skweak: Weak Supervision Made Easy for NLP

Pierre Lison
Norwegian Computing Center
Oslo, Norway
plison@nr.no

Jeremy Barnes
Language Technology Group
University of Oslo
jeremycb@ifi.uio.no

Aliaksandr Hubin
Department of Mathematics
University of Oslo
aliaksah@math.uio.no

Abstract

We present skweak, a versatile, Python-based software toolkit enabling NLP developers to apply *weak supervision* to a wide range of NLP tasks. Weak supervision is an emerging machine learning paradigm based on a simple idea: instead of labelling data points by hand, we use *labelling functions* derived from domain knowledge to automatically obtain annotations for a given dataset. The resulting labels are then aggregated with a generative model that estimates the accuracy (and possible confusions) of each labelling function.

The skweak toolkit makes it easy to implement a large spectrum of labelling functions (such as heuristics, gazetteers, neural models or linguistic constraints) on text data, apply them on a corpus, and aggregate their results in a fully unsupervised fashion. skweak is especially designed to facilitate the use of weak supervision for NLP tasks such as text classification and sequence labelling. We illustrate the use of skweak for NER and sentiment analysis. skweak is released under an open-source license and is available at:
https://github.com/NorskRegnesentral/skweak

1 Introduction

Despite ever-increasing volumes of text documents available online, labelled data remains a scarce resource in many practical NLP scenarios. This scarcity is especially acute when dealing with resource-poor languages and/or uncommon textual domains. This lack of labelled datasets is also common in industry-driven NLP projects that rely on domain-specific labels defined in-house and cannot make use of pre-existing resources. Large pre-trained language models and transfer learning (Peters et al., 2018, 2019; Lauscher et al., 2020) can to some extent alleviate this need for labelled data, by making it possible to reuse generic language representations instead of learning models from scratch.

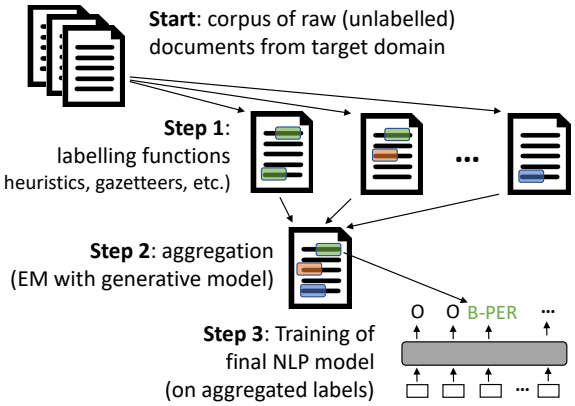

Figure 1: General overview of skweak: labelling functions are first applied on a collection of texts (step 1) and their results are then aggregated (step 2). A discriminative model is finally trained on those aggregated labels (step 3). The process is illustrated here for NER, but skweak can in principle be applied to any type of sequence labelling or classification task.

However, except for zero-shot learning approaches (Artetxe and Schwenk, 2019; Barnes and Klinger, 2019; Pires et al., 2019), they still require some amounts of labelled data from the target domain to fine-tune the neural models to the task at hand.

The skweak framework (pronounced /skwi:k/) is a new Python-based toolkit that provides solutions to this scarcity problem. skweak makes it possible to bootstrap NLP models without requiring any hand-annotated data from the target domain. Instead of labelling data by hand, skweak relies on *weak supervision* to programmatically label data points through a collection of *labelling functions* (Fries et al., 2017; Ratner et al., 2017; Lison et al., 2020; Safranchik et al., 2020a). The skweak framework allows NLP practitioners to easily construct, apply and aggregate such labelling functions for classification and sequence labelling tasks. skweak comes with a robust and scalable aggregation model that extends the HMM model of Lison et al. (2020). As

Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pages 337–346, August 1st - August 6th, 2021.

detailed in Section 4, the model now includes a feature weighting mechanism to capture the correlations that may exist between labelling functions. The general procedure is illustrated in Figure 1.

Another novel feature of skweak is the ability to create labelling functions that produce *underspecified labels*. For instance, a labelling function may predict that a token is part of a named entity (but without committing to a specific label), or that a sentence does *not* express a particular sentiment (but without committing to a specific sentiment category). This ability greatly extends the expressive power of labelling functions and makes it possible to define complex hierarchies between categories – for instance, COMPANY may be a sub-category of ORG, which may be itself a sub-category of ENT. It also enables the expression of "negative" signals that indicate that the output should *not* be a particular label. Based on our experience applying weak supervision to various NLP tasks, we expect this ability to underspecify output labels to be very useful in NLP applications.

2 Related Work

Weak supervision aims to replace hand-annotated 'ground truths' with labelling functions that are programmatically applied to data points – in our case, texts – from the target domain (Ratner et al., 2017, 2019; Lison et al., 2020; Safranchik et al., 2020b; Fu et al., 2020). Those functions may take the form of rule-based heuristics, gazetteers, annotations from crowd-workers, external databases, data-driven models trained from related domains, or linguistic constraints. A particular form of weak supervision is *distant supervision*, which relies on knowledge bases to automatically label documents with entities (Mintz et al., 2009; Ritter et al., 2013; Shang et al., 2018). Weak supervision is also related to models for aggregating crowd-sourced annotations (Kim and Ghahramani, 2012; Hovy et al., 2013; Nguyen et al., 2017).

Crucially, labelling functions do not need to provide a prediction for every data point and may "abstain" whenever certain conditions are not met. They may also rely on external data sources that are unavailable at runtime, as is the case for labels obtained by crowd-workers. After being applied to a dataset, the results of those labelling functions are aggregated into a single, probabilistic annotation layer. This aggregation is often implemented with a generative model connecting the latent (unobserved) labels to the outputs of each labelling function (Ratner et al., 2017; Lison et al., 2020; Safranchik et al., 2020a). Based on those aggregated labels, a discriminative model (often a neural architecture) is then trained for the task.

Weak supervision shifts the focus away from collecting manual annotations and concentrates the effort on developing good labelling functions for the target domain. This approach has been shown to be much more efficient than traditional annotation efforts (Ratner et al., 2017). Weak supervision allows domain experts to directly *inject* their domain knowledge in the form of various heuristics. Another benefit is the possibility to modify/extend the label set during development, which is a common situation in industrial R&D projects.

Several software frameworks for weak supervision have been released in recent years. One such framework is Snorkel (Ratner et al., 2017, 2019) which combines various supervision sources using a generative model. However, Snorkel requires data points to be independent, making it difficult to apply to sequence labelling tasks as done in skweak. Swellshark (Fries et al., 2017) is another framework optimised for biomedical NER. Swellshark, is however, limited to classifying already segmented entities, and relies on a separate, ad-hoc mechanism to generate candidate spans.

FlyingSquid (Fu et al., 2020) presents a novel approach based on triplet methods, which is shown to be fast enough to be applicable to structured prediction problems such as sequence labelling. However, compared to skweak, the aggregation model of FlyingSquid focuses on estimating the *accuracies* of each labelling function, and is therefore difficult to apply to problems where labelling sources may exhibit very different precision/recall trade-offs. A labelling function may for instance rely on a pattern that has a high precision but a low recall, while the opposite may be true for other labelling functions. Such difference is lost if accuracy is the only metric associated for each labelling function. Finally Safranchik et al. (2020b) describe a weak supervision model based on an extension of HMMs called linked hidden Markov models. Although their aggregation model is related to skweak, they provide a more limited choice of labelling functions, in particular regarding the inclusion of document-level constraints or underspecified labels.

skweak is also more distantly related to *ensemble methods* (Sagi and Rokach, 2018), as those meth-

ods also rely on multiple estimators whose results are combined at prediction time. However, a major difference lies in the fact that labelling functions only need to be aggregated once in skweak, in order to generate labelled training data for the final discriminative model (Step 3 of Figure 1). This difference is important as labelling functions may be computationally costly to run or rely on external resources that are not available at runtime, as is the case for annotations from crowd-workers.

3 Labelling functions

Labelling functions in skweak can be grouped in four main categories: heuristics, gazetteers, machine learning models, and document-level functions. Each labelling function is defined in skweak as a method that takes SpaCy Doc objects as inputs and returns text spans associated with labels. For text classification tasks, the span simply corresponds to the full document itself.

The use of SpaCy greatly facilitates downstream processing, as it allows labelling functions to operate on texts that are already tokenised and include linguistic features such as lemma, POS tags and dependency relations.[1] skweak integrates several functionalities on top of SpaCy to easily create, manipulate, label and store text documents.

Heuristics

The simplest type of labelling functions integrated in skweak are rule-based heuristics. For instance, one heuristic to detect entities of type COMPANY is to look for text spans ending with a legal company type (such as "Inc."). Similarly, a heuristic to detect named entities of the (underspecified) type ENT is to search for sequences of tokens tagged as NNPs. Section 6 provides further examples of heuristics for NER and Sentiment Analysis.

The easiest way to define heuristics in skweak is through standard Python functions that take a SpaCy Doc object as input and returns labelled spans. For instance, the following function detects entities of type MONEY by searching for numbers preceded by a currency symbol like $ or €:

```
def money_detector(doc):
  """Searches for occurrences of
  MONEY entities in text"""

  for tok in doc[1:]:
      if (tok.text[0].isdigit() and
          tok.nbor(-1).is_currency):
        yield tok.i-1, tok.i+1, "MONEY"
```

skweak also provides functionalities to easily construct heuristics based on linguistic constraints (such as POS patterns or dependency relations) or the presence of neighbouring words within a given context window.

Labelling functions may focus on specific labels and/or contexts and "abstain" from giving a prediction for other text spans. For instance, the heuristic mentioned above to detect companies from legal suffixes will only be triggered in very specific contexts, and abstain from giving a prediction otherwise. More generally, it should be stressed that labelling functions do not need to be perfect and should be expected to yield incorrect predictions from time to time. The purpose of weak supervision is precisely to combine together a set of weaker/noisier supervision signals, leading to a form of denoising (Ratner et al., 2019).

Labelling functions in skweak can be constructed from the outputs of other functions. For instance, the heuristic tagging NNP chunks with the label ENT may be refined through a second heuristic that additionally requires the tokens to be in title case – which leads to a lower recall but a higher precision compared to the initial heuristic. The creation of such derived labelling functions through the combination of constraints is a simple way to increase the number of labelling sources and therefore the robustness of the aggregation mechanism. skweak automatically takes care of dependencies between labelling functions in the backend.

Machine learning models

Labelling functions may also take the form of machine learning models. Typically, those models will be trained on data from other, related domains, thereby leading to some form of transfer learning across domains. skweak does not impose any constraint on type of model that can be employed.

The support for underspecified labels in skweak greatly facilitates the use of models across datasets, as it makes it possible to define hierarchical relations between distinct label sets – for instance, the coarse-grained LOC label from CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) may be seen as including both the GPE and LOC labels in Ontonotes (Weischedel et al., 2011).

[1] For languages not yet supported in SpaCy, the multi-language model from SpaCy can be applied.

Gazetteers

Another group of labelling functions are *gazetteers*, which are modules searching for occurrences of a list of words or phrases in the document. For instance, a gazetteer may be constructed using the geographical locations from Geonames (Wick, 2015) or names of persons, organisations and locations from DBPedia (Lehmann et al., 2015)

As gazetteers may include large numbers of entries, skweak relies on *tries* to efficiently search for all possible occurrences within a document. A trie, also called a prefix tree, stores all entries as a tree which is traversed depth-first. This implementation can scale up to very large gazetteers with more than one million entries. The search can be done in two distinct modes: a *case-sensitive* mode that requires an exact match between the entity in the trie and the occurrence and a *case-insensitive* mode that relaxes this constraint.

Document-level functions

Unlike previous weak supervision frameworks, skweak also provides functionalities to create *document-level* labelling functions that rely on the global document context to derive new supervision signals. In particular, skweak includes a labelling function that takes advantage of *label consistency* within a document. Entities occurring multiple times through a document are highly likely to belong to the same category (Krishnan and Manning, 2006). One can take advantage of this phenomenon by estimating the majority label of each entity in the document and then creating a labelling function that applies this majority label to each mention.

Furthermore, when introduced for the first time in a text, entities are often referred univocally, while subsequent mentions (once the entity is salient) frequently rely on shorter references. For instance, the first mention of a person in a text will often take the form of a full name (possibly complemented with job titles), but mentions that follow will often rely on shorter forms, such as the family name. skweak provides functionalities to easily capture such document-level relations.

4 Aggregation model

After being applied to a collection of texts, the outputs of labelling functions are aggregated using a generative model. For sequence labelling, this model is expressed as a Hidden Markov Model where the states correspond to the "true" (unobserved) labels, and the observations are the predictions of each labelling function (Lison et al., 2020). For document classification, this model reduces to Naive Bayes since there are no transitions.

This generative model is estimated using the Baum-Welch algorithm (Rabiner, 1990), which a variant of EM that uses the forward-backward algorithm to compute the statistics for the expectation step. For efficient inference, skweak combines Python with C-compiled routines from the hmmlearn package[2] employed for both parameter estimation and decoding.

4.1 Probabilistic Model

We assume a list of J labelling functions $\{\lambda_1, ..., \lambda_J\}$. Each labelling function produces a label for each data point (including a special "void" label denoting that the labelling function abstains from a concrete prediction, as well as underspecified labels). Let $\{l_1, ..., l_L\}$ be the set of labels that can be produced by labelling functions.

The aggregation model is represented as a hidden Markov model (HMM), in which the states correspond to the true underlying mutually exclusive class labels $\{l_1, ..., l_S\}$.[3] This model has multiple emissions (one per labelling function). For the time being, we assume those emissions to be mutually independent conditional on the latent state (see next section for a more refined model).

Formally, for each token $i \in \{1, ..., n\}$ and labelling function λ_j, we assume a multinomial distribution for the observed labels $\boldsymbol{Y_{ij}}$. The parameters of this multinomial are vectors $\boldsymbol{P_j^{s_i}} \in \mathcal{R}^L_{[0,1]}$. The latent states are assumed to have a Markovian dependence structure along the tokens $\{1, ..., n\}$. As depicted in Figure 2, this results in an HMM expressed as a dependent mixture of multinomials:

$$p(\lambda_j^{(i)} = \boldsymbol{Y_{ij}} | \boldsymbol{P_j^{s_i}}) = \text{Multinomial}\left(\boldsymbol{P_j^{s_i}}\right), \quad (1)$$

$$p(s_i = k | s_{i-1} = l) = \tau_{lk}. \quad (2)$$

where $\tau_{lk} \in \mathcal{R}_{[0,1]}$ are the parameters of the transition matrix controlling for a given state $s_{i-1} = l$ the probability of transition to state $s_i = k$.

The likelihood function includes a constraint that requires latent labels to be observed in at least one labelling function to have a non-zero probability.

[2] https://hmmlearn.readthedocs.io/

[3] Note that the set of observed labels $\{l_1, ..., l_L\}$ produced by the labelling functions may be larger than the set of latent labels $\{l_1, ..., l_S\}$, since those observed labels may also include underspecified labels such as ENT.

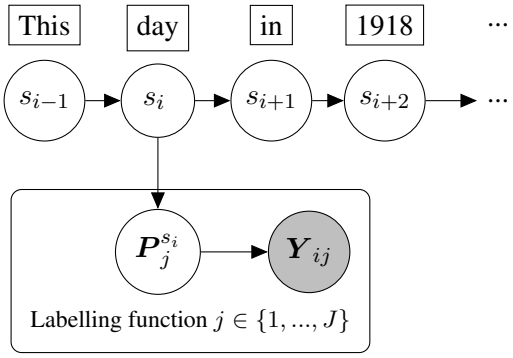

Figure 2: Aggregation model using a hidden Markov model with multiple multinomial emissions.

This constraint reduces the search space to a few labels at each step, and greatly facilitates the convergence of the forward-backward algorithm.

To initialise the model parameters, we run a majority voter that predicts the most likely latent labels based on the "votes" for each label (also including underspecified labels), each labelling function corresponding to a voter. Those predictions are employed to derive the initial transition and emission probabilities, which are then refined through several EM passes.

Performance-wise, skweak can scale up to large collections of documents. The aggregation of all named entities from the MUC-6 dataset (see Section 6.1) based on a total of 52 labelling functions only requires a few minutes of computation time, with an average speed of 1000-1500 tokens per second on a modern computing server.

4.2 Weighting

One shortcoming of the above model is that it fails to account for the fact that labelling functions may be correlated with one another, for instance when a labelling function is computed from the output of another labeling function. To capture those dependencies, we extend the model with a weighting scheme – or equivalently, a *tempering* of the densities associated with each labelling function.

Formally, for each labelling function λ_j and observed label k we determine weights $\{w_{jk}\}$ with respect to which the corresponding densities of the labelling functions are annealed. This flattens to different degrees the underlying probabilities for the components of the multinomials. The observed process has then a tempered multinomial distribution with a density of form:

$$p(\lambda_j^{(i)} = \boldsymbol{Y_{ij}} | \boldsymbol{P_j^{s_i}}, \boldsymbol{w_j}) \propto \prod_{k=1}^{L} P_{jk}^{s_i Y_{ijk} w_{jk}}. \quad (3)$$

The temperatures $\{w_{jk}\}$ are determined using a scheme inspired by delution priors widely used in Bayesian model averaging (George, 1999; George et al., 2010). The idea relies on *redundancy* as the measure of prior information on the importance of features. Formally, we define for each λ_j a neighbourhood $N(\lambda_j)$ consisting of labelling functions known to be correlated with λ_j, as is the case for labelling functions built on top of another function's outputs. The weights are then specified as:

$$w_{jk} = \exp\left(-\gamma \sum_{l \in N(\lambda_j)} R_{jlk}\right), \quad (4)$$

where γ is a hyper-parameter specifying the strength of the weighting scheme, and R_{jlk} is the recall between labelling functions λ_j and λ_l for label k. Informally, the weight w_{jk} of a labelling function λ_j producing the label k will decrease if λ_j exhibits a high recall with correlated sources, and is therefore at least partially redundant.

Also, the temperatures can be interpreted as weights of the log-likelihood function and Dimitroff et al. (2013) have shown that under some regularity conditions there exist weights that allow to maximize F_1 score when optimising the weighted log-likelihood (Field and Smith, 1994).

5 Example

With skweak, one can apply and aggregrate labelling functions with a few lines of code:

```
import spacy, re
from skweak import heuristics,
    gazetteers, aggregation, utils

# First heuristic (see Section 3)
lf1 = heuristics.FunctionAnnotator
       ("money", money_detector)

# Detection of years
lf2= heuristics.TokenConstraintAnnotator
     ("years", lambda tok: re.match
    ("(19|20)\d{2}$", tok.text), "DATE")

# Gazetteer with a few names
NAMES = [("Barack", "Obama"), ("Donald",
     "Trump"), ("Joe", "Biden")]
trie = gazetteers.Trie(NAMES)
lf3 = gazetteers.GazetteerAnnotator
      ("presidents", trie, "PERSON")
```

```
# We create a simple text
nlp = spacy.load("en_core_web_md")
doc = nlp("Donald Trump paid $750 in
    federal income taxes in 2016")

# apply the labelling functions
doc = lf3(lf2(lf1(doc)))

# aggregate them
hmm = aggregation.HMM("hmm",
       ["PERSON", "DATE", "MONEY"])
hmm.fit_and_aggregate([doc])

# and visualise the result (in Jupyter)
utils.display_entities(doc, "hmm")
```

skweak's repository provides Jupyter Notebooks with additional examples and explanations.

6 Experimental Results

We describe below two experiments demonstrating how skweak can be applied to sequence labelling and text classification. We refer the reader to Lison et al. (2020) for more results on NER.[4] It should be stressed that the results below are all obtained without using any gold labels.

6.1 Named Entity Recognition

We seek to recognise named entities from the MUC-6 corpus (Grishman and Sundheim, 1996), which contains 318 Wall Street Journal articles annotated with 7 entity types: LOCATION, ORGANIZATION, PERSON, MONEY, DATE, TIME, PERCENT.

Labelling functions

We apply the following functions to the corpus:

- Heuristics for detecting dates, times and percents based on handcrafted patterns
- Heuristics for detecting named entities based on casing, NNP part-of-speech tags or compound phrases. Those heuristics produced entities of underspecified type **ENT**
- One probabilistic parser (Braun et al., 2017) for detecting dates, times, money amounts, percents, and cardinal/ordinal values
- Heuristics for detecting person names, based on honorifics (such as Mr. or Dr.) along with a dictionary of common first names
- One heuristic for detecting company names with legal suffixes (such as Inc.)

[4] See also Fries et al. (2017) for specific results on applying weak supervision to biomedical NER.

Model	Token F_1	Entity F_1
Majority vote (all labelling functions)	0.61	0.57
HMM-aggregated labels:		
- only heuristics	0.57	0.43
- only gazetteers	0.36	0.35
- only NER models	0.60	0.56
- all but doc-level	0.80	0.71
- all labelling functions	0.81	0.72
Neural NER trained on HMM-aggregated labels	**0.82**	**0.72**

Table 1: Micro-averaged F_1 scores on MUC-6.

- Gazetteers for detecting persons, organisations and locations based on Wikipedia, Geonames (Wick, 2015) and Crunchbase
- Neural models trained on CoNLL 2003 & the Broad Twitter Corpus (Tjong Kim Sang and De Meulder, 2003; Derczynski et al., 2016)
- Document-level labelling functions based on (1) majority labels for a given entity or (2) the label of each entity's first mention.

All together (including multiple variants of the functions above, such as gazetteers in both case-sensitive and case-insensitive mode), this amounts to a total of 52 labelling functions.

Results

The token and entity-level F_1 scores are shown in Table 1. As baselines, we provide the results obtained by aggregating all labelling functions using a majority voter, along with results using the HMM on various subsets of labelling functions. The final line indicates the results using a neural NER model trained on the HMM-aggregated labels (with all labelling functions). The neural model employed in this particular experiment is a transformer architecture based on a large pretrained neural model, RoBERTa (Liu et al., 2019).

See Lison et al. (2020) for experimental details and results for other aggregation methods.

6.2 Sentiment Analysis

We consider the task of three class (positive, negative, neutral) sentiment analysis in Norwegian as a second case study. We use sentence-level annotations[5] from the NoReC$_{fine}$ dataset (Øvrelid et al.,

[5] Data: https://github.com/ltgoslo/norec_sentence

2020). These are created by aggregating the fine-grained annotations for sentiment expressions such that any sentence with a majority of positive sentiment expressions is assumed to be positive, and likewise with negative expressions. Sentences with no sentiment expressions are labelled neutral.

Labelling functions

Sentiment lexicons: NorSent (Barnes et al., 2019) is the only available lexicon in Norwegian and contains tokens with their associated polarity. We also use MT-translated English lexicons: **SoCal** (Taboada et al., 2011), the **IBM** Debater lexicon (Toledo-Ronen et al., 2018) and the NRC word emotion lexicon (**NRC emo.**) (Mohammad and Turney, 2010). Automatic translation introduces some noise but has been shown to preserve most sentiment information (Mohammad et al., 2016).

Heuristics: For sentences with two clauses connected by 'but', the second clause is typically more relevant to the sentiment, as for instance in "the food was nice, but I wouldn't go back there". We include a heuristic to reflect this pattern.

Machine learning models: We create a document-level classifier (**Doc-level**) by training a bag-of-words SVM on the NoReC dataset (Velldal et al., 2018), which contains 'dice labels' ranging from 1 (very negative) to 6 (very positive). We map predictions to positive (>4), negative (<3), and neutral (3 and 4). We also include two multilingual BERT models **mBERT-review**[6] (trained on reviews from 6 languages) and **mBERT-SST** (trained on the Stanford Sentiment Treebank). The predictions for both models are again mapped to 3 classes (positive, negative, neutral).

Results

Table 2 provides results on the NoReC sentence test split. As baseline, we include a **Majority class** which always predicts the neutral class. As upper bounds, we include a linear SVM trained on TF-IDF weighted (1-3)-grams (**Ngram SVM**), along with Norwegian BERT (NorBERT) models (Kutuzov et al., 2021) fine-tuned on the gold training data. Those two models are upper bounds as they have access to in-domain labelled data, which is not the case for the other models.

Again, we observe that the HMM-aggregated labels outperform all individual labelling functions as well as a majority voter that aggregates those functions. The best performance is achieved by a neural model (in this case NorBERT) fine-tuned on those aggregated labels.

	Source	Macro F1
baseline	Majority class	22.4
upper bounds	Ngram SVM	55.2
	NorBERT	68.5
lexicons	NorSent	45.3
	NorSent lemmas	33.7
	NRC VAD	8.2
	SoCal	46.1
	SoCal adv.	43.8
	SoCal Google	45.0
	SoCal Int.	36.5
	SoCal verb	37.2
	IBM	35.9
	NRC Emo.	41.7
heuristics	BUT	25.3
	BUT lemmas	24.0
trained models	Doc-level	33.0
	mBERT-review	44.3
	mBERT-SST	32.3
Aggregation	Majority vote	40.0
	HMM	49.1
Trained on agg.	NorBERT	**51.2**

Table 2: Macro F_1 on sentence-level NoReC data.

7 Conclusion

The skweak toolkit provides a practical solution to a problem encountered by virtually every NLP practitioner: how can I obtain labelled data for my NLP task? Using weak supervision, skweak makes it possible to create training data *programmatically* instead of labelling data by hand. The toolkit provides a Python API to apply labelling functions and aggregate their results in a few lines of code. The aggregation relies on a generative model that express the relative accuracy (and redundancies) of each labelling function.

The toolkit can be applied to both sequence labelling and text classification and comes along a range of novel functionalities such as the integration of underspecified labels and the creation of document-level labelling functions.

[6] https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment

References

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Jeremy Barnes and Roman Klinger. 2019. Embedding projection for targeted cross-lingual sentiment: Model comparisons and a real-world study. *Journal of Artificial Intelligence Research*, 66:691–742.

Jeremy Barnes, Samia Touileb, Lilja Øvrelid, and Erik Velldal. 2019. Lexicon information in neural sentiment analysis: a multi-task learning approach. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 175–186, Turku, Finland. Linköping University Electronic Press.

Daniel Braun, Adrian Hernandez Mendez, Florian Matthes, and Manfred Langen. 2017. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 174–185, Saarbrücken, Germany. Association for Computational Linguistics.

Leon Derczynski, Kalina Bontcheva, and Ian Roberts. 2016. Broad Twitter corpus: A diverse named entity recognition resource. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1169–1179, Osaka, Japan. The COLING 2016 Organizing Committee.

Georgi Dimitroff, Laura Toloşi, Borislav Popov, and Georgi Georgiev. 2013. Weighted maximum likelihood loss as a convenient shortcut to optimizing the F-measure of maximum entropy classifiers. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 207–214, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

C Field and B Smith. 1994. Robust estimation: A weighted maximum likelihood approach. *International Statistical Review/Revue Internationale de Statistique*, pages 405–424.

Jason Fries, Sen Wu, Alex Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data.

Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*.

E George. 1999. Discussion of "model averaging and model search strategies" by m. clyde. In *Bayesian Statistics 6–Proceedings of the Sixth Valencia International Meeting*.

Edward I George et al. 2010. Dilution priors: Compensating for model space redundancy. In *Borrowing Strength: Theory Powering Applications–A Festschrift for Lawrence D. Brown*, pages 158–165. Institute of Mathematical Statistics.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, page 466–471, USA. Association for Computational Linguistics.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 619–627, La Palma, Canary Islands. PMLR.

Vijay Krishnan and Christopher D. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1121–1128, Sydney, Australia. Association for Computational Linguistics.

Andrey Kutuzov, Jeremy Barnes, Erik Velldal, Lilja Øvrelid, and Stephan Oepen. 2021. Large-scale contextualised language modelling for Norwegian. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 30–40, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1518–1533, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Saif Mohammad and Peter Turney. 2010. Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34, Los Angeles, CA. Association for Computational Linguistics.

Saif M. Mohammad, Mohammad Salameh, and Svetlana Kiritchenko. 2016. How translation alters sentiment. *Journal of Artificial Intelligence Research*, 55(1):95–130.

An Thanh Nguyen, Byron Wallace, Junyi Jessy Li, Ani Nenkova, and Matthew Lease. 2017. Aggregating and predicting sequence labels from crowd annotations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 299–309, Vancouver, Canada. Association for Computational Linguistics.

Lilja Øvrelid, Petter Mæhlum, Jeremy Barnes, and Erik Velldal. 2020. A fine-grained sentiment dataset for Norwegian. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5025–5033, Marseille, France. European Language Resources Association.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Lawrence R. Rabiner. 1990. A tutorial on hidden markov models and selected applications in speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2019. Snorkel: rapid training data creation with weak supervision. *The VLDB Journal*.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, 1:367–378.

Esteban Safranchik, Shiying Luo, and Stephen Bach. 2020a. Weakly supervised sequence tagging from noisy rules. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5570–5578.

Esteban Safranchik, Shiying Luo, and Stephen Bach. 2020b. Weakly supervised sequence tagging from noisy rules. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5570–5578.

Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064, Brussels, Belgium. Association for Computational Linguistics.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Orith Toledo-Ronen, Roy Bar-Haim, Alon Halfon, Charles Jochim, Amir Menczel, Ranit Aharonov, and Noam Slonim. 2018. Learning sentiment composition from sentiment lexicons. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2230–2241, Santa Fe, New

Mexico, USA. Association for Computational Linguistics.

Erik Velldal, Lilja Øvrelid, Eivind Alexander Bergem, Cathrine Stadsnes, Samia Touileb, and Fredrik Jørgensen. 2018. NoReC: The Norwegian review corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

R. Weischedel, E. Hovy, M. Marcus, Palmer M., R. Belvin, S. Pradhan, L. Ramshaw, and N. Xue. 2011. OntoNotes: A large training corpus for enhanced processing. In *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*. Springer.

Marc Wick. 2015. Geonames Ontology.