g., a synonym, a hypernym, etc.). As can be seen in [51] this approach can lead to the retrieval of very accurate examples, but its main drawback lies in the number of senses covered. In fact, for all those synsets that do not have any monosemous relative, the system is unable to retrieve examples, thus heavily affecting the performance in terms of recall and F1.

Knowledge-based WSD, instead, bypasses the heavy requirement of sense-annotated corpora by applying algorithms that exploit a general-purpose semantic network, such as WordNet, which encodes the relational information that interconnects synsets via different kinds of relation. Approaches include variants of Personalised PageRank [4] and densest subgraph approximation algorithms [5] which, thanks to the availability of multilingual resources such as BabelNet, can easily be extended to perform WSD in arbitrary languages. Other approaches to knowledge-based WSD exploit the definitional knowledge contained in a dictionary. The Lesk algorithm [53] and its variants [54–56] aim at determining the correct sense of a word by comparing each word-sense definition with the context in which the target word appears. The limit of knowledge-based WSD, however, lies in the absence of mechanisms that can take into account the very local context of a target word occurrence, including non-content words such as prepositions and articles.

8.2. Word Sense Distribution Learning

While Word Sense Disambiguation has been a very active field in recent years, Word Sense Distribution Learning has not received the same degree of attention, even if it can provide a very useful piece of information [25] that WSD can exploit to perform better. Indeed, [57] proved that word senses follow a very skewed distribution and Agirre and Martinez [58] and Postma et al. [14] showed how much better WSD systems perform when they exploit this type of information. The first works in this direction aimed at finding the predominant sense of a word by leveraging a thesaurus [59] and WordNet similarity measures [60]. Chan and Ng [61] and Chan and Ng [62] followed by introducing two algorithms to rank WordNet synsets. They exploited an Expectation Maximization algorithm to estimate the prior of each class (i.e. word senses) and the marginalization of the confusion matrix of the word senses. More recently, Bennett et al. [25] presented an optimised version of HDP-WSI [63] – a topic modelling based approach for word sense distribution learning – to induce the distribution of word senses, while in [17] we introduced two unsupervised approaches for learning word sense distributions. Our work differs from those above as it is, to the best of our knowledge, the first attempt to exploit sense distribution learning methods applied to the set of documents we are interested in disambiguating in order to shape the training set. The methods we proposed proved to work well across all datasets and languages hence paving the way to future research on domain-driven, supervised and multilingual Word Sense Disambiguation.

9. Conclusion

In this paper we first introduced a novel semi-supervised technique to automatically create a sense annotated corpus, and then presented a method to shape the automatically generated training set to reflect the sense distribution in a given document, or collection of documents, that we want to disambiguate.

Moreover, we subsequently showed that this same approach can be applied to any language supported by BabelNet (currently 284 languages). To the best of our knowledge, ours is the first attempt to 1) enable supervised WSD systems on

[16] http://syntagnet.org.

multiple languages, 2) adapt the training set to better reflect the sense distribution of a target document, and 3) make the pipeline to train a supervised WSD system completely human-free.

Our experiments proved that it is possible to boost multilingual WSD performance by exploiting just the learned sense distribution and, moreover, that it is possible to automatically adapt the training set to any domain-specific document without any human intervention. Results, indeed, showed that our approach is able to effectively scale over different domains and different languages. This is accomplished by looking only at the text that we wish to disambiguate, and adapting the training set of the WSD system using the learned word senses distribution.

As future work we aim at using new resources such as SyntagNet [6] and VerbAtlas [64] to integrate the additional knowledge coming from words collocations and verb frames into Train-O-Matic hence refining the quality of the generated training set, adding new sentences for words and senses that are still not covered, i.e., those with other POS tags such as verbs adjectives and adverbs.

All the training corpora and the sense distributions for English, Italian, Spanish, German and French are made available to the community at http://trainomatic.org.

Declaration of competing interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487 under the European Union's Horizon 2020 research and innovation programme.

References

[1] R. Navigli, Word sense disambiguation: a survey, ACM Comput. Surv. 41 (2) (2009) 1–69.
[2] R. Navigli, Natural language understanding: instructions for (present and future) use, in: Proceedings of IJCAI, 2018, pp. 5697–5702.
[3] C. Fellbaum (Ed.), WordNet: An Electronic Database, MIT Press, Cambridge, MA, 1998.
[4] E. Agirre, O.L. de Lacalle, A. Soroa, Random walks for knowledge-based word sense disambiguation, Comput. Linguist. 40 (1) (2014) 57–84.
[5] A. Moro, A. Raganato, R. Navigli, Entity linking meets word sense disambiguation: a unified approach, Transactions of the Association for Computational Linguistics (TACL) 2 (2014) 231–244.
[6] M. Maru, F. Scozzafava, F. Martelli, R. Navigli, SyntagNet: challenging supervised word sense disambiguation with lexical-semantic combinations, in: Proceedings of EMNLP-IJCNLP, 2019.
[7] R. Tripodi, M. Pelillo, WSD-games: a game-theoretic algorithm for unsupervised word sense disambiguation, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), 2015, pp. 329–334.
[8] D.S. Chaplot, R. Salakhutdinov, Knowledge-based word sense disambiguation using topic models, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), 2018, pp. 5062–5069.
[9] Z. Zhong, H.T. Ng, It makes sense: a wide-coverage word sense disambiguation system for free text, in: Proceedings of the ACL 2010 System Demonstrations, Association for Computational Linguistics, Uppsala, Sweden, 2010, pp. 78–83.
[10] M. Kågebäck, H. Salomonsson, Word sense disambiguation using a bidirectional LSTM, in: Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon, CogALex@COLING 2016, Osaka, Japan, December 12, 2016, 2016, pp. 51–56.
[11] D. Yuan, J. Richardson, R. Doherty, C. Evans, E. Altendorf, Semi-supervised word sense disambiguation with neural models, in: Proceedings of COLING, 2016, pp. 1374–1385.
[12] A. Raganato, C. Delli Bovi, R. Navigli, Neural sequence learning models for word sense disambiguation, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, 2017, pp. 1156–1167.
[13] M.T. Pilehvar, R. Navigli, A large-scale pseudoword-based evaluation framework for state-of-the-art word sense disambiguation, Comput. Linguist. 40 (4) (2014) 837–881.
[14] M. Postma, R.I. Bevia, P. Vossen, More is not always better: balancing sense distributions for all-words word sense disambiguation, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 3496–3506.
[15] T. Pasini, R. Navigli, Train-O-Matic: large-scale supervised word sense disambiguation in multiple languages without manual training data, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 78–88.
[16] T. Pasini, F. Elia, R. Navigli, Huge automatically extracted training-sets for multilingual word sense disambiguation, in: Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018, 2018, pp. 1694–1698.
[17] T. Pasini, R. Navigli, Two knowledge-based methods for high-performance sense distribution learning, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, 2018, pp. 5374–5381.
[18] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Netw. ISDN Syst. 30 (1–7) (1998) 107–117.
[19] M.T. Pilehvar, D. Jurgens, R. Navigli, Align, disambiguate and walk: a unified approach for measuring semantic similarity, in: Proceedings of ACL, 2013, pp. 1341–1351.
[20] M.T. Pilehvar, N. Collier, De-conflated semantic representations, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Austin, TX, 2016, pp. 1680–1690.
[21] P.A. Lofgren, S. Banerjee, A. Goel, C. Seshadhri, Fast-ppr: scaling personalized pagerank estimation for large graphs, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 1436–1445.
[22] R. Navigli, S.P. Ponzetto BabelNet, The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, Artif. Intell. 193 (2012) 217–250.

[23] J. Camacho-Collados, M.T. Pilehvar, R. Navigli, NASARI: a novel approach to a semantically-aware representation of items, in: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, , DenverColorado, 2015, pp. 567–577.

[24] J. Camacho-Collados, M.T. Pilehvar, R. Navigli, Nasari: integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities, Artif. Intell. 240 (2016) 36–64.

[25] A. Bennett, T. Baldwin, J.H. Lau, D. McCarthy, F. Bond, LexSemTm: a semantic dataset based on all-words unsupervised sense distribution learning, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1513–1524.

[26] J. Camacho-Collados, R. Navigli, BabelDomains: large-scale domain labeling of lexical resources, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, vol. 2, 2017, pp. 223–228.

[27] M. Ziemski, M. Junczys-Dowmunt, B. Pouliquen, The United Nations parallel corpus v1.0, in: N.C.C. Chair, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis (Eds.), Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), European Language Resources Association (ELRA), Portoroz, Slovenia, ISBN 978-2-9517408-9-1, 2016, pp. 3530–3534.

[28] A. Raganato, J. Camacho-Collados, R. Navigli, Word sense disambiguation: a unified evaluation framework and empirical comparison, in: Proceedings of EACL, Valencia, Spain, 2017, pp. 99–110.

[29] P. Edmonds, S. Cotton, SENSEVAL-2: overview, in: The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems, Association for Computational Linguistics, 2001, pp. 1–5.

[30] B. Snyder, M. Palmer, The English all-words task, in: Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, Barcelona, Spain, 2004, pp. 41–43.

[31] S.S. Pradhan, E. Loper, D. Dligach, M. Palmer, SemEval-2007 task 17: English lexical sample, SRL and all words, in: Proceedings of the 4th International Workshop on Semantic Evaluations, 2007, pp. 87–92.

[32] R. Navigli, D. Jurgens, D. Vannella, SemEval-2013 task 12: multilingual word sense disambiguation, in: Proceedings of the 7Th International Workshop on Semantic Evaluation (SemEval 2013), in Conjunction with the Second Joint Conference on Lexical and Computational Semantics (*SEM 2013), Atlanta, USA, 2013, pp. 222–231.

[33] A. Moro, R. Navigli, SemEval-2015 task 13: multilingual all-words sense disambiguation and entity linking, in: Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015, 2015, pp. 288–297.

[34] G.A. Miller, C. Leacock, R. Tengi, R. Bunker, A semantic concordance, in: Proceedings of the 3rd DARPA Workshop on Human Language Technology, Plainsboro, N.J., 1993, pp. 303–308.

[35] K. Taghipour, H.T. Ng, One million sense-tagged instances for word sense disambiguation and induction, in: Proceedings of the 19th Conference on Computational Natural Language Learning, CoNLL 2015, Beijing, China, July 30-31, 2015, 2015, pp. 338–344.

[36] J. Camacho-Collados, C. Delli Bovi, A. Raganato, R. Navigli, A large-scale multilingual disambiguation of glosses, in: Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož Slovenia, May 23-28, 2016, 2016, pp. 1701–1708.

[37] C. Delli Bovi, J. Camacho-Collados, A. Raganato, R. Navigli, EuroSense: automatic harvesting of multilingual sense annotations from parallel text, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), vol. 2, 2017, pp. 594–600.

[38] P. Koehn, Europarl: a parallel corpus for statistical machine translation, in: MT Summit, vol. 5, 2005, pp. 79–86.

[39] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint, arXiv:1301.3781.

[40] M.P. Marcus, B. Santorini, M.A. Marcinkiewicz, Building a large annotated corpus of English: the penn treebank, Comput. Linguist. 19 (2) (1993) 313–330.

[41] A. Raganato, C. Delli Bovi, R. Navigli, Automatic construction and evaluation of a large semantically enriched Wikipedia, in: Proceedings of IJCAI, New York City, NY, USA, 2016, pp. 2894–2900.

[42] S.L. Manion, R. Sainudiin, An iterative "sudoku style" approach to subgraph-based word sense disambiguation, in: Proceedings of the Third Joint Conference on Lexical and Computational Se- Mantics (*SEM 2014), Dublin, Ireland, 2014, pp. 40–50.

[43] O. Melamud, J. Goldberger, I. Dagan, Context2vec: learning generic context embedding with bidirectional LSTM, in: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016, 2016, pp. 51–61.

[44] F. Luo, T. Liu, Q. Xia, B. Chang, Z. Sui, Incorporating glosses into neural word sense disambiguation, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, 2018, pp. 2473–2482.

[45] S. Kumar, S. Jat, K. Saxena, P. Talukdar, Zero-shot word sense disambiguation using sense definition embeddings, in: Proceedings of ACL, 2019, pp. 5670–5681.

[46] A. Otegi, N. Aranberri, A. Branco, J. Hajic, S. Neale, P. Osenova, R. Pereira, M. Popel, J. Silva, K. Simov, et al., QTLeap WSD/NED corpora: semantic annotation of parallel corpora in six languages, in: Proceedings of the 10th Language Resources and Evaluation Conference, LREC, 2016, pp. 3023–3030.

[47] P. Koehn, Europarl: a parallel corpus for statistical machine translation, in: MT Summit, vol. 5, 2005, pp. 79–86.

[48] E. Agirre, A. Branco, M. Popel, K. Simov, Europarl QTLeap WSD/NED Corpus, LINDAT/CLARIN Digital Library at the Institute of Formal and Applied Linguistics, Charles University in Prague, 2015.

[49] B. Scarlini, T. Pasini, R. Navigli, Just "OneSeC" for producing multilingual sense-annotated data, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, 2019, pp. 699–709.

[50] E. Agirre, O.L. de Lacalle, Publicly available topic signatures for all WordNet nominal senses, in: Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, Lisbon, Portugal, May 26-28, 2004, 2004, pp. 1123–1126.

[51] J. Fernández, M. Castillo Valdés, G. Rigau Claramunt, J. Atserias Batalla, J. Tormo, Automatic acquisition of sense examples using exretriever, in: IBERAMIA Workshop on Lexical Resources and the Web for Word Sense Disambiguation, 2004, pp. 97–104.

[52] C. Leacock, G.A. Miller, M. Chodorow, Using corpus statistics and WordNet relations for sense identification, Comput. Linguist. 24 (1) (1998) 147–165.

[53] M. Lesk, Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone, in: Proceedings of the 5th Annual Conference on Systems Documentation, Toronto, Ontario, Canada, 1986, pp. 24–26.

[54] S. Banerjee, T. Pedersen, An adapted Lesk algorithm for word sense disambiguation using WordNet, in: International Conference on Intelligent Text Processing and Computational Linguistics, Springer, 2002, pp. 136–145.

[55] A. Kilgarriff, J. Rosenzweig, Framework and results for English SENSEVAL, Comput. Humanit. 34 (1–2) (2000) 15–48.

[56] F. Vasilescu, P. Langlais, G. Lapalme, Evaluating variants of the lesk approach for disambiguating words, in: Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, Lisbon, Portugal, May 26-28, 2004, 2004, pp. 633–636.

[57] A. Kilgarriff, How dominant is the commonest sense of a word? in: Text, Speech and Dialogue, 7th International Conference, TSD 2004, Brno, Czech Republic, September 8-11, 2004, Proceedings, 2004, pp. 103–112.

[58] E. Agirre, D. Martinez, Unsupervised WSD based on automatically retrieved examples: the importance of bias, in: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004, pp. 25–32.

[59] S. Mohammad, G. Hirst, Determining word sense dominance using a thesaurus, in: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, Trento, Italy, 3–7 April 2006, 2006, pp. 121–128.

[60] D. McCarthy, R. Koeling, J. Weeds, J. Carroll, Finding predominant senses in untagged text, in: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, 21–26 July 2004, 2004, pp. 280–287.

[61] Y.S. Chan, H.T. Ng, Word sense disambiguation with distribution estimation, in: IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005, pp. 1010–1015.

[62] Y.S. Chan, H.T. Ng, Estimating class priors in domain adaptation for word sense disambiguation, in: ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17–21 July 2006, 2006, pp. 89–96.

[63] J.H. Lau, P. Cook, D. McCarthy, S. Gella, T. Baldwin, Learning word sense distributions, detecting unattested senses and identifying novel senses using topic models, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, 2014, pp. 259–270.

[64] A. Di Fabio, S. Conia, R. Navigli, VerbAtlas: a novel large-scale verbal semantic resource and its application to semantic role labeling, in: Proceedings of EMNLP-IJCNLP, 2019.