Web Intelligence 18 (2020) 53–67
DOI 10.3233/WEB-200428
IOS Press

Deep text classification of Instagram data using word embeddings and weak supervision[1]

Kim Hammar [*], Shatha Jaradat, Nima Dokoohaki and Mihhail Matskin
Department of Software and Computer Systems, KTH Royal Institute of Technology, Stockholm, Sweden
E-mails: kimham@kth.se, shatha@kth.se, nimad@kth.se, misha@kth.se

Abstract. With the advent of social media, our online feeds increasingly consist of short, informal, and unstructured text. Instagram is one of the largest social media platforms, containing both text and images. However, most of the prior research on text processing in social media is focused on analyzing Twitter data, and little attention has been paid to text mining of Instagram data. Moreover, many text mining methods rely on training data annotated manually by humans, which in practice is both difficult and expensive to obtain. In this paper, we present methods for weakly supervised text classification of Instagram text. We analyze a corpora of Instagram posts from the fashion domain and train a deep clothing classifier with weak supervision to classify Instagram posts based on the associated text.

With our experiments, we demonstrate that in absence of annotated training data, using weak supervision to train models is a viable approach. With weak supervision we were able to label a large dataset in hours, something that would have taken months to do with human annotators. Using the dataset labeled with weak supervision in combination with generative modeling, an F_1 score of 0.61 is achieved on the task of classifying the image contents of Instagram posts based solely on the associated text, which is on level with human performance.

Keywords: Instagram, weak supervision, word embeddings, deep learning

1. Introduction

Text processing is present in our everyday life and empowers several important utilities, such as, machine translation, web search, personal assistants, and user recommendations. Today, social media is one of the largest sources of text, and while social media fosters the development of a new type of text processing applications, it also brings with it its own set of challenges due to the informal language.

Text in social media is unstructured and has a more informal and conversational tone than text from conventional media outlets [3]. For instance, text in social media is rich of abbreviations, hashtags, emojis, and misspellings.

Traditional Natural Language Processing (NLP)-tools are designed for formal text and are less effective when applied on informal text from social media [28]. This is why recent research efforts have tried to adapt NLP tools to the social media domain [13]. Moreover, methods within the intersection of NLP and machine learning applied to social media have been successful in information extraction [29], classification [19], and conversation modeling [27].

Results of the previous work are not enough for our purposes due to the following reasons: (1) many results rely on access to massive quantities of annotated data, something that is not available in our domain; (2) most of the work is focused on Twitter, with little

[1] This paper is an extended version of our conference paper "Deep Text Mining of Instagram Data Without Strong Supervision" published in 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI).

[*] Corresponding author. E-mail: kimham@kth.se.

attention to image sharing platforms like Instagram;[2] and (3) to the best of our knowledge, no prior assessment of complex, multi-label, classification in social media has been made.

Acquisition of annotated data that is accurate and can be used for training text classification models is expensive. Especially in a shifting data domain like social media. In this research, we explore the boundaries of text mining methods that can be effective without this type of strong supervision. In particular, we evaluate text classifiers trained with a programmatic type of supervision referred to as *weak supervision*.

Even if we assume that the main research results from Twitter will be useful in our research on Instagram, we still should take into account several important differences between the two domains. The most prevalent discrepancies are that Instagram is an image-sharing medium while Twitter is a micro-blogging medium, and that Twitter has a character-limit per tweet.

In this paper, we focus on the task of classifying Instagram posts into clothing categories based on the associated text (Fig. 1), it is an extension of a previous conference paper [15]. The work presented in this paper is part of a larger research project. The project aspires to improve the state-of-the-art in fashion recommendation by employing activities in social media

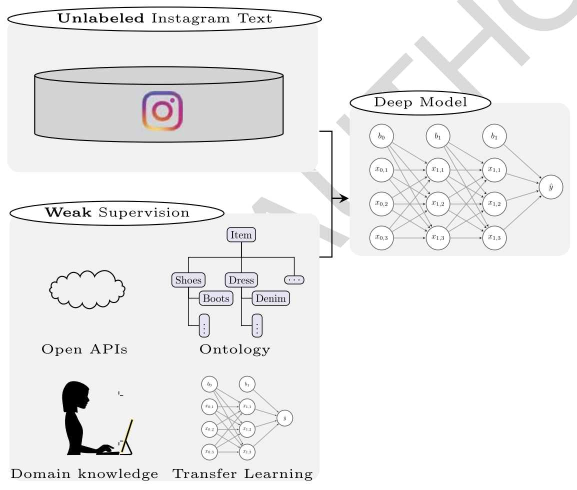


Fig. 1. In this paper we investigate methods for training a text classifier without manually labeled data (strong supervision). Instead, we use algorithmic labeling of a large text corpora from the fashion domain on Instagram, and use that dataset to train a deep model to classify Instagram posts into fashion categories.

[2] Instagram.com

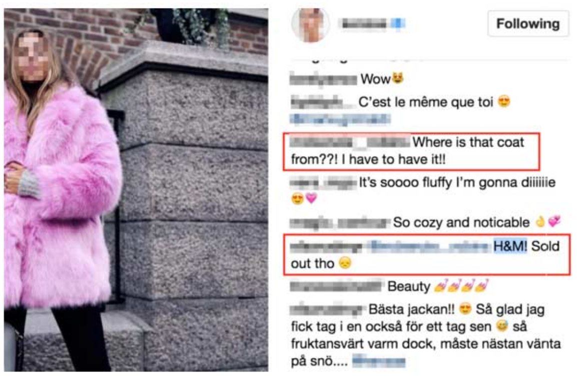
and using data crossing multiple domains in the recommendations [17]. In future work, the text processing methods presented in this paper will be integrated with computer vision models in the project.

Just as other consumption-driven industries, the fashion industry has been influenced by the emergence of social media. Social media is progressively getting more attention by fashion brands and retailers as a source for detecting trends, adapting user recommendations, and for marketing purposes [4]. To give an example, the image-sharing platform Instagram has become a popular medium for fashion branding and community engagement [1]. This is why extraction and classification of fashion attributes on Instagram is an important task for several modern applications working with user recommendation and detection of fashion trends.

In addition to hosting images, Instagram contains large volumes of user generated text. Specifically, an Instagram post can be associated with an image caption written by the author of the post, by comments written by other users, and by "tags" in the image that refer to other users. Despite being a platform rich of text, little prior work has paid attention to the promising applications of text mining on Instagram. From our case study on Instagram posts in the fashion community, it was revealed that the text often *indicates* the clothing on the associated image, an example of this is given in Fig. 2. We believe that there is a value in the text on Instagram that currently is unutilized. For example, the text on Instagram can be mined and used for predictive modeling and analytics.

Our contribution in this paper includes:

- An empirical study of Instagram text.
- An evaluation of word embeddings trained on Instagram text.


[redacted] Following

[redacted] Wow😺

[redacted] C'est le même que toi 😍

[redacted] Where is that coat from??! I have to have it!!

[redacted] It's soooo fluffy I'm gonna diiiiie 😍💗

[redacted] So cozy and noticable 👌💕

[redacted] H&M! Sold out tho 😔

[redacted] Beauty 💅💅💅💅

[redacted] Bästa jackan!! 😍 Så glad jag fick tag i en också för ett tag sen 😅 så fruktansvärt varm dock, måste nästan vänta på snö.... [redacted]

Fig. 2. An Instagram post from the fashion community.

– A novel pipeline for multi-label clothing classification of the text associated with Instagram posts using weak supervision and the *data programming* paradigm [26].

Our empirical study provides one of the few available studies on Instagram text and shows that the text is noisy, that the text distribution exhibits the long-tail phenomenon, and that comment sections on Instagram often are multi-lingual. Moreover, experimental results demonstrate that the FastText algorithm for training word embeddings [5], that can capture the morphology of words, is also suited for the noisy type of text that can be found in social media. Finally, we train a deep text classifier using weak supervision and data programming. The classifier achieves an F_1 score of 0.61 on the task of clothing prediction of Instagram posts based on the text. The accuracy of the classifier is on level with human performance on the task and beats a baseline that uses majority voting.

The rest of this paper is structured as follows. In Section 2 we describe related work, and in Section 3 we present our approach to the problem. In Section 4 we summarize the experimental setup and Section 5 contains the results from our evaluations as well as our interpretation of the results. Lastly, Section 6 includes our conclusions and suggestions for future research directions.

2. Related work

Our research extends prior work on learning domain specific word embeddings (Section 2.1) and weakly supervised text classification (Section 2.2) working with informal text from social media.

2.1. Learning domain specific word embeddings

The practice of constructing word embeddings targeted to a specific domain is a relatively new field of research as most prior research have focused on constructing generic word embeddings, not optimized to a specific domain. Prior work that resembles our effort in learning word embeddings for the fashion domain are (1) [30] introduces word embeddings for the construction domain; (2) [23] compared embeddings specific to the biomedical domain with off-the-shelf embeddings;[3] (3) [7] presents a study on the most important hyperparameters for learning embeddings for the biomedical domain; (4) [6] trains embeddings on a corpora of tweets to study transfer of sentiment across problem domains; (5) [11] presents emoji2vec, a method for learning word embeddings for emojis; and (6) [10] propose a supervised method for training embeddings, with hashtags as a supervision signal.

Similar to our experiments (1) used a domain-specific dataset for intrinsic evaluation for embeddings. However, they did not tune the hyperparameters, and their evaluation focused on a single set of off-the-shelf word embeddings. In summary, their results indicate that off-the-shelf embeddings performed comparably to domain-specific embeddings on several tasks.

In (2) an extrinsic evaluation of domain-specific embeddings was made, the evaluation compared embeddings trained on biomedical text with off-the-shelf embeddings on a classification task. Without detailed tuning of hyperparameters, the results indicated that the domain-specific embeddings only gave a modest improvement over the off-the-shelf ones.

In (3) it was found that performance of embeddings can be notably improved by tuning the hyperparameters, rather than sticking to the default values. Moreover, their results indicate that tuning of hyperparameters can be contradictory between intrinsic and extrinsic evaluations.

Our research differ from (1)–(3) by targeting noisy text from social media, rather than newswire text. The work in (4)–(6) is similar to ours in that they use word embeddings in the social domain but differ from our work in other aspects. In (4) word embeddings are trained using a corpora of tweets but the study lacks an comparison of the embeddings trained using tweets and generic embeddings trained on newswire text. (5) reports an improved accuracy when training embeddings directly on Unicode descriptions of emojis, instead of learning the embeddings on a large collection of tweets. Their motivation is, similar to ours, that off-the-shelf embeddings lack representations for many tokens that are commonplace in social media. Finally (6) use hashtags as a supervision signal when training word embeddings, this results in embeddings that are similar based on hashtags, which loses fine-grained word meanings required for the classification task studied in this paper.

[3] *Off-the-shelf* embeddings refers to pre-trained embeddings available online, trained on generic text rather than domain-specific text.

2.2. *Text classification without strong supervision*

For the task of classifying Instagram text, our research builds primarily on results from supervised machine learning. The success of this paradigm of machine learning has traditionally been coupled to annotated datasets. Notable results in supervised text classification are [18] and [9], both of which differ from our research in that they assume access to a large annotated text corpora for training the classifier.

More recently, research on exploiting *unlabeled* data for training has received attention. For certain tasks, completely unsupervised learning is enough, such as the task of learning word embeddings. For other tasks, a blend of supervised and unsupervised learning is appropriate. Semi-supervised and weakly-supervised learning are two approaches to learning with limited amount of supervision, while having access to an abundant amount of *unlabeled* data.

2.2.1. *Semi-supervised learning*

In semi-supervised learning, even though it is assumed that a smaller amount of labeled training data are available, the goal is to combine that data with a larger portion of unlabeled data. To train with unlabeled data, semi-supervised learning makes use of assumptions about the data, such as the data distribution. With the right assumptions, semi-supervised learning algorithms are able to relate the unlabeled data with the labeled data to drive the learning process.

2.2.2. *Weakly-supervised learning*

Weakly supervised learning methods rely on availability of weak-supervision signals and do not assume that any labeled data are available. A weak supervision signal can for instance be in the form of an external API, a crowdworker, or a domain heuristic. As opposed to strong supervision, weak supervision seldom has perfect accuracy or coverage.

Specifically, related to our research is the *data programming paradigm* presented in [26], the paradigm has achieved promising results on several text classification tasks. Data programming has been applied to binary and multinomial text extraction and classification tasks [25,26] and is currently being used within Google for training various classifiers [2]. To the best of our knowledge, it has neither been applied to multi-label classification tasks, nor to social media text.

3. Methodology

In Section 3.1 we outline how our analysis of the Instagram corpora was performed. Section 3.2 describes our second contribution, which is an evaluation of word embeddings trained on Instagram text. Finally, Section 3.3 presents the pipeline we used to train a deep text classifier using *weak supervision*. The code for the implementations and the trained embeddings are publicly available.[4]

3.1. *Empirical study of Instagram text*

Of special interest in our study was to elucidate how the Instagram text differs from newswire text, as it affects the choice of processing methods. We analyzed a corpora of Instagram posts by measuring the fraction of online-specific tokens, the number of Out-Of-Vocabulary (OOV) words, the number of languages in the corpora, and the text distribution.

3.2. *Learning domain-specific word embeddings*

Considering the peculiarity of Instagram text compared to newswire text, we have surveyed the benefit of training new word embeddings for the fashion domain on Instagram. We have performed an evaluation of embeddings trained on our corpora of Instagram posts using Word2vec, Glove, and FastText, with varying hyperparameters. Parameters that were not tuned in the evaluation, were kept to their default values, listed in Table 1.

To examine the difference between domain-specific word embeddings and generic word embeddings, the embeddings trained on the Instagram corpora were compared with the state-of-the-art off-the-shelf

Table 1
Default parameters used when training word embeddings

Parameter	*Value*
Iterations	15
MinCount	5
Learning rate	0.025
Learning rate update rate	100
Minimum n-gram (FastText)	3
Maximum n-gram (FastText)	6
Output layer	Hierarchical softmax
Max count (GloVe)	100

[4] https://github.com/shatha2014/FashionRec

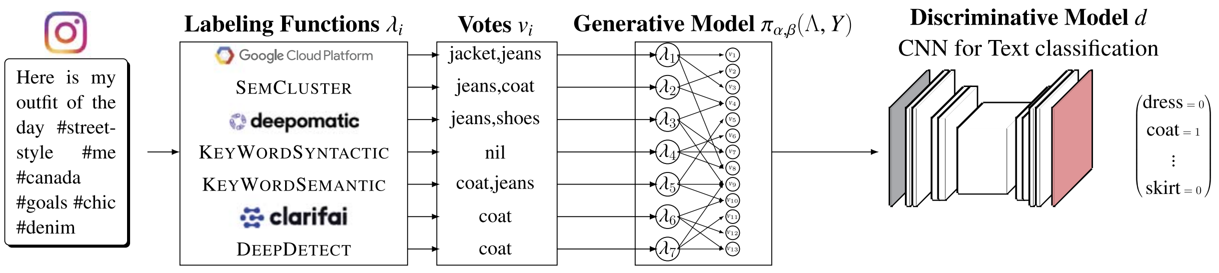


Fig. 3. A pipeline for weakly supervised text classification of Instagram posts.

embeddings, provided by Google, Facebook, and Stanford's NLP group. Specifically, the baselines were: (1) FASTTEXT-WIKI, consisting of embeddings pre-trained with the FastText algorithm on a corpus of Wikipedia articles, published by Facebook [5]; (2) WORD2VEC-GNEWS, consisting of embeddings pre-trained with the Word2vec algorithm on a corpus of Google news articles, published by Google [14]; (3) and (4) GLOVE-WIKI, and GLOVE-TWITTER, consisting of embeddings pre-trained with the GloVe algorithm respectively on a corpus of Wikipedia articles, and tweets, published by the Stanford NLP group [24].

3.3. *Clothing classification of Instagram posts*

This section presents a pipeline for weakly supervised text classification to predict clothing items in Instagram posts. The pipeline is visualized in Fig. 3 and includes steps devoted to labeling a dataset with weak supervision (Section 3.3.3), combining weak labels with data programming to produce probabilistic labels (Section 3.3.2), and training a discriminative model using the probabilistic labels (Section 3.3.5).

3.3.1. *The classification task*

Although multiple classifications are of interest in our research, such as brand classification, and fabric classification, we focus initially on the clothing item classification problem. This task is a multi-label multi-class classification problem with 13 classes. The classes are as follows: dresses, coats, blouses & tunics, bags, accessories, skirts, shoes, jumpers & cardigans, jeans, jackets, tights & socks, tops & t-shirts, and trouser & shorts.

3.3.2. *Data programming*

With the data programming paradigm [26], weak supervision is encoded with *labeling functions*. A labeling function is a black-box function $\lambda_i : x \rightarrow y \vee \emptyset$, that takes as input a training example x, and outputs a label y or abstain from labeling. A labeling function is typically realized through some domain heuristic and only labels a subset of the data. Naturally, labels produced by such functions are less accurate than labels produced by human annotators. However, weak labels can be *complementary* to each other. Several weak labels can be combined with the purpose of obtaining more accurate labels. The innovative part of data programming is the way that it learns a generative model of the labeling process in an unsupervised fashion. The parameters learned in the generative model can then be used to combine labels in a more sophisticated way than majority voting.

Formally, a labeling function λ_i has a probability β of labeling an input, and refrain from labeling an input with probability $1 - \beta$. Similarly, a labeling function has a probability α of labeling an input correctly. The combination of functions can be modeled as a generative model $\pi_{\alpha,\beta}(\Lambda, Y)$. Where Λ is the output matrix after applying all of the labeling functions to the unlabeled data ($\Lambda_{i,j} = \lambda_j(x_i)$), and Y is the true classes, modeled as latent variables.

In Λ, an empirical probability $\hat{p}_{i,j}$ that two labeling functions λ_i and λ_j agrees can be inferred. By using the observed probabilities of overlap, the accuracy of each labeling function can be *estimated* using maximum likelihood estimation. Consequently, the problem of training the generative model and finding the parameters α and β that best describe the observed overlaps among labeling functions can be phrased as the optimization problem defined in Eq. (1), where S denotes the training set [26].

$$(\hat{\alpha}, \hat{\beta}) = \arg\max_{\alpha,\beta} \sum_{x \in S} \log P_{(\Lambda,Y)\sim\pi_{\alpha,\beta}}\big(\Lambda = \lambda(x)\big) \tag{1}$$

Finally, after estimating α and β, the parameterized generative model is used to engender probabilistic (confidence-weighted) training labels $p(Y|\Lambda)$ from the unlabeled data and the output of the labeling functions. When producing the probabilistic labels, more weight is given to accurate labeling functions, and the uncertainty of each label is indicated by the probability. If labeling functions disagree on a training example, this is encoded as an uncertainty by giving the corresponding label a lower probability. Subsequently, the probabilistic training labels can be used to train a discriminative machine learning model in a supervised manner. As the labels are probabilistic and not binary, a *noise-aware* loss function is used when training a discriminative model with such labels. A noise-aware loss function is a loss function for minimizing the expected loss with respect to the probabilistic labels.

3.3.3. *Weak supervision for fashion attributes in Instagram posts*

We used seven labeling functions to label a dataset of 30K Instagram posts with fashion attributes. The purpose of using multiple functions is that we expect that the combination of functions will improve the accuracy of the supervision compared to what each function in isolation would provide. The functions are as follows.

1. λ_1, a function that uses Google's Cloud Vision API[5] to classify the image associated with the text.
2. λ_2, a function using an internally developed text mining system (SEMCLUSTER) that extracts fashion details using clustering techniques and a fashion ontology [15].
3. λ_3, a function that uses the Deepomatic[6] API for computer vision to classify the image associated with the text.
4. λ_4, a function that uses keyword matching to an ontology with fashion words, using Levenshtein distance [20].
5. λ_5, a function that uses keyword matching to an ontology with fashion words, using word embeddings and cosine similarity.
6. λ_6, a function that uses the Clarifai "Apparel" model[7] to classify the image associated with the text.
7. λ_7, a function that uses a pre-trained image-classifier provided by DeepDetect.[8]

[5]https://cloud.google.com/vision/
[6]https://www.deepomatic.com/
[7]https://www.clarifai.com/
[8]https://www.deepdetect.com/

Our choice of labeling functions are a mixture of keyword-based functions (λ_4, λ_5), heuristic based (λ_2) and model-based (λ_1, λ_3, λ_6, λ_7). The selection of labeling functions was ultimately done based on what type of supervision were available to us at the time. The category of functions (keyword-based, heuristic-based, and model-based) are similar to what is used internally at Google for training models with weak supervision and the data programming paradigm [2]. It should be clear that the kind of supervision provided by the aforementioned labeling functions is scalable and extremely cheap in comparison with supervision in the form of human annotations.

3.3.4. *Using data programming to combine multi-labels*

In the original data programming paper, a binary classification scenario is studied and it is assumed that labeling functions are binary [26]. We have extended the data programming paradigm from binary classification to the multi-label setting. To make use of the data programming paradigm for *multi-label* classification, we model the labeling process with one generative model for each class. With this approach, the combination of generative models is able to represent separate accuracy estimates of the labeling functions for each class.

Formally, the generative model $\pi_{\alpha,\beta}(\Lambda^{(k)}, Y^{(k)})$ is trained using the observed overlaps between the labeling functions applied to the unlabeled data for class k. In this notation, $\Lambda^{(k)}_{i,j} = (\lambda_j(x_i))_k$, and $Y^{(k)}$ is the truth labels for class k, modeled as latent variables. Once the generative model is trained, the parameters learned by the model are used to produce probabilistic labels $p(Y^{(k)}|\Lambda^{(k)}) \in \mathbb{R}^n \wedge p(Y^{(k)}|\Lambda^{(k)})_i \in [0, 1]$, for each class k and training example $i \in \{1, \ldots n\}$. The probabilistic labels for each class then constitute as column vectors in a matrix of probabilistic labels $p(Y|\Lambda) \in \mathbb{R}^{n\times|C|}$, that can be used to train a discriminative model in a supervised fashion (Eq. (2)).

$$p(Y|\Lambda) = \begin{pmatrix} p(Y^{(1)}|\Lambda^{(1)})_1 & \ldots & p(Y^{(|C|)}|\Lambda^{(|C|)})_1 \\ \vdots & \ddots & \vdots \\ p(Y^{(1)}|\Lambda^{(1)})_n & \ldots & p(Y^{(|C|)}|\Lambda^{(|C|)})_n \end{pmatrix} \quad (2)$$

In our experiments, we used the Snorkel[9] implementation [25] to train the generative models on the unlabeled data. For completeness, the definition of the training procedure in Snorkel is presented below.

First, the labeling functions are applied to the unlabeled data $\Lambda^{(k)}_{i,j} = (\lambda_j(x_i))_k$. Then the generative model is encoded by using a vector $\phi^{(k)}_i(\Lambda^{(k)}, Y^{(k)})$ of factors for each unlabeled data point x_i and class k. The vector contains concatenated values representing the labeling propensity (encoded with a 1 for each labeling function that labeled x_i), estimated accuracy of each labeling function (encoded with a 1 if the function agrees with the estimated label), and pairwise correlations of labeling functions (a 1 is added if two functions agree with each other). Using these vectors for each data point and labeling function, as well as a vector of model parameters $w^{(k)}$, the model can be defined as in Eq. (3) [25]. Where $Z_{w^{(k)}}$ is a normalizing constant, and m is the number of unlabeled data points.

$$p_{w^{(k)}}\left(\Lambda^{(k)}, Y^{(k)}\right) = Z^{-1}_{w^{(k)}} \exp\left(\sum_{i=1}^{m}\left(w^{(k)}\right)^T \phi^{(k)}_i\left(\Lambda^{(k)}, y^{(k)}_i\right)\right) \tag{3}$$

The parameters of the model $w^{(k)}$ are learned by minimizing the negative log marginal likelihood based on $\Lambda^{(k)}$ and the latent variables $Y^{(k)}$:

$$\hat{w}^{(k)} = \underset{w^{(k)}}{\arg\min} - \log \sum_{Y^{(k)}} p_{w^{(k)}}\left(\Lambda^{(k)}, Y^{(k)}\right) \tag{4}$$

The implementation uses an interleaving of stochastic gradient descent and Gibbs sampling to maximize the objective [25]. After training, the predictions of the model constitute as the probabilistic labels $p_{\hat{w}^{(k)}}(Y^{(k)} | \Lambda^{(k)})$ for class k.

3.3.5. Discriminative model

For the discriminative model, we have used a variant of the Convolutional Neural Network (CNN) model for text classification presented in [18]. This model was chosen as it is established as one of the best performing text classifiers for short texts. However, nearly any model could have been used, the only requirement is that the loss function can be modified to work with probabilistic labels.

[9] https://github.com/HazyResearch/snorkel

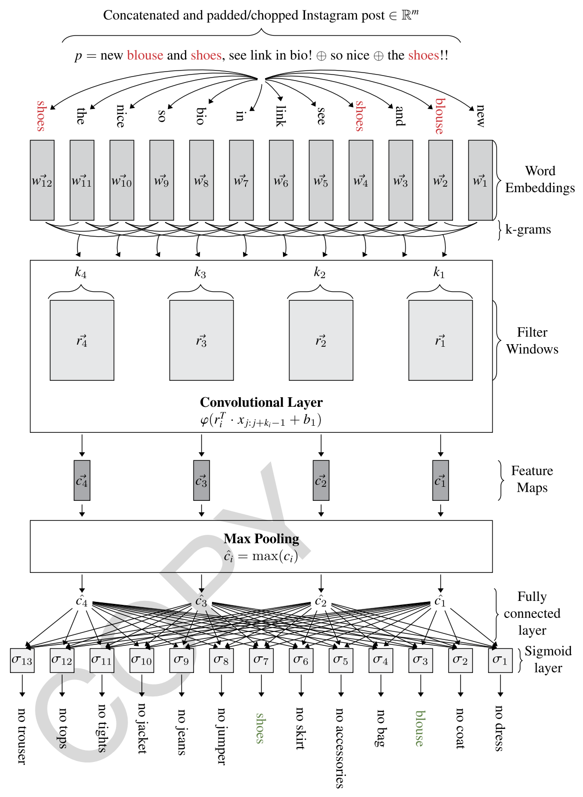

Concatenated and padded/chopped Instagram post $\in \mathbb{R}^m$
p = new blouse and shoes, see link in bio! ⊕ so nice ⊕ the shoes!!
shoes
the
nice
so
bio
in
link
see
shoes
and
blouse
new
$\vec{w}_{12}$ $\vec{w}_{11}$ $\vec{w}_{10}$ $\vec{w}_9$ $\vec{w}_8$ $\vec{w}_7$ $\vec{w}_6$ $\vec{w}_5$ $\vec{w}_4$ $\vec{w}_3$ $\vec{w}_2$ $\vec{w}_1$
Word Embeddings
k-grams
k_4 k_3 k_2 k_1
$\vec{r}_4$ $\vec{r}_3$ $\vec{r}_2$ $\vec{r}_1$
Filter Windows
Convolutional Layer
$\varphi(r_i^T \cdot x_{j:j+k_i-1} + b_1)$
$\vec{c}_4$ $\vec{c}_3$ $\vec{c}_2$ $\vec{c}_1$
Feature Maps
Max Pooling
$\hat{c}_i = \max(c_i)$
$\hat{c}_4$ $\hat{c}_3$ $\hat{c}_2$ $\hat{c}_1$
Fully connected layer
σ_{13} σ_{12} σ_{11} σ_{10} σ_9 σ_8 σ_7 σ_6 σ_5 σ_4 σ_3 σ_2 σ_1
Sigmoid layer
no trouser
no tops
no tights
no jacket
no jeans
no jumper
shoes
no skirt
no accessories
no bag
blouse
no coat
no dress

Fig. 4. CNN for multi-label text classification.

The neural network architecture in [18] consists of an embedding input layer, a convolutional layer, and a fully-connected layer of softmax or sigmoid output units. Moreover, the architecture employs max-over-time pooling to detect keywords in the input. The architecture is illustrated in Fig. 4 and defined mathematically below.

Embedding and convolutional layers Let $p = \langle w_1 \oplus w_2, \oplus \ldots \oplus w_m \rangle \in \mathbb{R}^m$ represent an Instagram post index-encoded with respect to a vocabulary V and padded or chopped off to a fixed length m. The symbol $\oplus$ denotes the concatenation operator, and w_i denotes the i-th word in the concatenated text of caption, usertags, and user comments.

The first layer is the embedding layer, that serves as a lookup step, where each word w_i is encoded as its corresponding word embedding $\vec{w}_i \in \mathbb{R}^d$, and d is the dimension of the embeddings. The embeddings are updated as part of training and can either be initialized randomly or with pre-trained embeddings. Let $x = [\vec{w}_1, \ldots, \vec{w}_m] \in \mathbb{R}^{m \times d}$ denote the output of the embedding layer E.

Next is the convolutional layer that performs two-dimensional convolutions over the sequence of embeddings. The convolutional layer consists of n filter windows $W_1 = [r_1, \dots, r_n]$ of variable sizes $[k_1, \dots, k_n]$, $r_i \in \mathbb{R}^{d \times k_i}$. Each filter is slided across the input, x, to produce new *local* feature representations $[c_1, \dots, c_n]$, called *feature maps*. During every step when traversing the input, filter r_i is applied to a k-gram of length k_i. For each k-gram, the filter applies a non-linearity φ (such as ReLu or tanh) to the weighted sum of the embeddings of the k-gram and the filter weights plus a bias term b_1, producing a scalar output v_j.

Let $x_{i:i+k-1} \in \mathbb{R}^{d \times k}$ denote a k-gram of consecutive words encoded as their embeddings $\vec{w_i}, \vec{w_{i+1}}, \dots \vec{w_{i+k-1}}$. With this notation, a filter of size k that is slided over the entire input of m words will cover the k-grams $G = \{x_{1:k}, x_{2:k+1}, \dots, x_{m-k+1:m}\} \wedge |G| = m - k + 1$. Following these definitions, the steps to compute a feature map $c_i \in \mathbb{R}^{|G|}$ with filter r_i can be defined as in Eq. (5).

$$\begin{aligned} v_j &= \varphi\big(r_i^T \cdot x_{j:j+k_i-1} + b_1\big) \\ c_i &= [v_1, \dots, v_{|G|}] \end{aligned} \tag{5}$$

Output and loss layer After the convolutional layer, max-over-time pooling [8] is applied to the feature maps. The max-over-time pooling yields new subsampled features $[\hat{c_1}, \dots, \hat{c_n}]$, one for each of the n filters, where $\hat{c_i} = \max(c_i)$. Intuitively, this operation captures the most significant features. Finally, these features are input to a fully-connected layer of $|C|$ softmax or sigmoid output units, one for each class.

The original architecture in [18] is designed for the multi-class setting and uses a softmax output layer. We have extended the network to the multi-label setting working with probabilistic labels by switching out the loss function with a noise-aware loss function for multi-label classification. The loss is defined as the cross-entropy over sigmoid outputs with respect to probabilistic labels (Eq. (6)). $|C|$ is the number of classes, $p(Y^{(k)}|\Lambda^{(k)})$ is the probabilistic labels for class k, σ is the logistic sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$), $W_2 \in \mathbb{R}^{n \times |C|}$ is the weights between the max-pooled features and the output layer, b_2 is a bias term for the fully connected layer, $\vec{z} \in \mathbb{R}^n$ is the vector of max-pooled features, $\hat{y^{(k)}}$ is the logits for class k, and $\theta = \langle E, W_1, W_2, b_1, b_2 \rangle$ denotes the model parameters.

$$\begin{aligned} \vec{z} &= [\hat{c_1}, \dots, \hat{c_n}] \\ \hat{y} &= W_2^T \vec{z} + b_2 \\ L(\theta) &= \frac{1}{|C|} \\ &\quad \times \sum_{k=0}^{|C|} -\big(p\big(Y^{(k)}|\Lambda^{(k)}\big) \log\big(\sigma\big(\hat{y^{(k)}}\big)\big) \\ &\quad + \big(\big(1 - p\big(Y^{(k)}|\Lambda^{(k)}\big)\big) \\ &\quad \times \log\big(1 - \sigma\big(\hat{y^{(k)}}\big)\big)\big)\big) \end{aligned} \tag{6}$$

Model analysis There are a few key concepts that characterizes the CNN architecture for text classification. Most prevalent is the assumption that a smaller amount of tokens in the input are decisive for classification. This assumption is expressed both with the max-over-time pooling and by using ReLU activations, that have a sparsity effect on the network. Moreover, since all the neurons inside a single filter share weights, each filter can be seen as a *feature-learner*, that looks for a certain feature in the input. As weights are not shared across filters, increasing the number of filters can allow the network to learn to detect more distinct features in the input. The training procedure will cause the filters to learn different features to minimize the loss. How many filters to use depends on the task. If too many filters are used, some filters typically become so called "dead filters" that never activate and always output zeros.

4. Experimental setup

This section outlines the experimental setup that was used to produce the results presented in the following section (Section 5). The experiments include data analysis of an Instagram corpora, comparing domain-specific embeddings to pre-trained embeddings, and training a deep text classifier for clothing classification of Instagram posts.

4.1. Data

Experiments were conducted using textual data from the Instagram platform, this section describes the datasets in more detail.

4.1.1. Instagram corpora

The empirical study of Instagram text was conducted on a provided dataset, consisting of Instagram posts from a community of users in the fashion domain. The data are in the form of a corpora consisting of image captions, user comments, and usertags associated with each post. In entirety, the corpora consists of 143 accounts, 200K posts, 9M comments, and 62M tokens, out of which 2M are unique. The numbers were computed before any pre-processing, except applying the NLTK [21] TweetTokenizer and removing user-handles.

4.1.2. Training dataset

When training classifiers, a subset of the Instagram corpora, consisting of 30K Instagram posts annotated with weak labels produced by the labeling functions described in Section 3.3.3 was used.

4.1.3. Evaluation dataset

For evaluation purposes, a smaller, manually annotated, dataset of 200 Instagram posts was used. The annotation was a collective work by four participants in our research group. Noteworthy is that the truth labels are based on the image associated with the text. In that sense, the evaluation is unfavorable for the text-based analysis. Since the labels are decided by the image, certain posts can have labels that cannot be inferred from the text alone, degrading the measured performance of the developed text classification models.

4.2. Data analysis

The data analysis was conducted on the entire Instagram corpora. To measure the fraction of emojis, hashtags, and user-handles, the NLTK [21] TweetTokenizer was used to tokenize the text, and regular expressions were applied to extract the desirable tokens. To quantify the amount of OOV words, two vocabularies were used, the Google-news vocabulary [14], and GNU aspell v0.60.7. Finally, `langid.py` [22] was used to capture the distribution of languages in the corpora.

4.3. Word embeddings

To find out the best set of embeddings for the Instagram domain we trained a large set of embeddings on the Instagram corpora using a variety of hyperparameters and algorithms. The embeddings were evaluated using intrinsic evaluation that included a comparison with off-the-shelf embeddings.

4.3.1. Evaluation of word embeddings

Three datasets were used to evaluate trained word embeddings on the word similarity task, (1) WordSim353, introduced by [12], is a dataset consisting of 353 word pairs with accompanying relatedness scores; (2) SimLex-999, presented in [16], is a dataset of 999 word pairs and similarity labels; and (3) FashionSim, an open-source[10] dataset consisting of 307 fashion related words and relatedness scores, collectively annotated by our research group in cooperation with fashion experts.

4.4. Weakly supervised text classification

This section describes the setup used to train and evaluate text classifiers.

4.4.1. Evaluation

Classifiers were evaluated after training by freezing the weights of the models and comparing the models' predictions to the annotated dataset.

4.4.2. CNN models and baselines

Two classifiers were evaluated. The CNN model described in Section 3.3.5 was trained using the training dataset annotated with weak labels described in Section 4, and where labels had been combined into probabilistic labels using the data programming framework prior to training (CNN-DataProgramming). The same model was also trained using the same training dataset but where the weak labels had been combined using the majority vote rather than data programming (CNN-MajorityVote).

The CNN models were compared against a human benchmark (DomainExpert). The human benchmark represents the average performance on the classification task of three people from our research group. Human test participants were faced with the same task as the other models, namely to classify Instagram posts based solely on the text.

4.4.3. Hyperparameters

Limited hyperparameter tuning was done prior to the experiments. We used 128 filter windows of size 3, 4, and 5, and a mini-batch size of 256. Moreover we used a vector dimension in the embedding layer of 300 with randomly initialized embeddings updated as part of training. For regularization we used a dropout keep probability of 0.7 and a l_2 constraint of 0. Finally, ReLU ($f(z) = \max(0, z)$) was used as the activation

[10]https://github.com/shatha2014/FashionRec

function, and the padding strategy was set to VALID and the learning rate to 0.01. The values were chosen based on hyperparameter tuning using random-search on 10% randomly chosen examples from the training dataset.

5. Results and discussion

In this section, we present our experimental results.

5.1. *What characterizes Instagram as a source of text?*

This section presents results from exploratory data analysis of the Instagram corpora.

5.1.1. *Lexical noise measurements*

Table 2 contains statistics that capture the distinctive properties of the Instagram corpora compared with newswire text. Removing all online-specific tokens (hashtags, user-handles, emojis, URLs) results in an OOV fraction of 0.30 based on the aspell dictionary, that can be compared with 0.25 that was obtained by [3] on a Twitter corpora using the same pre-processing and dictionary.

5.1.2. *Language distribution*

Although all Instagram posts in the corpora are from English accounts, the comments sections are often multi-lingual. Applying `langid.py` [22] on the set of 9 million comments reveals that 52% of the comments are primarily written in English. The Language identified as the second most common was Chinese on 6.5%, followed by Japanese on 5%, German on 3%, and Spanish on 2%. In total, 97 languages were identified in the set of comments.

5.1.3. *Text distributions*

The number of comments associated with Instagram posts is varying. Data analysis indicate that the distribution of comments and amount of text associated with posts exhibit the long tail phenomenon. The frequencies of number of comments in Instagram posts roughly follows a *power law* relationship (Fig. 5). Some posts have no comments at all, while other posts have a few thousand comments. The mean length of captions and comments in the corpora is 29, and 6 tokens, respectively.

Table 2
Measurements of lexical noise in the corpora

Text Statistic	*Fraction of corpora size*	*Average/post*
Emojis	0.15	48.63
Hashtags	0.03	9.14
User-handles	0.06	18.62
Google-OOV words	0.46	145.02
Aspell-OOV words	0.47	147.61

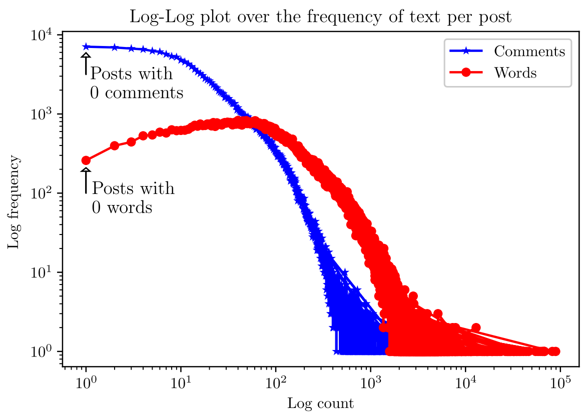


Fig. 5. The text distribution in the corpora.

5.1.4. *Discussion*

In comparison with measurements on Twitter corpora [3], text from Instagram is just as noisy based on our measurements (Table 2). Notable is also the high diversity of languages occurring in the comment sections on Instagram and the short length of comments (mean length measured to be 6 tokens).

The long-tail distribution of text on Instagram can be explained with the follower count of the post author and the preferential attachment theory [31]. As an Instagram post attracts a lot of comments, it will get a larger spread on the Instagram platform. This causes a snowball effect, where a post that already has many comments will be more likely to attract even more comments.

5.2. *Comparison of word embeddings for the Instagram domain*

In this section, word embeddings trained on the Instagram corpora are examined. The experiments include a comparison between Instagram embeddings and off-the-shelf embeddings, as well as hyperparameter tuning of embeddings trained on Instagram text.

5.2.1. *Are domain-specific embeddings better than pre-trained embeddings?*

Off-the-shelf embeddings outperform the domain-specific embeddings on general evaluation metrics such as Simlex-999 [16], and Wordsim353 [12]. However, on the FashionSim evaluation dataset the reversed relationship occurs (Fig. 6). To exemplify, the embeddings FASTTEXT-FASHION had the lowest score on the Simlex-999 evaluation dataset and the highest score on the FashionSim dataset.

5.2.2. *What are suitable hyperparameters?*

It can be observed that FastText and Word2vec are highly dependent on the hyperparameter settings, while Glove is stable in comparison (Fig. 7). FastText demonstrated the best results on the given task. With FastText, the top accuracy was achieved with Skip-gram and context window size 2. A prevalent trend in the results is that CBOW performed better with larger window sizes, as opposed to Skip-gram that achieved the highest results with smaller context windows. Additionally, a substantial boost in accuracy was observed when increasing the vector dimension from 50 to 100, and then a less significant increase when further raising the dimension up to 300. When the dimension is increased above 300 there is a diminishing return of increased accuracy relative to the increased dimension.

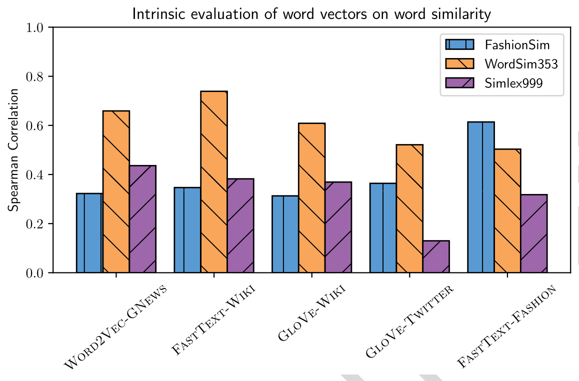


Fig. 6. Intrinsic evaluation on the word similarity task (p–value < 0.001).

5.2.3. *Discussion*

When comparing the state-of-the-art algorithms for training word embeddings, FastText embeddings yielded the most accurate semantics on the intrinsic evaluation. FastText explicitly models the morphology of words by incorporating information about subwords in the embeddings, this is useful for languages that are rich on morphology. According to our results, FastText is also suited for noisy text, as can be found in social media. This is not surprising, as social media language can be characterized as containing a large vocabulary, with many rare words, where the subword embeddings can enhance generalization between words.

5.3. *Deep text classification with weak supervision*

This section outlines the results from the experiments with deep text classification of Instagram text using weak supervision provided by the labeling functions from Section 3.3.3.

5.3.1. *The data programming paradigm versus majority voting*

Table 3 compares results from the CNN model trained with weak labels combined through majority voting with results from the same model trained with probabilistic labels obtained with data programming. The data programming approach achieves the best F_1 result, on level with the human benchmark, beating CNN-MAJORITYVOTE. The human benchmark had a higher precision but a lower recall than the CNN models.

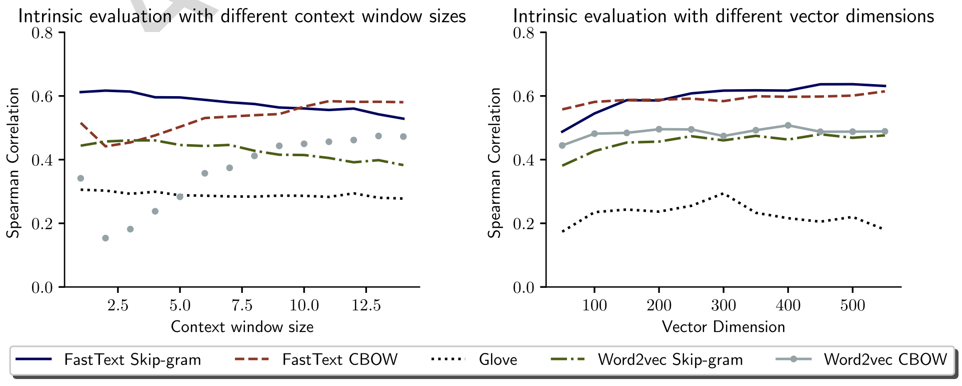


Fig. 7. Hyperparameter tuning on the FashionSim evaluation dataset (p-value $< 1.76e{-5}$). When tuning the context-window size the dimension was 300. When tuning the dimension the fine-tuned window sizes 2, 11, 12, 3, 13 for FastText skip-gram, FastText CBOW, glove, Word2vec skip-gram,Word2vec CBOW, was used.

Table 3
The average performance from three training runs

Model	*Accuracy*	*Precision*	*Recall*	*Micro-F_1*	*Macro-F_1*	*Hamming Loss*
CNN-DataProgramming	**0.797** ± 0.01	**0.566** ± 0.05	0.678 ± 0.04	**0.616** ± 0.02	**0.535** ± 0.01	**0.195** ± 0.02
CNN-MajorityVote	0.739 ± 0.02	0.470 ± 0.06	**0.686** ± 0.05	0.555 ± 0.03	0.465 ± 0.05	0.261 ± 0.03
DomainExpert	**0.807**	**0.704**	0.529	0.604	0.534	**0.184**

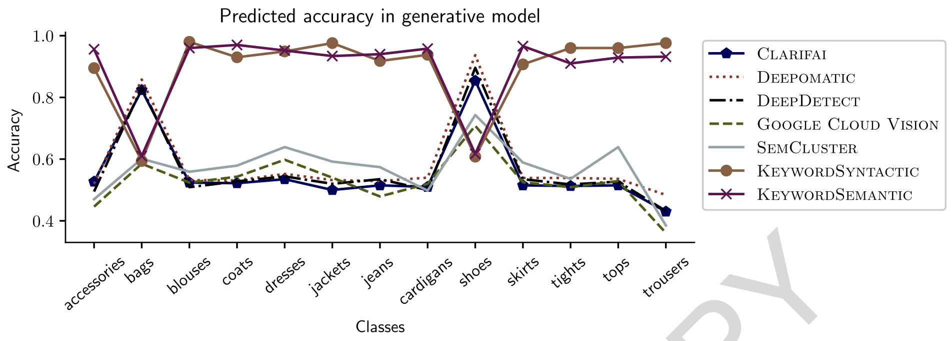


Fig. 8. Accuracy of labeling functions in generative models.

5.3.2. *Generative models of the labeling functions*

Fig. 8 visualizes the relative accuracy between labeling functions that was learned by the generative models in CNN-DataProgramming. The keyword-functions were given the highest accuracy overall, indicating that when the keywords are found in the text it tend to be telling for the image contents. This implies that the keyword functions often agrees with the majority in their votes, which in turn gives them a high estimated accuracy. In general, the relative accuracy among labeling functions differed from class to class. The spikes in the accuracy of Clarifai, Deepomatic, and DeepDetect on the classes of "bags" and "shoes" indicate that the APIs are especially consistent in their predictions on those classes.

5.3.3. *Error analysis*

A part of the error is attributable to the disparity between the labels in the test set and the text. As the ground truth is determined based on the image contents of the Instagram post, there is an inherent error when information is lacking in the text. This is also evident from the relatively low human benchmark on the task (0.60 F_1). Moreover, the performance on the dev and train set were significantly better than on the test set for all of the trained models. This is owing to the difference between the weak labels and the ground truth. Figure 9 depicts the performance on the dev and train set per training iteration of the CNN-DataProgramming model. The performance on the dev and train sets were significantly higher than on the test set.

5.3.4. *How Do the* CNN-DataProgramming *Model Make Its Predictions?*

After training the CNN-DataProgramming model, the learned weights of the model can be frozen and used for inference. Figure 10 illustrates how the trained CNN-DataProgramming model infers clothing items from the text. The heatmap in Fig. 10 was produced by running each word in the sample text through the trained model and recording the output scores (logits) for the output class "dresses". After recording the scores, they were normalized and put on a scale and visualized.

5.3.5. *Discussion*

Considering that not all clothing items can be inferred from the text and that the human benchmark on the task is 0.60, the achieved F_1 score of 0.61 is promising. A substantial improvement is to be expected when integrating the text classifier with a model analyzing the image contents.

When combining the labels by using generative models, rather than majority voting, an increase of six F_1 points was observed. This indicates that when taking the majority vote for training, potential signals to learn from is lost. The results are concordant with prior work using data programming [2,25,26]. This result indicates that when combining labels using majority voting, potential signals to learn from is lost.

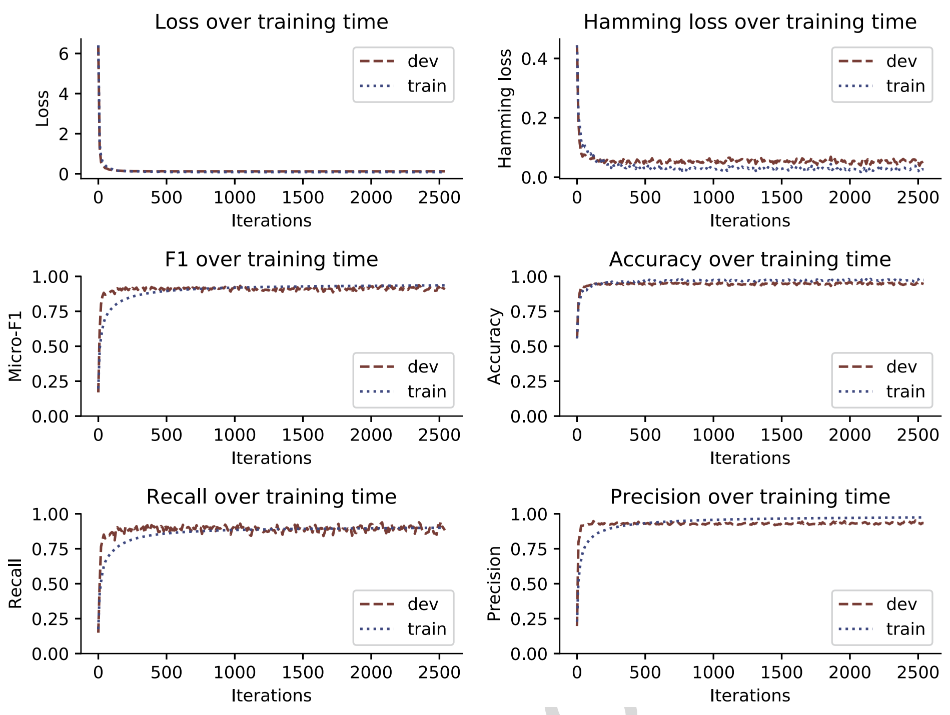


Fig. 9. Statistics on the dev and train set during training of the CNN-DataProgramming model.

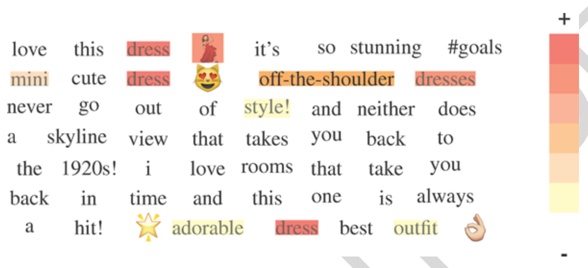

love this dress it's so stunning #goals
mini cute dress off-the-shoulder dresses
never go out of style! and neither does
a skyline view that takes you back to
the 1920s! i love rooms that take you
back in time and this one is always
a hit! adorable dress best outfit
+
-

Fig. 10. Heatmap of a sample Instagram text, where a higher heat indicates a larger logit in the trained CNN-DataProgramming model.

In [26] it is assumed that labeling functions are binary. We propose to extend the base model to the multi-label scenario by learning a separate generative model for each class. In our experiments, the relative accuracy of labeling functions differed between classes, strengthening our belief that learning separate generative models for each class is useful.

6. Conclusion and future work

In this paper we presented the first empirical study of Instagram text that we are aware of. Moreover, we evaluated domain-specific embeddings trained on Instagram text and presented a novel pipeline that utilizes weak supervision to train a deep classifier to recognize fashion clothing based on text from Instagram posts.

With weak supervision, we were able to label a large dataset in hours, something that would have taken months to do with human annotators. The weak supervision signals were combined with the data programming paradigm, which makes for a proof-of-concept of the paradigm in a new domain. Moreover, the original model for binary classification was extended to the multi-label setting by learning a separate generative model for each class.

The results demonstrate that the text on Instagram is just as noisy as have been reported in studies on Twitter text, that the text distribution has a long tail, and that the comment sections on Instagram are multi-lingual. Our experiments also indicate that there is a mismatch between text in social media and off-the-shelf embeddings trained on newswire text. We also confirmed that weak supervision is a viable approach for training deep models with unlabeled data, achieving human-level performance on the classification task. In all measures, combining weak supervision signals with the proposed combination of generative models outperformed a baseline that uses majority voting.

In future work we plan to combine the text mining methods presented in this paper with a model that analyzes the image contents associated with the text in Instagram posts.

References

[1] N. Alter, Four ways Instagram is redefining the fashion industry, 2016, Accessed: 2018-04-09.

[2] S.H. Bach, D. Rodriguez, Y. Liu, C. Luo, H. Shao, C. Xia, S. Sen, A. Ratner, B. Hancock, H. Alborzi, R. Kuchhal, C. Ré and R. Malkin, Snorkel DryBell: A case study in deploying weak supervision at industrial scale, CoRR, 2018, http://arxiv.org/abs/1812.00417, abs/1812.00417.

[3] T. Baldwin, P. Cook, M. Lui, A. MacKinlay and L. Wang, How noisy social media text, how diffrnt social media sources? in: *IJCNLP, Asian Federation of Natural Language Processing / ACL*, 2013, pp. 356–364.

[4] P.R. Berthon, L.F. Pitt, K. Plangger and D. Shapiro, Marketing meets web 2.0, social media, and creative consumers: Implications for international marketing strategy, *Business Horizons* **55**(3) (2012), 261–271, SPECIAL ISSUE: STRATEGIC MARKETING IN A CHANGING WORLD, http://www.sciencedirect.com/science/article/pii/S0007681312000080. doi:10.1016/j.bushor.2012.01.007.

[5] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, Enriching word vectors with subword information, CoRR, 2016. http://arxiv.org/abs/1607.04606.

[6] F. Bravo-Marquez, E. Frank and B. Pfahringer, Transferring sentiment knowledge between words and tweets, *Web Intelligence* **16**(4) (2018), 203–220, https://content.iospress.com/articles/web-intelligence/web389. doi:10.3233/WEB-180389.

[7] B. Chiu, G.K.O. Crichton, A. Korhonen and S. Pyysalo, How to train good word embeddings for biomedical NLP, in: *BioNLP@ACL, Association for Computational Linguistics*, 2016, pp. 166–174.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, Natural language processing (almost) from scratch, *J. Mach. Learn. Res.* **12** (2011), 2493–2537, http://dl.acm.org/citation.cfm?id=1953048.2078186.

[9] A. Conneau, H. Schwenk, L. Barrault and Y. LeCun, Very deep convolutional networks for natural language processing, CoRR, 2016.

[10] B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl and W.W. Cohen, Tweet2Vec: Character-based distributed representations for social media, CoRR, 2016.

[11] B. Eisner, T. Rocktäschel, I. Augenstein, M. Bosnjak and S. Riedel, emoji2vec: Learning emoji representations from their description, CoRR, 2016, http://arxiv.org/abs/1609.08359, abs/1609.08359.

[12] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman and E. Ruppin, Placing search in context: The concept revisited, *ACM Trans. Inf. Syst.* **20**(1) (2002), 116–131. doi:10.1145/503104.503110.

[13] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan and N.A. Smith, Part-of-speech tagging for Twitter: Annotation, features, and experiments, in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers – Volume 2, HLT '11*, Association for Computational Linguistics, Stroudsburg, PA, USA, 2011, pp. 42–47, http://dl.acm.org/citation.cfm?id=2002736.2002747. ISBN 978-1-932432-88-6.

[14] Google, word2vec, 2013. https://code.google.com/archive/p/word2vec/.

[15] K. Hammar, S. Jaradat, N. Dokoohaki and M. Matskin, Deep text mining of Instagram data without strong supervision, in: *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2018, pp. 158–165. doi:10.1109/WI.2018.00-94.

[16] F. Hill, R. Reichart and A. Korhonen, Simlex-999: Evaluating semantic models with genuine similarity estimation, *Comput. Linguist.* **41**(4) (2015), 665–695. doi:10.1162/COLI_a_00237.

[17] S. Jaradat, Deep cross-domain fashion recommendation, in: *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17*, ACM, New York, NY, USA, 2017, pp. 407–410, http://doi.acm.org/10.1145/3109859.3109861. ISBN 978-1-4503-4652-8. doi:10.1145/3109859.3109861.

[18] Y. Kim, Convolutional neural networks for sentence classification, in: *EMNLP, ACL*, 2014, pp. 1746–1751.

[19] K. Lee, D. Palsetia, R. Narayanan, M.M.A. Patwary, A. Agrawal and A. Choudhary, Twitter trending topic classification, in: *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops, ICDMW '11*, IEEE Computer Society, Washington, DC, USA, 2011, pp. 251–258. ISBN 978-0-7695-4409-0. doi:10.1109/ICDMW.2011.171.

[20] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, *Soviet Physics Doklady* **10**(8) (1966), 707–710, Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[21] E. Loper and S. Bird, NLTK: The natural language toolkit, in: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics – Volume 1, ETMTNLP '02*, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 63–70. doi:10.3115/1118108.1118117.

[22] M. Lui and T. Baldwin, Langid.Py: An off-the-shelf language identification tool, in: *Proceedings of the ACL 2012 System Demonstrations, ACL '12, Association for Computational Linguistics*, Stroudsburg, PA, USA, 2012, pp. 25–30, http://dl.acm.org/citation.cfm?id=2390470.2390475.

[23] V. Major, A. Surkis and Y. Aphinyanaphongs, Utility of general and specific word embeddings for classifying translational stages of research, CoRR, 2017, http://arxiv.org/abs/1705.06262 abs/1705.06262.

[24] J. Pennington, R. Socher and C.D. Manning, Glove: Global vectors for word representation, in: *EMNLP, ACL*, 2014, pp. 1532–1543.

[25] A. Ratner, S.H. Bach, H.R. Ehrenberg, J.A. Fries, S. Wu and C. Ré, Snorkel: Rapid training data creation with weak supervision, CoRR, 2017, http://arxiv.org/abs/1711.10160.

[26] A.J. Ratner, C.M. De Sa, S. Wu, D. Selsam and C. Ré, Data programming: Creating large training sets, quickly, in: *Advances in Neural Information Processing Systems 29*, D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon and R. Garnett, eds, Curran Associates, Inc., 2016, pp. 3567–3575.

[27] A. Ritter, C. Cherry and B. Dolan, Unsupervised modeling of Twitter conversations, in: *Human Language Technologies: The*

2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT '10, Association for Computational Linguistics, Stroudsburg, PA, USA, 2010, pp. 172–180. ISBN 1-932432-65-5.

[28] A. Ritter, S. Clark, Mausam and O. Etzioni, Named entity recognition in tweets: An experimental study, in: *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11, Association for Computational Linguistics*, Stroudsburg, PA, USA, 2011, pp. 1524–1534, http://dl.acm.org/citation.cfm?id=2145432.2145595. ISBN 978-1-937284-11-4.

[29] A. Ritter, Mausam, O. Etzioni and S. Clark, Open domain event extraction from Twitter, in: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12*, ACM, New York, NY, USA, 2012, pp. 1104–1112. ISBN 978-1-4503-1462-6. doi:10.1145/2339530.2339704.

[30] A.J. Tixier, M. Vazirgiannis and M.R. Hallowell, Word embeddings for the construction domain, CoRR, 2016, http://arxiv.org/abs/1610.09333, abs/1610.09333.

[31] R. Zsuzsanna Albert and A.-L. Barabasi, Statistical mechanics of complex networks **74** (2001).