A Brief Survey of Semi-Supervised Learning and Its Application in Natural Language Processing

Konstantinos Christopher Tsiolis
McGill University
kc.tsiolis@mail.mcgill.ca

Abstract

We provide a survey of the motivations, key notions, methods, and theoretical results that underpin semi-supervised learning. Since work on learning from labelled and unlabelled data is extensive and spans several decades, this work highlights specific examples of semi-supervised methods and theoretical analyses of them, rather than providing a broad overview of the topic as a whole. We discuss self-training, generative models, discriminative models, and word embeddings. We highlight the relevance of these methods to natural language processing, an example of an area where manual annotation of data is expensive and inter-annotator agreement is often low.

1 Introduction

In machine learning, the paradigm of supervised learning involves training a model on a set S of examples drawn i.i.d. from a set $\mathcal{X}$ which all possess labels in some set $\mathcal{Y}$. (In the case of classification, on which we focus our attention, $\mathcal{Y} = \{c_1, \ldots, c_n\}$). This paradigm has been applied successfully in a wide variety of domains, leading to rapid progress in solving previously difficult tasks. For example, in natural language processing (NLP), there exist large text corpora for the tasks of part-of-speech (POS) tagging and syntactic parsing, such as the Penn Treebank Wall Street Journal corpus (Marcus et al., 1993). Supervised classifiers have leveraged this large dataset to achieve impressive results on these tasks (Chen and Manning, 2014; Huang et al., 2015).

However, the requirement that the dataset be labelled is a limiting one, as there is a wealth of unlabelled real-world data. Though there may be large tagged and parsed corpora for English, this is not the case for all languages. Furthermore, such an annotation process is extremely expensive. As is discussed in Marcus et al. (1993), the development of the Penn Treebank was a years-long process which involved the manual annotation by experts who at the very least were graduate students in linguistics. More recent approaches for dataset creation, such as Williams et al. (2018), aim to circumvent this problem with crowdsourcing, where (non-expert) workers are asked to manually annotate examples and, in some cases, write sentences. Though this allows for the creation of large datasets in a comparatively inexpensive manner, this can be hampered by the lack of expert annotations and potentially low inter-annotator agreement.

Hence, for tasks where minimal labelled data is available, it is desirable to use methods which can incorporate large amounts of unlabelled data (which can be obtained in a much more straightforward fashion, such as by crawling the web). It is here that the semi-supervised learning paradigm proves to be quite useful. In semi-supervised learning, a model is trained on a set S_l of m_l labelled examples drawn i.i.d. from some distribution $\mathcal{D}$ on $\mathcal{X}$ as well as a (usually much larger) set S_u of m_u unlabelled examples drawn i.i.d. from $\mathcal{D}$. We write the training set as $S := S_l \cup S_u$ and define $m := m_l + m_u$. The aim is to obtain a more performant model than one that is solely trained on the labelled data S_l.

As is discussed in Zhu (2005), this aim can be achieved because the unlabelled data S_u provides information about the structure of the data. More precisely, S_u provides information about the distribution of features $p(\mathbf{x})$. Though this is not the main distribution of interest for classification (i.e. $p(y \mid \mathbf{x})$), it can still influence the choice of classifier. For example, in the case of a support vector machine classifier,

we wish for the margin between the separating hyperplane and regions of $\mathcal{X}$ with high probability to be large (Balcan and Blum, 2010). As another example, Nigam et al. (2000) point out that, given data generated from a mixture of two normal distributions (one for class 0 and the other for class 1), it is possible to determine the parameters of each of the normal distributions from unlabelled data. From there, a small amount of labelled data suffices to determine which distribution corresponds to class 0 and which corresponds to class 1.

In this work, we focus most of our attention on semi-supervised learning approaches developed prior to the advent of deep learning (though we also provide a treatment of the latter at the end of this work in our discussion of word embeddings). The approaches that we discuss fall into two categories (using the standard vocabulary in the literature): transductive semi-supervised learning and inductive semi-supervised learning. The former consists of using the labelled examples in S_l to infer the labels of examples in S_u. The latter consists of using both S_l and S_u to train a model that can perform inference on new unlabelled examples. Hence, transductive learning is used in the case where one wishes to work solely with the given data, while inductive learning is used in an attempt to generalize to unseen data (Zhu, 2005).

Self-training, the first approach we discuss, can be used both as a transductive method and as an inductive method. It is an iterative procedure where a classifier is trained (in supervised fashion) on S_l, and then used to predict labels for the examples in S_u. The unlabelled examples for which the classifier outputs a label with high confidence (this will be formalized later) are then labelled and added to S_l before proceeding to the next iteration. The Yarowsky algorithm (Yarowsky, 1995) is an example of this.

Generative models model the joint distribution of features and labels, $p(\mathbf{x}, y)$, rather than modelling the conditional distribution of interest $p(y \mid \mathbf{x})$ directly. Inductive semi-supervised learning can be performed with a generative model by viewing the unknown labels in S_u as latent variables and performing expectation maximization (Dempster et al., 1977) to maximize the joint likelihood. We explore the specific example of combining naive Bayes with expectation maximization for text classification (Nigam et al., 2000).

Discriminative models, on the other hand, model $p(y \mid \mathbf{x})$ directly. In the semi-supervised case, they make use of $p(\mathbf{x})$ as a way of determining which classifiers are most "compatible" with the data. Semi-supervised support vector machines are an example of this, and PAC bounds can be stated and proved for these discriminative models, as in Balcan and Blum (2010).

Graph-based semi-supervised learning represents instances as vertices in a graph. The vertices are joined by weighted edges, where weights represent the "similarity" between two instances (Sammut and Webb, 2011). This follows the intuition that similar examples should belong to the same class.

Co-training is an ensemble method where the feature set is split into multiple subsets. For each subset, a classifier which only has access to the features in that subset is trained on the examples in S_l, with the additional constraint that its predictions on S_u should align with the other classifiers as much as possible. Nigam and Ghani (2000) discuss this approach and Dasgupta et al. (2001) provide PAC generalization bounds for it.

Neural NLP has taken these approaches out of the mainstream to a certain degree and brought pre-trained word embeddings to the forefront. Word embeddings are vector representations of words learned in an unsupervised fashion from word co-occurrence statistics. That is, they are learned from statistics which describe how frequently pairs of words appear together in an unlabelled text corpus. It has been shown empirically that the use of word embeddings leads to performance improvements on a variety of natural language processing tasks (Turian et al., 2010; Mikolov et al., 2013; Pennington et al., 2014). In this form of semi-supervised learning, unlabelled data is used to learn feature representations, which then serve as input to a supervised model. Analyses of popular word embedding algorithms have shown that in spite of their differing loss functions, many word embedding algorithms share the same objective (Levy and Goldberg, 2014b; Kenyon-Dean et al., 2020).

2 Self-Training

The self-training method for semi-supervised learning consists of using a classifier's own predictions to train it. That is, a classifier h is trained on the labelled data S_l and then used to make predictions on each instance in S_u. When making predictions, h outputs a score for each class in $\mathcal{Y}$. Scores can be normalized to sum to one and be interpreted as probabilities. For each $\mathbf{x} \in S_u$, if h assigns a score greater than some threshold value β to the class c_j, then $(\mathbf{x}, c_j)$ is added to S_l and $\mathbf{x}$ is removed from S_u. Then, this process is repeated with the new S_l and S_u obtained from the above steps.

We provide more insight into the self-training method by considering a specific example of it: the Yarowsky algorithm, which was introduced in Yarowsky (1995). In the latter work, it is presented as an algorithm for word sense disambiguation, the problem of determining the sense of a polysemous word in a given context. To use the example from the paper, the word "plant" can be used in the biological sense or to refer to a manufacturing plant.

To disambiguate a word w with possible senses $s_1, \ldots, s_n$, the Yarowsky algorithm proceeds as follows:

- Step 1: Identify all instances of the word w in the corpus, as well as its surrounding context (e.g. five words on either side of it).
- Step 2: For each s_i, identify a "seed set" $S_{l,i}$ of contexts where w takes on the sense s_i. Then, $S_l := \bigcup_{i=1}^{n} S_{l,i}$ is our set of labelled examples.
- Step 3: Train a supervised classifier h on S_l. (Yarowsky (1995) uses an algorithm called the "decision list algorithm", the details of which are interesting from the perspective of solving word sense disambiguation, but not as much from the perspective of theoretical analysis).
- Step 4: Make predictions on the remaining data in the corpus (i.e. S_u) using h. Instances where the predicted label j is given probability greater than some threshold β are added to the seed set s_j.
- Step 5: For each document d in the corpus, if an instance of w has been tagged with a sense s_j, tag all other instances of w in d with s_j.
- Step 6: Repeat Steps 3-5 with the new seed sets. Stop when no new instances are added to the seed set in an iteration.
- Step 7: In the case of inductive semi-supervised learning, the trained classifier h can now be applied to new data.

In Yarowsky (1995), this algorithm is shown empirically to be competitive with fully supervised methods.

Haffari and Sarkar (2007) provide an interpretation of the Yarowsky algorithm. They first pose the algorithm in a more general way and make slight modifications (which we discuss at the end of the section after establishing the notation). Then, they proceed to show that the algorithm is in fact minimizing a modified cross-entropy loss that is defined in terms of the Bregman divergence. We provide an overview and an interpretation of their result.

Given discrete probability distributions p, q (with support of size n) and a strictly convex real-valued function ψ, the associated Bregman divergence is defined as

$$B_\psi(p, q) = \sum_{i=1}^{n} \psi(p_i) - \psi(q_i) - \psi'(q_i)(p_i - q_i). \tag{1}$$

The authors also define a modified version of the Shannon entropy called the ψ-entropy (once again for a strictly convex real-valued function ψ):

$$H_\psi(p) = -\sum_{i=1}^{n} \psi(p_i). \tag{2}$$

Using the above two definitions, they define the ψ-cross entropy as

$$H_\psi(p \,||\, q) = H_\psi(p) + B_\psi(p, q). \tag{3}$$

This is a generalization of the standard notion of cross-entropy, as taking the convex function $\psi(t) = t \log t$ yields

$$\begin{aligned} H_{t\log t}(p \,||\, q) &= H_{t\log t}(p) + B_{t\log t}(p, q) \\ &= -\sum_{i=1}^{n} \big[p_i \log(p_i)\big] + \sum_{i=1}^{n} \big[p_i \log(p_i) - q_i \log(q_i) - \big(1 + \log(q_i)\big)(p_i - q_i)\big] \\ &= \sum_{i=1}^{n} -p_i + q_i - p_i \log(q_i) \\ &= \sum_{i=1}^{n} -p_i \log(q_i) \\ &= H(p, q), \end{aligned} \tag{4}$$

which is the usual cross-entropy.

The proof of the following is detailed in Haffari and Sarkar (2007):

Theorem 1. *The modified Yarowsky algorithm minimizes the following:*

$$K_{t^2}(\phi, \theta) = \sum_{\mathbf{x} \in \mathcal{X}} \sum_{f \in F_\mathbf{x}} \big[H_{t^2}(\phi_\mathbf{x} \,||\, \theta_f)\big], \tag{5}$$

where $\phi_\mathbf{x}$ is the labelling distribution for an instance $\mathbf{x}$ and θ_f is the prediction distribution of a feature f. That is, for every feature f in the set of features of an instance $\mathbf{x}$ (this set of features is called $F_\mathbf{x}$), θ_f describes the probability of an instance $\mathbf{x}$ having a particular label given that it possesses the feature f. That is, $\theta_f = (\theta_{f_1}, \ldots, \theta_{f_n})$ models $\big(p(y = 1 \mid f), \ldots, p(y = n \mid f)\big)$.

In the context of the Yarowsky algorithm, the classifier parameters $\theta_f = (\theta_{f_1}, \ldots, \theta_{f_n})$ represent probability distributions of the label y given that the feature f is present.

For labelled instances $(\mathbf{x}^{(l)}, y)$, the labelling distribution $\phi_{\mathbf{x}^{(l)}}$ is simply the Dirac distribution at y, while it is unknown for unlabelled instances $\mathbf{x}^{(u)}$. Thus, $\phi_{\mathbf{x}^{(u)}}$ must be learned for all $\mathbf{x}^{(u)} \in S_u$ according to the above objective.

2.1 Discussion

This result from Haffari and Sarkar (2007) is particularly interesting because it allows for a parallel to be drawn between the semi-supervised self-training approach and supervised learning where the loss is specified by some notion of distance between probability distributions. However, there is a key difference. In supervised learning with cross-entropy loss $\sum_{\mathbf{x} \in \mathcal{X}} H(p_\mathbf{x}, q_\mathbf{x})$, $p_\mathbf{x}$ is thought of as the fixed "true" distribution of the labels given $\mathbf{x}$, while q is the model's predicted distribution. The minimization procedure in this case entails finding q that minimizes the above. However, in the semi-supervised case reflected by Equation 5, because we do not have access to $\phi_{\mathbf{x}^{(u)}}$, the "true" distribution $\phi_\mathbf{x}$ is no longer fixed, and $\phi_{\mathbf{x}^{(u)}}$ must be learned as part of the minimization procedure along with the model's predicted distribution, which is parametrized by θ.

However, this result leaves us with some outstanding questions. Haffari and Sarkar (2007) made two modifications to the original Yarowsky algorithm which were necessary in the proof of their result. The effect of these modifications was not verified empiricially. The modifications pertained to the function which predicts the label based on the classifier parameters θ_{fj} and the parameter update rules. Hence, an interpretation of the original Yarowsky algorithm remains elusive, and it is unclear as to whether or not the modified versions of it which are analyzed in Haffari and Sarkar (2007) achieve the same level of performance.

3 Generative Models with Expectation Maximization

Another promising breakthrough for semi-supervised learning in NLP came in the form of generative models which make use of the expectation maximization (EM) algorithm (Dempster et al., 1977), such as Nigam et al. (2000), where a naive Bayes classifier is used to maximize the joint likelihood of inputs and labels, where unknown labels are treated as latent variables. Their approach was developed specifically in the context of text classification, where a label must be predicted for a span of text. Examples of this include topic classification and sentiment analysis. We provide a summary of their approach and discuss its implications.

Rather than modelling $p(y \mid \mathbf{x})$ directly, a generative model instead models the joint distribution of inputs and labels $p(\mathbf{x}, y)$. Then, the desired conditional probability can easily be obtained from Bayes Rule:

$$p(y \mid \mathbf{x}) = \frac{p(y)p(\mathbf{x} \mid y)}{p(\mathbf{x})} = \frac{p(y)p(\mathbf{x} \mid y)}{\sum_{i=1}^{n} p(c_i)p(\mathbf{x} \mid c_i)}. \tag{6}$$

Since the denominator is identical for each class c_j, it suffices to take $\operatorname{argmax}_j p(\mathbf{x}, c_j) = \operatorname{argmax}_j p(c_j)p(\mathbf{x} \mid c_j)$ as the prediction.

In the supervised case, where $S = S_l$, the parameters θ for $p(y)$ and $p(\mathbf{x} \mid y)$ are learned by maximizing the joint likelihood over all of the documents in S (which are assumed to have been sampled i.i.d.):

$$L(\theta; S) = \prod_{(\mathbf{x},y)\in S} p_\theta(y)p_\theta(\mathbf{x} \mid y). \tag{7}$$

More precisely, the joint log likelihood is maximized:

$$\ell(\theta; S) = \sum_{(\mathbf{x},y)\in S} \log \big(p_\theta(y)p_\theta(\mathbf{x} \mid y)\big). \tag{8}$$

For the specific case of text classification, Nigam et al. (2000) represent $\mathbf{x}$ as a vector of words $(w_1, \ldots, w_m)$, where each word is assumed to be independent of the others given the label. This independence assumption is necessary for the computational efficiency of naive Bayes, though it rarely holds in practice. In spite of this, naive Bayes has still been shown empirically to achieve solid text classification performance. Additionally, Nigam et al. (2000) discuss maximum a posteriori estimation (MAP), where there is a Dirichlet prior for θ (this is done for smoothing purposes), but this detail is not essential to our forthcoming discussion of EM, so we focus on maximum likelihood.

In the semi-supervised case (i.e. $S = S_l \cup S_u$, $S_u \neq \emptyset$), maximum likelihood estimation cannot be done in the same fashion as in the above due to the class labels for instances in S_u being unknown. To remedy this, Nigam et al. (2000) employ expectation minimization, where latent indicator variables z_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$ are introduced to replace the class labels. The z_{ij} are defined such that $z_{ij} = 1$ if $y_i = c_j$ and it equals zero otherwise. Then, the authors define the "complete" likelihood to be

$$\ell_c(\theta \mid D; \mathbf{z}) = \sum_{\mathbf{x}_i \in S} \sum_{j=1}^{n} z_{ij} \log \big(p_\theta(c_j)p_\theta(\mathbf{x}_i \mid c_j)\big). \tag{9}$$

We remark that if the class label (and thus, the z_{ij}) was known for every instance, this is exactly the log likelihood of the joint distribution, as in Equation 8. Since this is not the case, the E-step of the EM algorithm replaces the z_{ij} in ℓ_c with

$$\hat{z}_{ij} := \mathbb{E}_\theta[z_{ij}] = p_\theta(y_i = c_j \mid \mathbf{x}). \tag{10}$$

Then, in the M-step, $\ell_c(\theta \mid D; \hat{\mathbf{z}})$ is maximized and new model parameters $\hat{\theta}$ are obtained. The E-step and the M-step are then repeated iteratively until convergence to obtain a maximum likelihood estimate of the model parameters θ. (Note that prior to the first iteration, θ is initialized as the maximum likelihood estimate on the labelled data only). A proof that the EM algorithm indeed converges to a maximum likelihood estimate of θ is provided in Wu (1983).

The authors caution that the above generative model is limited by a simplifying assumption. Indeed, this formulation assumes that D is produced by a mixture model, where each mixture component corresponds exactly to one of the class labels. The authors alleviate this problem by down-weighting the unlabelled examples in ℓ_c, since, in practice, there are far more unlabelled examples than labelled ones and only the labelled examples carry information about the label distribution. Additionally, they provide a general version of the model which allows for multiple mixture components per class.

Experimental results on various text classification benchmarks show that the semi-supervised EM approach does lead to performance gains (and even greater gains with the modified approach), especially when the amount of available labelled data is small. As this amount increases, entirely supervised approaches are competitive with the semi-supervised approach. In fact, the authors do remark that in select cases where the amount of labelled data was high, performance degradation was observed for the unmodified EM.

3.1 Discussion

These results leave much to ponder from a theoretical perspective. Interestingly, the naive Bayes model performs well in practice in spite of its independence assumption. And yet, the unmodified EM, which makes additional assumptions on how the data was generated, does not consistently perform well. There is no theoretical explanation as to why this is observed.

In addition, we can draw parallels between the generative modelling with EM approach and the self-training approach discussed in the previous section. In both cases, an iterative procedure is employed, where the model's own predictions on the unlabelled data are used to train it. Indeed, in the E-step of EM, for each unlabelled instance, the class labels are weighted probabilistically based on the model parameters θ (this is reflected by $\mathbb{E}_\theta[z_{ij}]$). For example, if for some unlabelled instance $\mathbf{x}_i \in S_u$, we have $p(c_j \mid \mathbf{x}; \theta) \approx 1$, then $\mathbb{E}_\theta[z_{ij}] \approx 1$, and so the term in the complete likelihood ℓ_c corresponding to the instance $\mathbf{x}_i$ is dominated by $\log\big(p(c_j;\theta)p(\mathbf{x}_i \mid c_j;\theta)\big)$. And so, in the M-step, when ℓ_c is maximized, the instance $\mathbf{x}_i$ can essentially be viewed as a labelled training instance $(\mathbf{x}_i, c_j)$. This is similar to the Yarowsky algorithm, which labels the instances on which the classifier is most confident and adds them to the seed set.

Taking the comparison further, we observe that the latent variables z_{ij} play a similar role to $\phi_{\mathbf{x}}$ in the analysis of the Yarowsky algorithm. Indeed, the z_{ij} are defined as indicators for the class labels. In the modified Yarowsky algorithm, $\phi_{\mathbf{x}}$ serves the same purpose. In both cases, the values of $\mathbf{z}$ and ϕ are known and fixed on the labelled examples. They are unknown on the unlabelled examples and are thus estimated by the model's own predictions. Equations 5 and 9 reflect that the goal to bring the model parameters θ of the model distribution in line with the true distribution (parametrized by ϕ and z, respectively).

4 Discriminative Models

In contrast to generative methods, discriminative methods model the conditional distribution $p(y \mid \mathbf{x})$ directly. In the supervised setting, there exist several well-known discriminative models, such as logistic regression, support vector machines, and feedforward neural networks. In this section, we explore discriminative semi-supervised learning models and, especially, we explore the PAC learning bounds derived for these methods by Balcan and Blum (2010).

Discriminative semi-supervised learning can only lead to improvements over supervised learning if it is formulated in the correct way. The goal is to use insights from $p(\mathbf{x})$ to aid in the inference of $p(y \mid \mathbf{x})$. However, Seeger (2000) points out that since $p(y \mid \mathbf{x})$ decomposes as in Equation 6, $p(\mathbf{x})$ has no bearing on $p(y \mid \mathbf{x})$ (the denominator $p(\mathbf{x})$ is the same for all class labels). Thus, Balcan and Blum (2010) provide an alternative perspective, where $p(\mathbf{x})$ is instead used as part of a notion of "compatibility" that determines which hypotheses in the hypothesis space $\mathcal{H}$ best align with the structure of the data.

For example, Balcan and Blum (2010) and Zhu (2005) discuss semi-supervised support vector machines, where an additional term is added to the supervised loss that encourages the margin between the separating hyperplane and the unlabelled examples to as large as possible. The result is an overall loss

function which "trades off" the loss on the labelled data and the loss on the unlabelled data:

$$L(h) = \underbrace{\sum_{i=1}^{m_l} \max\big(0, 1 - y_i h(\mathbf{x}_i)\big)}_{\text{supervised term}} + \underbrace{\lambda_1 \sum_{i=n_l+1}^{m} \max\big(0, 1 - |h(\mathbf{x}_i)|\big)}_{\text{unsupervised term}} + \underbrace{\lambda_2 ||\mathbf{w}||_2^2}_{\text{regularization term}}, \tag{11}$$

where $\mathbf{w}$ is the vector of parameters of h.

The unsupervised term is extremely similar to the supervised term, with the only difference being that it lacks information from a label and it assumes that the predictions made by h on the unlabelled data are correct (this is indicated by the absolute value $|\cdot|$). Hence, for the unlabelled examples, only the magnitude of the margin is considered, rather then whether or not the example is on the correct side of the margin.

For general discriminative models, Balcan and Blum (2010) define compatibility as a function $\chi : \mathcal{H} \times \mathcal{X} \to [0, 1]$. For example, in the semi-supervised SVM case, for $h \in \mathcal{H}$ and $\mathbf{x} \in \mathcal{X}$, the compatibility function used is $\chi(h, \mathbf{x}) = \max\big(0, 1 - |h(\mathbf{x})|\big)$. This definition is then extended to distributions $\mathcal{D}$ over $\mathcal{X}$, where $\chi(h, \mathcal{D}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\chi(h, \mathbf{x})]$. For samples $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \sim \mathcal{D}^m$, $\chi(h, S)$ is defined to be $\frac{1}{m}\sum_{i=1}^{m} \chi(h, \mathbf{x}_i)$.

Now, suppose that we have a sample $S = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ drawn i.i.d. from a distribution $\mathcal{D}$ over $\mathcal{X}$. Let S_l be the subset of labelled instances in S and define $m_l := |S_l|$. Similarly define S_u and m_u for the unlabelled instances. Let h^* be the target hypothesis which assigns the true labels to each of the points in the sample. Then, for a candidate hypothesis, h, the labelled error is defined in the same way as in the supervised case:

$$\mathrm{err}(h) := \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\big[\mathbb{I}\big(h(\mathbf{x}) \neq h^*(\mathbf{x})\big)\big], \tag{12}$$

and the empirical labelled error $\widehat{\mathrm{err}}$ is simply the sample mean of this quantity using the sample S.

Now, the unlabelled error is defined using the compatibility function:

$$\mathrm{err}_{unl}(h) := \chi(h, \mathcal{D}), \tag{13}$$

and the empirical unlabelled error is simply $\widehat{\mathrm{err}}_{unl}(h) = \chi(h, S)$.

Furthermore, Balcan and Blum (2010) define the following quantity for $t \in [0, 1]$:

$$\mathcal{H}_{\mathcal{D},\chi}(t) := \{h \in \mathcal{H} : \mathrm{err}_{unl}(h) \leq t\}, \tag{14}$$

which is the set of hypotheses with unlabelled error at most t. For the sample S, $\mathcal{H}_{\mathcal{D},\chi}$ is defined analogously using the empirical unlabelled error.

We are now ready to discuss the first result proven by Balcan and Blum (2010).

Theorem 2. *Let $\delta, \varepsilon > 0$. Suppose $\mathcal{H}$ is a finite set, $h^* \in \mathcal{H}$, and $\mathrm{err}_{unl}(h^*) = 0$. If*

$$m_u = \frac{1}{\varepsilon}\left[\log(|\mathcal{H}|) + \log\left(\frac{2}{\delta}\right)\right]$$

and

$$m_l = \frac{1}{\varepsilon}\left[\log\left|\mathcal{H}_{\mathcal{D},\chi}(\varepsilon)\right| + \log\left(\frac{2}{\delta}\right)\right],$$

then a hypothesis $h \in \mathcal{H}$ with $\widehat{\mathrm{err}}(h) = 0$ and $\widehat{\mathrm{err}}_{unl}(h) = 0$ has $\mathrm{err}(h) \leq \varepsilon$ with probability at least $1 - \delta$.

This theorem makes very strong assumptions, namely the realizability assumption, the assumption that the hypothesis space $\mathcal{H}$ is finite, and the assumption that the target classifier h^* is perfectly compatible with $\mathcal{D}$ (i.e. it has zero unlabelled error). However, the result has an intuitively satisfying interpretation, as it tells us that the number of unlabelled examples grows with the (logarithm of) the size of $\mathcal{H}$, while the number of labelled examples grows with the size of the set of hypotheses that have unlabelled error at most ε, which may be significantly smaller than the original hypothesis set size. Hence, this formalizes the aforementioned idea that the unlabelled examples allow for a restriction to a subset of "reasonable" hypotheses which respect the structure of the data. This is also reflected in the proof.

Proof of Theorem 2. We fill in the details in the proof given in Balcan and Blum (2010).

Given $h \in \mathcal{H}$ with $\mathrm{err}_{unl}(h) > \varepsilon$, the probability that $\widehat{\mathrm{err}}_{unl}(h) = 0$ is

$$\begin{aligned} P\big(\widehat{err}_{unl}(h) = 0\big) &\leq (1-\varepsilon)^{m_u} \\ &\leq e^{-m_u \varepsilon} \\ &= \exp\left[-\log(|\mathcal{H}|) - \log\left(\frac{2}{\delta}\right)\right] \\ &= \frac{\delta}{2|\mathcal{H}|}. \end{aligned}$$

Then, by the union bound, the probability that some $h \in \mathcal{H}$ has this property is at most $\frac{\delta}{2}$. That is, the probability that there exists $h \in \mathcal{H}$ with $\widehat{\mathrm{err}}_{unl}(h) = 0$ and $h \notin \mathcal{H}_{\mathcal{D},\chi}(\varepsilon)$ is at most $\frac{\delta}{2}$. Furthermore, for all $h \in \mathcal{H}_{\mathcal{D},\chi}$ with $\widehat{\mathrm{err}}(h) = 0$, then we can apply the same strategy as in the above to conclude that $\mathrm{err}(h) > \varepsilon$ with probability at most $\frac{\delta}{2}$. Hence, by the union bound, given $h \in \mathcal{H}$ with $\widehat{\mathrm{err}}(h) = 0$ and $\widehat{\mathrm{err}}_{unl}(h) = 0$, the probability that $h \notin \mathcal{H}_{\mathcal{D},\chi}(\varepsilon)$ or that $h \in \mathcal{H}_{\mathcal{D},\chi}$ but $\mathrm{err}(h) > \varepsilon$ is at most δ. □

In the case of infinite hypothesis spaces, learning bounds involve a notion of complexity of $\mathcal{H}$ rather than the size of $\mathcal{H}$, as in the supervised setting.

Balcan and Blum (2010) define the quantity:

$$\hat{\varepsilon}_t = \sqrt{\frac{24}{m_l} \log\big(8\mathcal{H}_{S,\chi}(t)[m_l, S]\big)},$$

where $\mathcal{H}_{S,\chi}(t)[m_l, S]$ is the number of partitions of the m_l labelled points of S by hypothesis in $\mathcal{H}_{S,\chi}$. Note that this is increasing in t.

Then, Balcan and Blum (2010) prove the following agnostic result:

Theorem 3. *Let* $h_t^* = \mathrm{argmin}_{h' \in \mathcal{H}}[\mathrm{err}(h') : \mathrm{err}_{unl}(h') \leq t]$ *and define* $\hat{\varepsilon}(h') = \hat{\varepsilon}_{t'}$ *for* $t' = \hat{\mathrm{err}}_{unl}(h')$. *Let*

$$h = \mathrm{argmin}_{h' \in \mathcal{H}}[\widehat{\mathrm{err}}(h') + \hat{\varepsilon}(h')].$$

Then,

$$\mathrm{err}(h) \leq \min_t \left[\mathrm{err}(h_t^*) + \hat{\varepsilon}(h_t^*)\right] + 5\sqrt{\frac{\log(8/\delta)}{m_l}}.$$

Unlike the previous result, the above does not assume realizability or a bound on $\mathrm{err}(h^*)$. Here, the chosen hypothesis h is obtained by minimizing a loss which trades off the empirical labelled error $\widehat{\mathrm{err}}(h)$ and a term which is increasing in the unlabelled error $\widehat{\mathrm{err}}_{unl}(h)$.

4.1 Discussion

The theoretical PAC framework proposed by Balcan and Blum (2010) is quite flexible, as it simply requires that a notion of compatibility is defined. In their work, the authors show that many existing semi-supervised learning approaches fit into this framework by appropriately defining the compatibility function, including graph-based and co-training models, which were briefly discussed in Section 1. In fact, they show that generative models can even fit into this framework by defining $\chi(h, \mathbf{x}) = p_\theta(\mathbf{x})$, but they caution that this comes with the caveat that generative models make an assumption that the data being generated by a mixture model. As was mentioned in Section 3, this assumption is potentially detrimental.

With the above in mind, an interesting outstanding question is whether or not this framework can be applied to other semi-supervised methods, such as self-training. The latter method trains a model on its own most confident predictions. This does not seem to align with the notion of compatibility defined earlier in this section.

5 Word Embeddings

Particularly since the introduction of word2vec (Mikolov et al., 2013), pre-trained word embeddings (also called "distributed representations of words") have become a fixture of natural language processing. They allow for words to be embedded in a low-dimensional vector space (here, "low" is meant to mean low with respect to the size of the vocabulary), where the cosine similarity reflects the similarity between two words. For example, one would expect that the cosine similarity between the vectors for the words "cat" and "dog" is high. This phenomenon can be empirically observed with word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) embeddings, among others. Word embeddings are trained on unlabelled text corpora using word co-occurrence statistics (i.e. how often pairs of words appear together within a given context window) and then fed as input into a downstream model for a specific NLP task, which is trained in supervised fashion. Essentially, the word embeddings serve as features. We can view their incorporation into downstream models as a form of semi-supervised learning (Turian et al., 2010).

Skip-gram with negative sampling (SGNS) (Mikolov et al., 2013), the most popular word2vec algorithm, trains two sets of vectors: context vectors $\mathbf{c}$ and target word vectors $\mathbf{w}$. The former are typically discarded after training, while the latter comprise the word embeddings used for downstream tasks. At a high level, the objective of SGNS is for $\langle \mathbf{w}_i, \mathbf{c}_j \rangle$ to be high for pairs of words (i, j) which occur together (i.e. co-occur) frequently and to be low for pairs which do not. A target word and a context word are said to co-occur if the context word appears in a window of fixed size L on either side of the target word in the text. In applications, $L = 5$ is typically used, though SGNS can be refined to accomodate arbitrary notions of coocurrence (Levy and Goldberg, 2014a). Then, the function being maximized by SGNS for a single co-occurrence (i, j) is the following:

$$\log \sigma(\langle \mathbf{w}_i, \mathbf{c}_j \rangle) + \sum_{\ell=1}^{k} \mathbb{E}_{\mathbf{c}_\ell \sim U} \big[\log \sigma(-\langle \mathbf{w}_i, \mathbf{c}_\ell \rangle) \big],$$

where σ is the sigmoid function and U is the unigram distribution (i.e. the probability distribution over individual words, which is estimated from word frequencies in the corpus). The first term reflects the desire for the dot product between $\mathbf{w}_i$ and $\mathbf{c}_j$ to be high, while the second term is a "negative sampling term" which reflects the desire to drive down the dot product between $\mathbf{w}_i$ and context vectors for words which sparingly co-occur with j. The number of negative samples k is a model hyperparameter.

Then, summing the above over all context-target word pairs, (Levy and Goldberg, 2014b) formulate the overall objective function as:

$$L = \sum_{\mathbf{w}_i} \sum_{\mathbf{c}_j} N_{ij} \big(\log \sigma(\langle \mathbf{w}_i, \mathbf{c}_j \rangle) + k \cdot \mathbb{E}_{\mathbf{c}_\ell \sim U} \big[\log \sigma(-\langle \mathbf{w}_i, \mathbf{c}_\ell \rangle) \big] \big),$$

where N_{ij} is the number of times the words i and j co-occur in the corpus.

As pointed out by Levy and Goldberg (2014b), if two distinct words a and b appear in the same contexts, then their target word vectors $\mathbf{w}_a$ and $\mathbf{w}_b$ have high dot products with the same context vectors. From this, it seems intuitive that $\langle \mathbf{w}_a, \mathbf{w}_b \rangle$ will also be high. This has been verified empirically for SGNS and is in line with the "distributional hypothesis", which states that "similar words appear in similar contexts". However, the authors were unaware of any theoretical proof that dot products between target word vectors reflect word similarity as defined by the distributional hypothesis, and, to our knowledge, this remains an open question.

However, Levy and Goldberg (2014b) prove the following:

Theorem 4. *Let all $\mathbf{w}_i, \mathbf{c}_j$ be such that the objective function of SGNS is maximized. Then,*

$$\langle \mathbf{w}_i, \mathbf{c}_j \rangle = \log \left(\frac{N_{ij} \cdot N}{N_i N_j} \right) - \log k, \tag{15}$$

where N_i, N_j denote the frequencies of i and j in the corpus, respectively, and N is the total number of co-occurrences in the corpus.

The quantity $\log\left(\frac{NN_{ij}}{N_iN_j}\right)$ is the pointwise mutual information PMI of the words i and j. To interpret this, we follow (Kenyon-Dean et al., 2020) and consider

$$\text{PMI}(i,j) = \log\left(\frac{NN_{ij}}{N_iN_j}\right) = \log\left(\frac{\frac{N_{ij}}{N}}{\frac{N_i}{N}\cdot\frac{N_j}{N}}\right) = \log\left(\frac{p_{ij}}{p_ip_j}\right), \tag{16}$$

where p_{ij} is the empirical probability of observing an (i,j) co-occurrence, and p_i, p_j are the empirical probabilities of viewing word i and word j as a target word, respectively. Written this way, the PMI of i and j measures the deviation of the joint distribution of (i,j) from independence. If i and j co-occur more often than would be expected if they were independent, then the PMI is positive. Similarly, if they co-occur less often than would be expected if they were independent, then the PMI is negative. It is this notion that is being modelled by the context vector-target vector dot product $\langle \mathbf{w}_i, \mathbf{c}_j \rangle$, rather than the frequency at which i and j co-occur.

5.1 Discussion

Word embeddings are not as theoretically well-understood as the other methods presented in this work. Though Levy and Goldberg (2014b) find that SGNS has an interpretable objective, and Kenyon-Dean et al. (2020) also find that GloVe has a similar objective in terms of PMI, there remain several unanswered questions, as well as some promising signs of progress. As was previously mentioned, we are not aware of any proof that the dot product between word vectors indeed reflects semantic similarity. Furthermore, (Mikolov et al., 2013) show that SGNS embeddings have an additional notable property, often called the "analogy" property, which is best illustrated an example:

$$\mathbf{w}_{Berlin} - \mathbf{w}_{Germany} + \mathbf{w}_{France} \approx \mathbf{w}_{Paris}. \tag{17}$$

This phenomenon was not well understood until the work of Allen and Hospedales (2019), which shows how word analogy relations such as the above arise from word embeddings.

Outside of studying word embedding properties, another vital theoretical question is that of how word embeddings lead to improve downstream task performance in NLP. We are left only with the intuition that word embeddings carry information about word similarity and word co-occurrence, which may be difficult to learn when training in a supervised fashion on the downstream task alone. It is not clear if or how word embeddings can be analyzed with the insights and theoretical tools discussed in the previous sections. A key difference between semi-supervised learning with word embeddings and the other approaches that we discussed is word embeddings are trained on unlabelled data in a task-agnostic way. That is, they are not trained on unlabelled data with the objective of inferring class labels for a downstream problem. This is in stark contrast to the Yarowsky algorithm and EM in particular, which make predictions on the unlabelled data.

Another interesting question to consider for word embeddings would be to investigate the effect of changing the notion of similarity to a notion of distance between words. With this, the SGNS objective would change to one where we wish to minimize the distance between frequently co-occurring words, and where the negative sampling term seeks to maximize the distance between the target word and randomly sample words. It is likely that a clever choice of distance function would be required. The Euclidean distance, a naive choice, is unbounded, which may be problematic for maximizing the objective.

6 Conclusion

In this work, we explored various semi-supervised learning methods, namely self-training, generative models with expectation maximization, discriminative models, and word embeddings. These methods were motivated through the lens of natural language processing, where data annotation is expensive and unlabelled text data is readily available. We provided summaries of these methods, as well as of theoretical work which analyzes them. We also addressed the connections between the different methods, as well as open questions.

References

Carl Allen and Timothy Hospedales. 2019. Analogies explained: Towards understanding word embeddings. In *International Conference on Machine Learning*, pages 223–231.

Maria-Florina Balcan and Avrim Blum. 2010. A discriminative model for semi-supervised learning. *Journal of the ACM (JACM)*, 57(3):1–46.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Sanjoy Dasgupta, Michael Littman, and David McAllester. 2001. Pac generalization bounds for co-training. *Advances in neural information processing systems*, 14:375–382.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

GholamReza Haffari and Anoop Sarkar. 2007. Analysis of semi-supervised learning with the yarowsky algorithm. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 159–166.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Kian Kenyon-Dean, Edward Newell, and Jackie Chi Kit Cheung. 2020. Deconstructing word embedding algorithms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8479–8484.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Claude Sammut and Geoffrey I Webb. 2011. *Encyclopedia of machine learning*. Springer Science & Business Media.

Matthias Seeger. 2000. Learning with labeled and unlabeled data. Technical report.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

CF Jeff Wu. 1983. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

Xiaojin Jerry Zhu. 2005. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.