Annual Progress Report (2022 – 2023)

Ayush Maheshwari (184050007)
Ekal Fellow & Research Scholar
Supervisor : Prof. Ganesh Ramakrishnan
Department of CSE
Indian Institute of Technology Bombay

Contents

Chapter 1

Introduction

The Ekal Vidyalaya Foundation is a non-profit organisation that initiates, supports, and runs one-teacher schools (popularly known as Ekal Vidyalayas) all over the country. The Ekal Vidyalaya movement aims to help eradicate illiteracy from rural and tribal India by 2015. To date, Ekal Vidyalaya is a movement of over 102,643 teachers, 11,000 (Approximately) voluntary workers, 35 field organizations (scattered in 22 Indian states), more than 27 lakh students and 8 support agencies as of August 2020. With this tremendous human force, the Ekal Vidyalaya movement strives to create a network of literacy centers that will educate and empower children in rural and tribal India. With the participation of numerous non-profit trusts and organizations, this program has now become the greatest education movement in the country.

Ekal Vidyalayas - 'One Teacher Schools' aims to ensure functional literacy among all children and further link them to formal schooling. These schools provide free, non-formal education to the children in the age group of 6 to 14 years, operate for 2.5 hours to 3 hours a day, for about 22 days a month throughout the year. Acharya (Teacher) is essentially from the same village who teaches 25 to 30 students. It has almost 50 percent girl-child participation. Despite several villages now having Government primary schools under Sarva Shiksha Abhiyan and Rightto Education Act(RTE),the Ekal schools continue to be the main source of basic education, for reason of other schools either not having teachers at all or if posted, the poor attendance of teachers in such remote areas.

Ekal Prashikshan app is a teaching learning app targeted for Ekal Vidyalaya teachers to capture, complement and disseminate knowledge and practices. The app has been downloaded 10,000+ times and active installs are around 2400. The app consists of several videos categorised into topics such as Science, farming, mathematics, english, samskrit,

etc. Due to the cheap and pervasive availability of the Internet, Ekal Prashikshan app is thought to be a complementary tool for teacher development and continuous learning.

1.1 Importance of *Prashikshan*

In 2018, a study was conducted to evaluate and compare the performance of Ekal students viz., students of government and private schools. Following observations were made by the study :

- Ekal students on average score more than 55% marks in English, local language and General Awareness. The only area of concern is Mathematics.
- Around 58% of Ekal students in the age 6-10yrs demonstrate learning* expected in a Class 2nd student. Around 52% of Ekal students in the age 10-14yrs demonstrate learning* expected in a Class 4th student.
- Ekal students score above 55% marks even in General Awareness which isn't covered in the formal schooling system. Ekal students are learning concepts and not only having rote learning.
- Ekal only students consistently rank at par in values of benevolence, security, tradition and conformity with those students who also go to another school.
- Females at Ekal schools do as well as males. Also, great to see more females than males in Ekal schools!
- The **performance of Ekal students is correlated to the quality of the teacher** as measured by an assessment and the literacy rate. Where the literacy rate is better and Ekal has better teachers, students show better results. This indicates that Ekal's teacher has an impact on student's learning.
- Performance of Ekal students is at par with their peers at Government schools.

As pointed out in the study, the performance of Ekal students is correlated to the quality of the teacher. Hence, teacher prashikshan is a crucial component for enhancement of students' performance. As pointed above, Ekal prashikshan mobile application is available for continuous teacher training. However, there are few issues with the application that prevents its widespread usage among the teachers. Analysis of app usage shows that there is very less interest in watching videos among the teachers.

1.2 Research Topic

Learning from less data in a weakly supervised manner

1.3 Outcomes

In addition to publications, following are the outcomes till date:

1. Open-source tool for translation of large documents - We introduce UDAAN, an open-source post-editing tool that can reduce manual editing efforts to quickly produce publishable-standard documents in several Indic languages. The tool has won the Best Demo Paper Awards at CODS-COMAD conference, 2023. The tool has been used by AICTE for translating more than 900 diploma and engineering books in Indian languages.

2. Open-source library to quickly label training data - We present Spear, an open-source python library for data programming with semi supervision. The package implements several recent data programming approaches including facility to programmatically label and build training data.

3. Free service for translation - We have build an automatic translation model that leverages domain dictionaries to translate text and producing translations adhering to the given domain. We provide translation service across 11 Indian languages including English. The translation service is hosted for public to use for free. Furthermore, the translation technology has been licensed to an external company for the purpose of converting it into a production system.

1.4 Research and Ekal

In my current PhD research, I am looking into the problem of producing domain specific translations without access to labeled training corpus. Ekal Vidyalas are present in several geographies of India. The teachers and students are conversant in their mother tongue. This necessitates to continuously produce educational content in their local languages. We need to prepare multi-lingual educational content in both physical and online platforms.

Ekal Vidyalas are spread throughout the country crossing language barriers, therefore it is essential that video, audio or text content is available in Indian languages. Translation

of content is a key component to ensure uniform maintenance of teaching quality. In my PhD research, I am investigating the automatic translation of content among Indian languages. We have developed a tool that ensure rapid and quality translation of the content[1]. UDAAN is an open-source post-editing tool that can reduce manual editing efforts to quickly produce publishable documents in different languages.

We have developed a domain-aware automatic translation system that leverages domain vocabulary to generate coherent, consistent and domain specific translation.

1.5 Featured Publications

1. **Adaptive mixing of auxiliary losses in supervised learning**

   Durga Sivasubramanian, **Ayush Maheshwari**, Prathosh AP, Pradeep Shenoy , Ganesh Ramakrishnan, AAAI, 2023 (Oral Presentation)

2. **UDAAN-Machine Learning based Post-Editing tool for Document Translation**,

   Ayush Maheshwari, Ajay Ravindran, Venkatapathy Subramanian, Ganesh Ramakrishnan, CODS-COMAD Demonstrations, 2023 (Best Demo Paper Award)

3. **A Benchmark and Dataset for Post-OCR text correction in Sanskrit.**

   Ayush Maheshwari, Singh N., Krishna A., Ganesh Ramakrishnan Findings of EMNLP, 2022

4. **SPEAR: Semi-supervised Data Programming in Python.**

   Ayush Maheshwari, Guttu Sai Abhishek, Harshad Ingole, Parth Laturia, Vineeth Dorna, Ganesh Ramakrishnan, Rishabh Iyer , EMNLP Demonstrations, 2022

5. **Learning to Robustly Aggregate Labeling Functions for Semi-supervised Data Programming**

   Ayush Maheshwari, KrishnaTeja Killamsetty, Ganesh Ramakrishnan, Rishabh Iyer , Marina Danilevsky , & Lucian Popa, Findings of ACL, 2022

[1]Udaan Tool https://www.udaanproject.org

6. **Data Programming using Semi-Supervision and Subset Selection**

   Ayush Maheshwari, Oishik Chatterjee , KrishnaTeja Killamsetty, Rishabh Iyer , & Ganesh Ramakrishnan, Findings of ACL, 2021

7. **Joint Learning of Hyperbolic Label Embeddings for Hierarchical Multi-label Classification**

   Ayush Maheshwari , Soumya Chatterjee , Ganesh Ramakrishnan & Saketha Nath Jagaralpudi , EACL, 2021

Chapter 2

Udaan - Post editing tool for Translation

2.1 Introduction

Recently, neural-based MT systems have led to tremendous improvement in the performance of machine translation [8]. However, machine translation quality not at par with the quality of human translation [15]. In practice, human translators perform post-editing (PE) on top of the output produced by NMT systems with the help of computer-aided translation (CAT) tools. Existing CAT tools incorporate MT in two ways. One way is to use translation memory (TM), a linguistic database that stores translation segments and suggests similar or identical matching words during future translations. The other way is to use a pre-translated text (MT output) to improve the final translation. The latter has been found to yield productivity gains of 36% using neural MT [40] and faster technical translations [47].

Existing CAT tools rely on TM so that translated pairs can be reused later, much like a find-and-replace function but between two languages. Additionally, TMs are integrated with additional domain-specific terminologies in a format similar to bilingual glossaries. However, the use of TMs has resulted in over-recycling of sentences (or their parts). This may not conform to the context of the given text to be translated [9]. Moreover, translators may use TM deliberately to reduce variance in the translation resulting in poor translation quality and inconsistency [27, 12]. We circumvent this problem by allowing the user to incorporate domain-specific lexicons while producing raw MT. In addition to the

Figure 2.1: Screenshot of the UDAAN tool with highlights about its key features.

interactive translation suggestions, we reduce post-editing efforts by suggesting the users (optionally) apply edits made on the current page across all pages, with the additional visual context for correct replacement decisions.

UDAAN is an end-to-end document translation pipeline with an efficient PE tool that combines both the benefits of TM and MT. We integrate both OCR and MT pipeline, thus allowing users to upload PDF or image for translation. This reduces the additional overhead of OCR conversion[1]. To mitigate the overall load for translators, we introduce the following features in our tool:

Interactive Translation: The UDAAN tool provisions for interactive translation and post-editing. By learning from the edit patterns of a user, we provision for either "global replacements" or "local suggestions for replacements" based on similar patterns. This reduces editing time and ensures consistency in the edited output.

Lexicon suggestions: Besides the machine-generated translation for the entire source document, domain-specific technical dictionaries (source-target) are presented to guide the user's edits. In addition, users can specify domain dictionaries while uploading the document for translation, thus producing a domain-specific raw translation. The tool

[1]Our tool can be used for OCR post-editing as well

Figure 2.2: UDAAN tool allows cropping, resizing and addition of images and tables, *etc.*

allows users to track modifications based on global replacements or technical dictionary suggestions. This tracking is maintained by highlighting the words using a variety of colours.

Sentence alignment: To reduce the cognitive load on the users, we highlight sentences in the source as well as the translated documents as shown in Figure 2.1. Source and target sentences are highlighted on mouse hover over the corresponding sentences in either window pane(source/target contents). Alignment is preserved by tracking unique ids in the source and target sentences. In the case of image files, the unique IDs are also mapped to their corresponding bounding boxes detected by the OCR engine.

Rich Text Formatting: Users can perform rich text editing similar to word processor software. Our tool addition of images, equations, tables *etc.*, also allowing cropping and resizing of images. The final post-edited document can be exported in various popular formats such as docx, pdf, latex, *etc.* (refer to Figure 2.2).

Offline working and sync facility: Often, translators prefer working offline using their desktops. The tool can be used in a completely offline mode on a desktop (Windows or Linux). Continuous access to the internet is not required. Additionally, our tool offers version control and synchronisation using Git for continuous backup and seamless collaboration.

Multiple roles: Our tool can be accessed with different roles such as **Correctors**- who edits a document, **Verifiers**- who validates the output from Correctors, and **Proofreaders**- who validates the final translated document.

Most of our target languages are of *Indic* language family which is typed in Devanagari

script. In addition, the tool allows users to enter text through SLP1 [2] format which is found to be more convenient by several users.

2.2 Related Work

Professional translators often use CAT (computer-aided translation) tools [41] for localisation tasks. Due to drastic improvements in NMT models, MT has improved substantially and is now actively integrated into the CAT tools [22, 14, 18, 37]. Recent studies demonstrate that post-editing MT output reduces translation errors and improves translation productivity[40].

Additionally, these tools provide features like translation memory with quality estimation and concordance functionality [14], alignment between source and MT [38], translation suggestion [22] auto-completion assistance or intelligibility assessments [42]. However, these tools require source documents in a parse-able format such as XML, JSON, doc, *etc.* and work properly for post-editing short documents. To the best of our knowledge, CAT tools are neither designed for long-document translations nor accept source documents in PDF or other difficult-to-parse formats.

2.3 System Description

2.3.1 Overview

UDAAN is an offline interactive interface for post-editing MT outputs. It provides the facility to upload a PDF and retrieve the translated document. Along with NMT, OCR is integrated into the tool pipeline. To enable active collaboration among translators and continuous backup of the work, we upload the document translation files over Github[3]. However, the project files can be uploaded over any remote version control system.

Once a project is opened in the tool, the input document and MT are displayed output side-by-side: the uploaded original document (PDF, image, *etc.*) on the right and machine-translated document on the left (see Figure 2.1). Similar to the input document, the MT output is split into multiple pages retaining the *formatting style* of the original document (*e.g.*, alignment). Non-textual part of a document such as images or equations

[2]SLP1 is an ASCII transliteration scheme for the Sanskrit language from and to the Devanagari script.
[3]Github is a popular online collaborative software development and version control platform.

will not be embedded in the MT document. Therefore, the tool provides facility to mark and insert images, equations and tables in the PE window.

2.3.2 Optical Character Recognition

Several existing PE tools assume that input documents are available in clean and parse-able formats (e.g., xml, json, docx, xls). However, users often come across documents available in PDF and image formats (*e.g.*, receipts, journal papers, government reports, scanned books, *etc.*) Our system provides the facility to upload PDF or image documents and receive OCR output for the input document. We use Tesseract 4 to perform OCR, which is an LSTM-based [17] [39] OCR engine. In a few cases where default LSTM models perform poorer, we fine-tune the models based on preliminary data collected from the tool.[4] One example is fine-tuning the model on Devanagari font in which our fine-tuned model performs better than the default version provided by Tesseract[5].

2.3.3 Machine Translation

The input provided to our MT system is an OCR document or user-provided parse-able document. Our NMT model produces translation and stores the project as a private Github repository. We demonstrate our translation capability from English to the Indic languages such as Hindi, Marathi, Tamil, Telugu, Kannada, and Malayalam. Our system can be easily extended to any other language.

We use Samanantar corpus [32] consisting of 8.6 million parallel sentences in English and Hindi. We build our NMT model based on Transformer [44] using fairseq [30]. Byte-pair encoding is used to develop the subword vocabulary. Each model contains 6 encoder layers, 6 decoder layers and 8 attention heads per layer. The input embeddings are of size 512, the output of each attention head is of size 64 and the feedforward layer in the transformer block has 2048 neurons [32].

2.3.4 Word Alignment

Existing CAT tools [2, 22] align source and target words and highlight their correspondence when the user places the mouse or text cursor on a word. In contrast to previous approaches, which provide Word documents as an input, we ingest PDF or image documents

[4] https://tesseract-ocr.github.io/tessdoc/tess4/TrainingTesseract-4.00.html
[5] https://tinyurl.com/san-tesseract

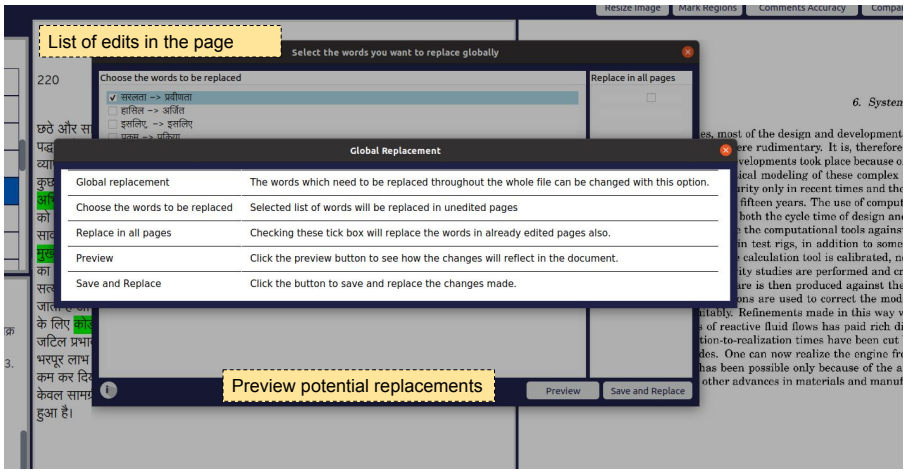

Figure 2.3: Dialog box listing all the post-edits on the current page. The user is shown list of edits and the option to replace them in (i) unedited pages and (ii) all the pages included already edited. Users can preview the potential replacements by clicking the Preview button.

as input and feed OCR output for page-wise translation. We obtain word-level alignments by fine-tuning multilingual-BERT embeddings using AwesomeAlign [28]. AwesomeAlign is an unsupervised approach to increase alignment between words having similar representation and reduce spurious alignments. It uses a combination of mask language modeling and self-training objective to encourage similarity between source and target by exploiting their agreement. We use a Samanantar dataset [32] which is an English-Hindi parallel corpus consisting of around 8 million sentences to fine-tune AwesomeAlign.

Inference AwesomeAlign deduces the final alignment between words by taking the intersection of source-to-target and target-to-source matrices, assuming a defined threshold. We use greedy algorithm followed by simAlign [36] for inference while keeping the trained model fine-tuned on AwesomeAlign objective. We observed that the lesser words are dropped, leading to better alignment. However, few words were dropped on qualitative observation due to hard threshold criteria.

2.3.5 Translation Memory and Domain specific lexicon

The tool provides several ways to include translation memory from the source-target, target-target and domain-specific lexicons.

Translation Memory (TM): Existing tools [14, 22, 2] provide the facility of TM to assist post-editing. These tools perform sentence-wise translation in the interface where

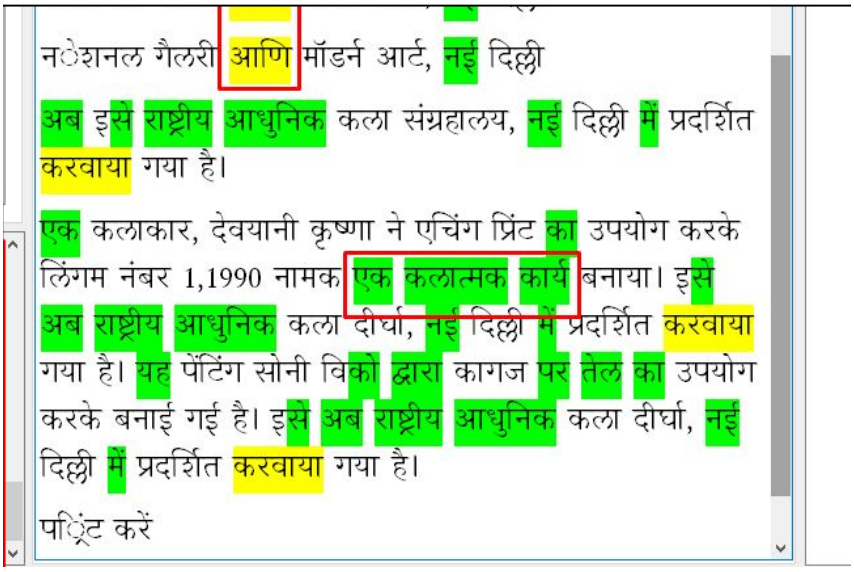

Figure 2.4: Users can track edits based on global replacements or technical dictionary suggestions implemented by highlighting the words using different colours. Yellow highlight refers to globally replaced word and green highlight refers to dictionary word replacement.

the source and target sentences are available in a parse-able format, thus making it easier to store source and target translations. In contrast, our tool performs document-level translation wherein it is difficult to keep track of the corresponding source and target sentences. Instead, we use word alignment to induce source-target translation during pre-processing.

Target-Target PE: We store target-target segment edits made by the users. There is a facility to download the segment edits. The user can in turn upload the segment edits in the tool for a different document and replace the segments in a single click.

Domain specific lexicon: In technical translation, a domain-specific lexicon helps to disambiguate translated words. Users can choose to include lexicons from domains such as Physics, Mathematics, Chemistry, Computer Science, Banking, *etc.* These lexicons are available in PDF format from the government website[6]. For each page, words from lexicons are uniquely identified using green color highlight (ref Figure 2.1).

[6]http://csttpublication.mhrd.gov.in/

2.3.6 Lexically constrained translation

NMT models are biased towards the training corpus and produce domain-oblivious translations. Lexically constrained translation allows ingestion of domain-specific terminology and other phrasal constraints during decoding [19, 3]. Current tools do not directly help incorporate lexical constraints since they perform sentence-wise translation oblivious to the translation domain. We use alignment-based constrain decoding methods that incorporate domain-specific terminology without requiring to train domain-specific NMT models. We use Transformer-based NMT models for alignment-based constrained methods [11, 19] that captures source to target alignment using weights of intermediate decoder layers. This way, we can determine whether a source constraint is currently being translated and then revise target tokens based on the corresponding target constraints. We adopt this method to produce domain-aware raw MT, thus reducing the effort required for post-editing.

2.3.7 Global replacement of edit segments

As shown in Figure 2.3, when a user makes changes in the target document, they are prompted with the list of edits made on the current page. The prompt provides an option to replace the edited words across all pages. The user can choose i. to replace words within the current page or ii. replace them in the unedited pages or iii. replace them globally across all pages. A preview option is provided to view the changes and aid the users in informed decisions on the global replacement strategy. To keep track of edits that are made using external lexicon and global replacements, they are highlighted in different colours to aid the editing experience, as shown in Figure 2.4.

2.3.8 Logging

The tool logs following unit of information during a PE session:. i. Number of edits made by a user in a page, ii. time spent on each page. iii. Word replacement (either global replacement or single replacement) along with the word that was being replaced.

2.3.9 Multi-user access

The tool is designed such that multiple post-editors can work on the same project. The tool saves the project in the user's local system and pushes the project to a remote git

repository (in our case, Github). Thus, multiple post-editors can work on different pages of the document simultaneously.

2.4 Experiments

We conducted a user study to evaluate the effectiveness of the UDAAN tool. We measured the time spent on each page during a performance evaluation session. We compared this with two other methods: Translating from scratch and post-editing the MT output in a simple word editor.

2.4.1 User Study

Participants and Study Design: We recruited 6 participants (aged 25-50 years) from our user-base working in different domains such as banks, publications houses, *etc.,* who are fluent in both English and Hindi. The participants were familiar with the tool, having used it to translate one or more documents. They were asked to translate English documents into Hindi.

Translation Interfaces The three methods are labelled: **MS-Word**, **MT-Only** and **UDAAN**. For MS-Word, given a PDF document, the user was asked to type the translated text from scratch in a word document. In MT-only, participants are provided with the MT document and its corresponding PDF and are asked to edit them in a word document. In the UDAAN setup, we ask participants to use our tool for PE. The English documents consist of 3 pages each and are from three domains: finance, mechanics, and literature. The per-page statistics for each document is given in Table 2.1.

Document	Page1	Page2	Page3
Literature	230	263	254
Finance	404	309	456
Technical	456	402	481

Table 2.1: Number of words per page for each source document.

In Table 2.2, we summarise the results of our user study. As shown, our tool speeds up the translation approximately by a factor of three. Additionally, the standard deviation measured across different participants is lower with our tool. This establishes the fact that the performance is consistent across users.

Document	MS-Word	MT-Only	UDAAN Tool
Finance	78.3 (25.7)	70.3(51.3)	**27.7(15)**
Technical	94.7(77.4)	52.3 (62.9)	**27.0(5.2)**
Literature	88.3 (52.6)	68.3(66.4)	**29.0(9.6)**

Table 2.2: Average time to complete translation (in minutes). Numbers in brackets represents standard deviation of translation time.

We asked participants to fill out an exit form to analyse the effectiveness of our tool qualitatively. We used a 5-point Likert scale [4] to measure the level of agreement or disagreement. Table 2.3 shows rating given for the questions. Overall, the user study results demonstrated the effectiveness of our tool both quantitatively and qualitatively. We found that human translators could streamline their post-editing process with the features offered in our tool.

2.5 Conclusion and Future Work

We introduce UDAAN, a post-editing tool for translation, and OCR. The pipeline of UDAAN consists of producing lexically constrained MT, which is further used for post-editing using the tool. It provides several post-editing features that significantly improve the quality of translation. Compared to translation from scratch and post-editing over MT, the user study shows that our tool achieves 3-3.5 times increase in speed. Finally, the applicability of our tool was well evaluated in the user study.

In our upcoming work, we plan to make our replacement module robust by including context-sensitive and morphologically-aware replacement techniques for suggesting replacements. Based on the candidate replacement patterns generated during post-editing stage, we build upon [1] to produce syntactically and semantically similar patterns for replacements. This will help reduce the post-editing effort. We plan to build a pipeline that improves our MT performance continuously from the user-edited triplet of source segment, MT segment, and post-edited segment. Finally, we will develop a web interface that will allow users to work in an online mode.

Questions	Rating	#Resp
system was easy to use	5	4
system was easy to use	4	2
system increased my productivity	5	4
system increased my productivity	4	2
felt very confident using the system	5	2
felt very confident using the system	4	2
felt very confident using the system	3	2
various functions in the system were well integrated	5	2
various functions in the system were well integrated	4	4
could use the system without having to learn anything new	5	2
could use the system without having to learn anything new	4	2
could use the system without having to learn anything new	2	2
there was too much inconsistency in this system	1	2
there was too much inconsistency in this system	2	4
system was slow	1	6

Table 2.3: The usability of our tool evaluated on the 5-point Likert Scale indicates the relatively high level of confidence experienced by most users. Rating 1 refers to strong disagreement whereas rating 5 refers to strong agreement.

Chapter 3

Generating relevant videos

As discussed above, the main challenge faced by the Ekal Prashikshan app is to produce high quality and consistent relevant training material for the teachers. The key problem is to define and understand what users/teachers perceive as relevant. One of the solutions is to collate data from several teachers and classify videos as relevant and irrelevant. This approach could drive significant benefits but the methodological solution is not scalable and requires too many efforts from the stakeholders (teachers here). In my current research, I focus on developing methods that require very less effort from stakeholders and generate relevant results for the end users. Technically, we call this problem as Learning from less data. Our paper titled Data Programming and Semi-supervision using subset selection is one of the efforts towards addressing this challenge. Below, we summarize our approach and findings. Modern machine learning techniques rely excessively on large amounts of labelled training data for text classification tasks such as spam detection, (movie) genre classification, sequence labelling, etc. Supervised learning approaches have utilised such large amounts of labelled data and this has resulted in huge successes in the last decade. However, acquisition of labelled data, in most cases, entails a pain-staking process requiring months of human effort. Several techniques such as active learning, distant supervision, crowd-consensus learning, and semi-supervised learning have been proposed to reduce the annotation cost. However, clean annotated labels continue to be critical for reliable results . Recently, a paradigm of data programming, in which several Labelling Functions (LF) written by humans, are used to weakly associate labels with the instances. In data programming, users encode the weak supervision in the form of labelling functions. On the other hand, traditional semi-supervised learning methods combine a small amount of labelled data with large unlabelled data.

We present a novel formulation for jointly learning the parameters over features and Labelling Functions in a semi-supervised manner. We jointly learn a parameterized graphical model and a classifier model to learn our overall objective. We study a subset selection approach to select the set of examples which can be used as the labelled set. We show, in particular, that through a principled data selection approach, we can achieve significantly higher accuracies than just randomly selecting the seed labelled set for semi-supervised learning with labelling functions. We demonstrate that by effectively combining semi-supervision, data-programming and subset selection paradigms, we significantly outperform the current state-of-the-art on seven publicly available datasets. More details about the algorithm can be found in the paper.

We have developed an algorithm to address the problem of finding relevant videos. Now, we are going to test the algorithm on the Ekal vidyalas setup to test effectiveness of our method. Refer to section 5.1 for details about the approach, methods and experiments.

3.0.1 What defines relevant videos

Consider a Youtube video that consists of textual details like owner, title, description of the video. It also contains other information such as number of views, likes, dislikes, shares and also comments about the video. Building a relevant video system involves consideration of above mentioned details in addition to end-user response. Comments on the video constitute an important feedback about the usefulness of the video. We observed that alignment of video comments and end-users feedback constitute utility of the video. Technically, alignment of two textual descriptions is an inference problem or natural language inference (NLI). The NLI problem is to determine whether a hypothesis sentence can be inferred from the premise sentence. Consider the sentences: A: An older and younger man smiling., B: Two men are smiling and laughing at the cats playing on the floor. and C: Some men are laughing. Statement A is a premise and, B and C are the hypotheses. If statement B or C can be inferred from A, then the output label is entailment. If B or C contradicts A, then the output label is a contradiction, and if nothing can be determined, then neutral. In the above example, statements A and B are neutral while A and C have an entailment relationship. NLI task relies on efficient learning of parse trees for determining entailment relationship. However, the parse tree has a high annotation cost as it requires a significant amount of expert level supervision.

3.0.2 Approach for aligning comments

Given an input sentence, our model learns the parse tree by applying composition functions using Gumbel Tree-LSTM. We learn to attend over the output of the composition function for classification that enables our model to dynamically compose an unlabelled parse tree in a bottom-up manner. Finally, sentence representations are transformed to form a feature vector that is used for the downstream classification task. In this work, we proposed an attention mechanism over Gumbel Tree-LSTM to learn hierarchical structures of natural language sentences. Our model introduces the attention over composition query vectors such that constituents are weighted and merged according to its latent part of speech information. We demonstrate the benefit of our method on three tasks: natural language inference, semantic relatedness, and sentiment analysis. We show that our approach outperforms recent state-of-the-art RvNN-based models on all three tasks. Further, we perform extensive quantitative and qualitative analysis of the induced parse trees and observe that learned induced trees are more explainable, semantically meaningful, and grammatically correct than recent RvNN models. This approach would be useful for aligning users understanding of relevant videos with the comments on the video.

3.0.3 Classify videos in the context of hierarchy

Any news article is classified into a category of topics. For eg. consider a sample headline Voice Recognition Is Improving, but Don't Stop the Elocution Lessons for which labels are Top/News/Technology. Here, article is labelled in the form of category of Top $\rightarrow$ News $\rightarrow$ Technology. Similarly, videos are also provided hashtags to gain prominence in searches. Here, we focus on the problem of classifying into a given categories arranged within a fixed hierarchy.

Automatically assigning videos into the categories arranged within a fixed hierarchy is an important co-occurring problem. In the Ekal app, videos need to be categorised, for eg., Educational $\rightarrow$ Maths $\rightarrow$ Class X, Songs $\rightarrow$ Devotional $\rightarrow$ Ram Bhajan, *etc.*. We have published a paper in *European Association for Computation Linguistics,* 2021 addressing the problem of hierarchical classification. We undertake the task of labelling documents with classes that are hierarchically organised; this problem is popularly known as *hierarchical multi-label text classification* (HMC). The main challenge in HMC is in modelling classification of the document into a large, imbalanced and structured output space. In HMC, the label taxonomy is a partially ordered set $(L, \prec)$ where L is a finite

set of all class labels.

Chapter 4

Quickly Creating Training Data using Spear

4.1 Introduction

Supervised machine learning approaches require large amounts of labeled data to train robust machine learning models. For classification tasks such as spam detection, (movie) genre categorization, sequence labelling, and so on, modern machine learning systems rely heavily on human-annotated *gold* labels. Creating labeled data can be a time-consuming and expensive procedure that necessitates a significant amount of human effort. To reduce dependence on human-annotated labels, various techniques such as semi-supervision, distant supervision, and crowdsourcing have been proposed. In order to help reduce the subjectivity and drudgery in the labeling process, several recent data programming approaches [7, 10, 6, 24] have proposed the use of *human-crafted* labelling functions or automatic LFs [25] to *weakly* associate labels with the training data. Users encode supervision in the form of labelling functions (LFs), which assign noisy labels to unlabeled data, reducing dependence on human labeled data.

While most data-programming approaches cited above provide their source code in the public domain, a unified package providing access to all data programming approaches is however missing. In this work, we describe Spear, a python package that implements several existing data programming approaches while also providing a platform for integrating and benchmarking newer ones. Inspired by frameworks such as Snorkel [7, 33] and

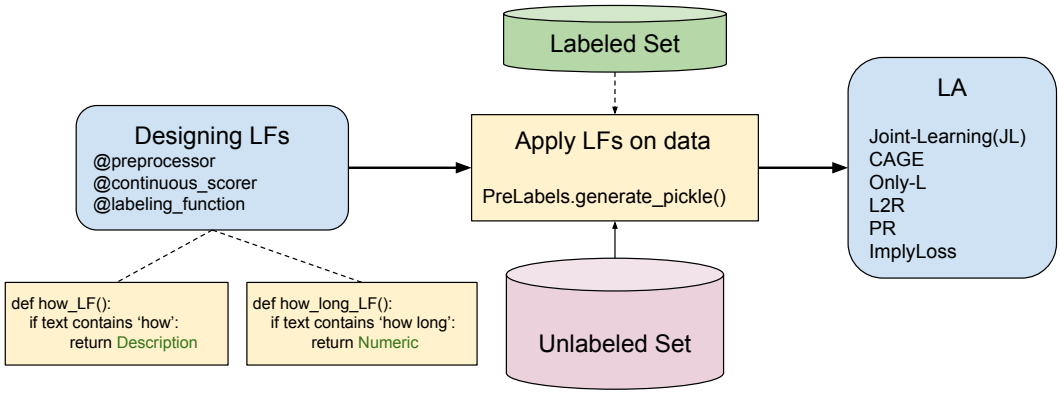

Figure 4.1: Flow of the SPEAR library.

algorithm based labeling in Matlab[1], we provide a facility for users to define LFs. Further, we develop and integrate several recent data programming models that uses these LFs. We provide many easy-to-use jupyter notebooks and video tutorials for helping new users get quickly started. The users can get started by installing the package using the below command.

```
pip install decile-spear
```

4.2 Package Flow

The SPEAR package consists of three components (and they are applied in the same order): (i) Designing LFs, (ii) applying LFs, and (iii) applying a label aggregator (LA). At the outset, the user is expected to declare an *enum* class listing all the class labels. The *enum* class associates the numeric class label with the readable class name. As part of (i), SPEAR provides the facility for manually creating LFs. LFs can be in the form of regex rules as well. Additionally, we also provide the facility of declaring a *@preprocessor* decorator to use an external library such as *spacy*[2], nltk, *etc.* which can be optionally invoked by the LFs. Thereafter, as part of (ii), the LFs can be applied on the unlabeled (and labeled) set using an *apply* function that returns a matrix of dimension #LFs $\times$ #instances. The matrix is then provided as input to the selected label aggregator (LA) in (iii), as shown in Figure4.1. We integrate several LA options into SPEAR. Each LA

[1]https://www.mathworks.com/help/vision/ug/create-automation-algorithm-for-labeling.html
[2]https://spacy.io

aggregates multiple noisy labels (obtained from the LFs) to associate a single class label with an instance. Additionally, we have also implemented in SPEAR, several joint learning approaches that employ semi-supervision and feature information. The high-level flow of the SPEAR library is presented in Figure 4.1.

Package	Designing & applying LFs	Continuous LFs	Unsup LA	Semi-sup LA	Labeled-data subset selection
Snorkel[33]	✓	✗	✗	✓	✗
Imply Loss [6]	✗	✗	✗	✓	✗
Matlab	✓	✗	✗	✗	✗
SPEAR	✓	✓	✓	✓	✓

Table 4.1: Comparison of SPEAR against available packages such as Snorkel [7], Matlab and Imply-Loss [6]. Snorkel provides support for designing and applying LFs and semi-supervised LA approaches but does not have facility for continuous LFs, unsupervised LA and labeled-data subset selection.

4.3 Designing and Applying LFs

User interacts with the library by designing labeling functions. Similar to [33], labeling functions are python functions which take a candidate as an input and either associates class label or abstains. However, continuous LFs returns a continuous score in addition to the class label. These continuous LFs are more natural to program and lead to improved recall [10].

4.3.1 Designing LFs

SPEAR uses a `@labeling_function()` decorator to define a labeling function. Each LF, when applied on an instance, can either return a class label or not return anything, *i.e.* abstain. The LF decorator has an additional argument that accepts a list of preprocessors. Each preprocessor can be either declared as a pre-defined function or can employ external libraries. The pre-processor transforms the data point before applying the labeling function.

```
@labeling_function(cont_scorer, resources, preprocessors, label)
```

```
def CLF1(x,**kwargs):
  return label if kwargs["continuous_score"] >= threshold else ABSTAIN
```

The LF can express pattern matching rules in the form of heuristics, distant supervision by using external knowledge bases and other data resources to label datapoints. LFs on SMS dataset can be seen in the example notebook here.

Continuous LFs: In the discrete LFs, users construct heuristic patterns based on dictionary lookups or thresholded distance for the classification tasks. However, the keywords in hand-crafted dictionaries might be incomplete. [10] proposed a comprehensive alternative that design continuous valued LFs that return scores derived from soft match between words in the sentence and the dictionary.

SPEAR provides the facility to declare continuous LFs, each of which returns the associated label along with a confidence score using the `@continuous_scorer` decorator. The continuous score can be accessed in the LF definition through the keyword argument `continuous_score`. As evident from Table 4.1, no other existing package provisions for continuous LFs.

```
@continuous_scorer()
def similarity(sentence,**kwargs):
  word_vecs = featurizer(sentence)
  keyword_vecs = featurizer(kwargs["keywords"])
  return similarity(word_vecs,keyword_vecs)
```

4.3.2 Applying LFs

Once LFs are defined, users can analyse labeling functions by calculating coverage, overlap, conflicts, empirical accuracy for each LF which helps to re-iterate on the process by refining new LFs. The metrics can be visualised within the SPEAR tool, either in the form of a table or graphs as shown in Figure 4.2.

PreLabels is the master class which encapsulates a set of LFs, the dataset to label and enum of class labels. PreLabels facilitates the process of applying the LFs on the dataset, and of analysing and refining the LF set. We provide functions to store labels assigned by LFs and associated meta-data such as mapping of class name to numeric class labels on the disk in the form json file(s). The pre-labeling performed using the LFs

can be consolidated into labeling performed using several consensus models described in Section 4.4.

```
sms_pre_labels = PreLabels(name="sms", data=X_V, gold_labels=Y_V,
data_feats=X_feats_V, rules=rules, labels_enum=ClassLabels, num_classes=2)
```

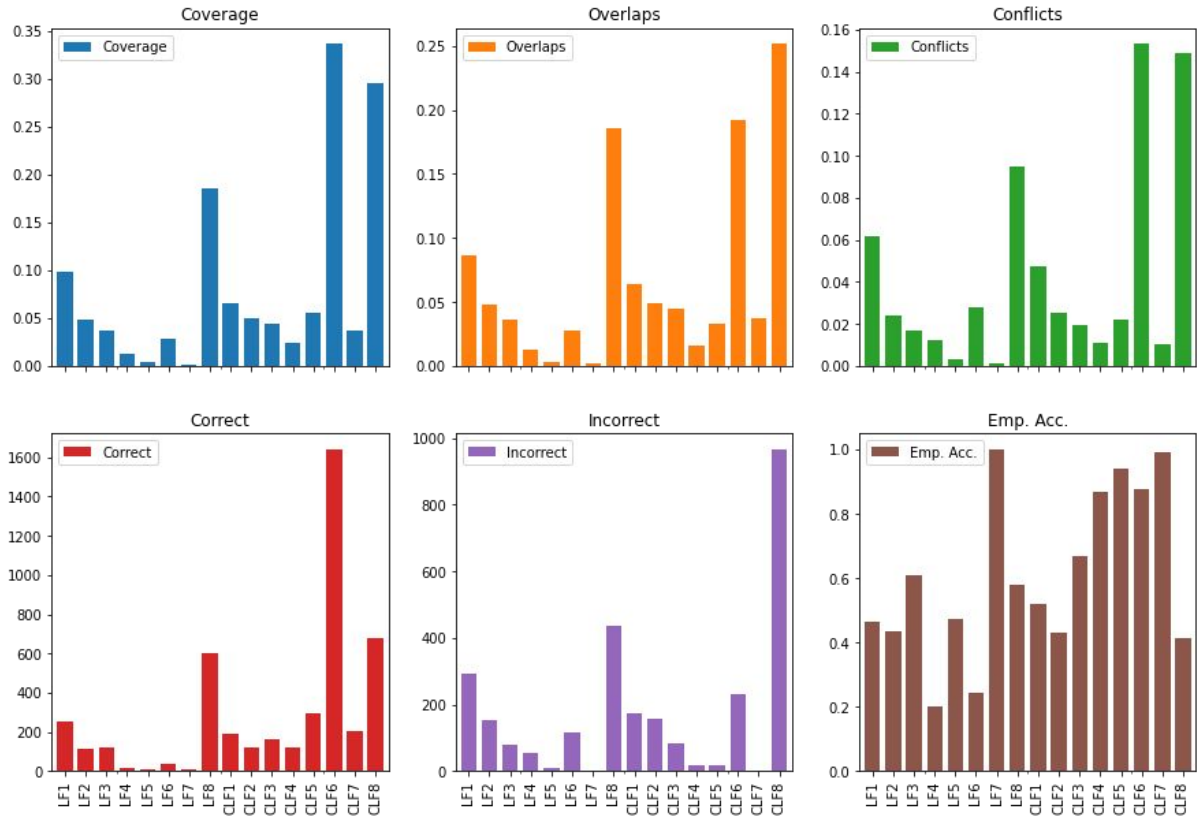

Figure 4.2: LF analysis on the SMS dataset presented in the form of graph visualization within the Spear tool. The statistics include precision, coverage, conflict and empirical accuracy for each LF.

4.4 Models

We implement several data-programming approaches in this demonstration that includes simple baselines such as fully-supervised, semi-supervised and unsupervised approaches.

4.4.1 Joint Learning [24]

The joint learning (JL) module implements a semi-supervised data programming paradigm that learns a joint model over LFs and features. JL has two key components, *viz.*, feature model (fm) and graphical model (gm) and their sum is used as a training objective. During training, the JL requires labeled ($\mathcal{L}$), validation ($\mathcal{V}$), test ($\mathcal{T}$) sets consisting of true labels and an unlabeled ($\mathcal{U}$) set whose true labels are to be inferred. The model API closely follows that of *scikit-learn* [31] to make the package easily accessible to the machine learning audience. The primary functions are: (1) `fit_and_predict_proba`, which trains using the prelabels assigned by LFs and true labels of $\mathcal{L}$ data and predicts the probabilities of labels for each instance of $\mathcal{U}$ data (2) `fit_and_predict`, similar to the previous one but which predicts labels of $\mathcal{U}$ using maximum posterior probabilities (3) `predict_(fm/gm)_proba`, predicts the probabilities, using feature model(fm)/graphical model(gm) (4) `predict_(fm/gm)`, predicts labels using fm/gm based on learned parameters. We also provide functions `save` or `load_params` to save or load the trained parameters.

As another unique feature (*c.f.* Table 4.1), our library supports a *subset-selection framework* that makes the best use of human-annotation efforts. The $\mathcal{L}$ set can be chosen using submodular functions such as facility location, max cover, *etc.* We utilise the submodlib[3] library for the subset selection algorithms. The function alternatives for subset selection are `rand_subset, unsup_subset, sup_subset_indices, sup_subset_save_files`.

4.4.2 Only-$\mathcal{L}$

In this, the classifier $P(y|\mathbf{x})$ is trained only on the labeled data. Following [24], we provide facility to use either Logistic Regression or a 2-layered neural network. Our package is flexible to allow other architectures to be plugged-in as well.

4.4.3 Cage [10]

This accepts both continuous and discrete LFs. Further, each LF has an associated quality guide component, that refers to the fraction of times the LF predicts the correct label; this stabilises training in absence of $\mathcal{V}$ set. In our package, CAGE accepts $\mathcal{U}$ and $\mathcal{T}$ sets during training. CAGE has member functions similar to (except there are no fm or gm variants

[3]https://github.com/decile-team/submodlib

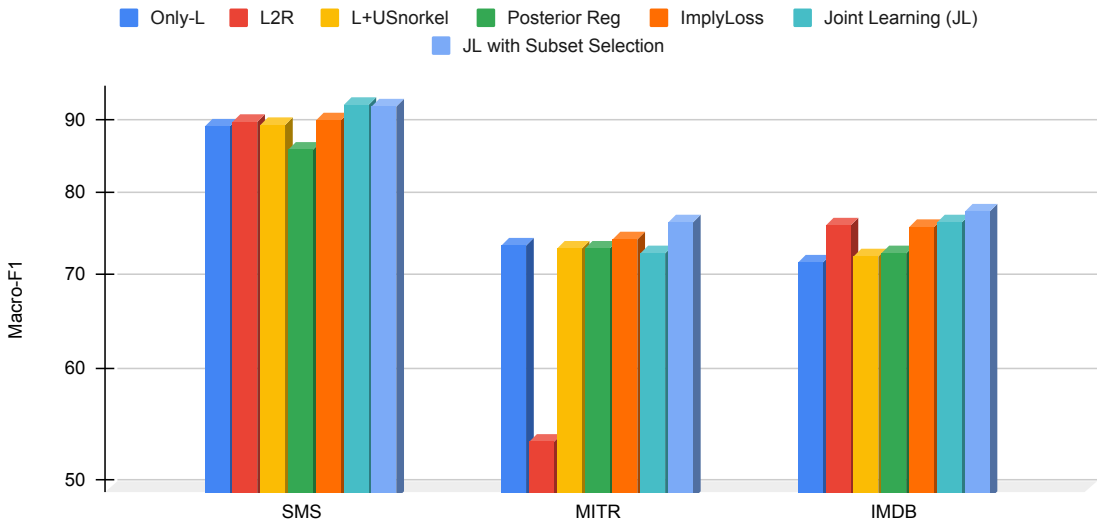

Figure 4.3: Experiments on SMS and MIT-R dataset and comparison with various approaches.

to `predict_proba`, `predict` functions in Cage) JL module, with different arguments, serving the same purpose. It should be noted that this model doesn't need labeled($\mathcal{L}$) or validation($\mathcal{V}$) data.

4.4.4 Learning to Reweight (L2R) [35]

This method is an online meta-learning approach for reweighting training examples with a mix of $\mathcal{U}$ and $\mathcal{L}$. It leverages validation set to determine and adaptively assigns importance weights to examples based on the gradient direction. This does not employ additional parameters to weigh or denoise individual rules.

4.4.5 $\mathcal{L} + \mathcal{U}_{\textbf{Snorkel}}$ [33]

This method trains a supervised classifier on $\mathcal{L}$ set and Snorkel's generative model on $\mathcal{U}$ set. Snorkel is a generative model that models class probabilities based on discrete LFs for consensus on the noisy and conflicting labels. It assigns a linear weight to each rule based on an agreement objective and label examples in $\mathcal{U}$.

4.4.6 Posterior Regularization (PR) [20]

This is a method that enables to simultaneously learn from $\mathcal{L}$ and logic rules by jointly learning a rule and feature network in a teacher-student setup. The student network learns parameter θ using the $\mathcal{L}$ set and teacher networks attempts to imitates the student network in a joint learning manner. The teacher network encodes logic rules as a regularization term in the overall loss objective.

4.4.7 Imply Loss [6]

This approach uses additional information in the form of labeled rule exemplars and trains with a denoised rule-label loss. They leverage both rules and labeled data by mapping each rule with exemplars of correct firings (i.e., instantiations) of that rule. Their joint training algorithms denoise over-generalized rules and train a classification model. It has two main components:

1. Rule Network: It learns to predict whether a given rule has overgeneralized on a given sample using latent coverage variables.

2. Classification Network: It is trained on $\mathcal{L}$ and $\mathcal{U}$ to predict the output label and maximize the accuracy on unseen test instances using a soft implication loss.

This module contains the following primary classes:

1. DataFeeder - It will essentially take all the parameters as input and create a data feeder class with all these parameters as its attributes.

2. HighLevelSupervisionNetwork (HLS) - It will take the 2 networks, the mode or the approach that needs to be used to train the model, the required parameters, the directory storing model checkpoints at different instances and the instances and labels from the labeled dataset ($\mathcal{L}$) and create an object named "hls".

HLS object will have many member functions of which the 2 significant are:
(a) **hls.train**: This function, when called with the required mode, will train the 2 network attributes of the object.
(b) **hls.test**: It supports 3 types of testing:
(i) test_w: this will test the rule network and the related model of the object.
(ii) test_f: this will test the classification network and the related model of the object.
(iii) test_all: this will test both the networks and models of the class.

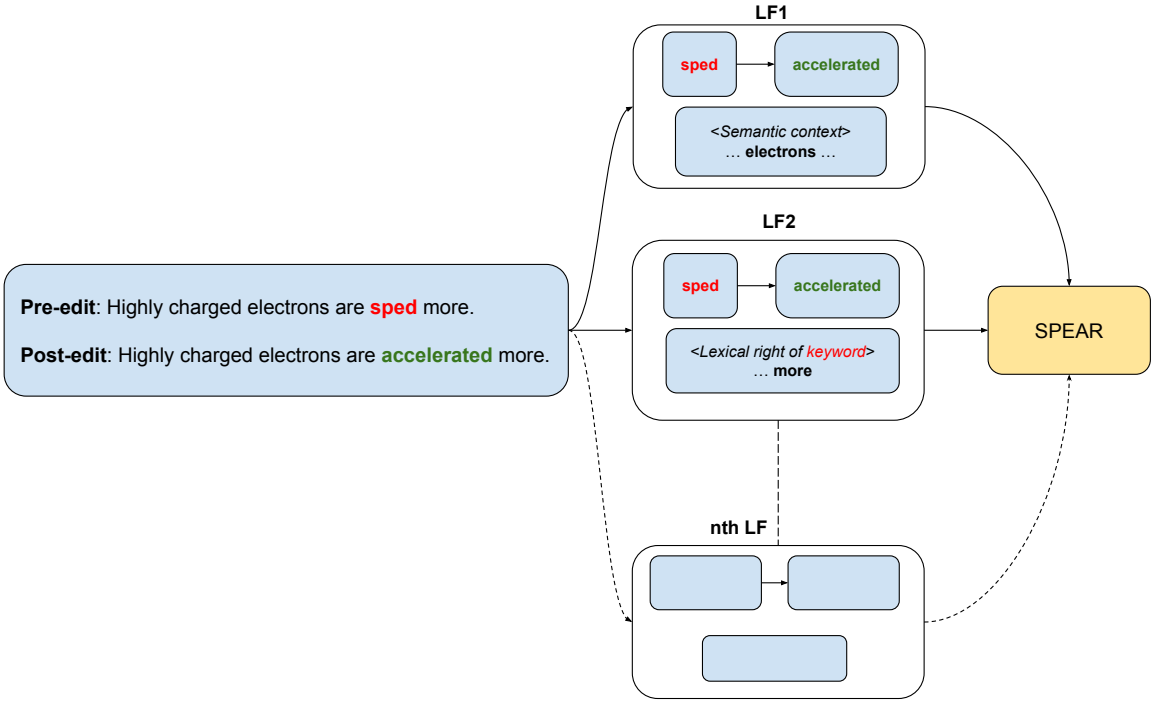

Figure 4.4: Mutiple LFs generated from post-editor edits based on semantic and lexical features while editing science (domain-specific) document in english.

4.5 Experiments

We prepared jupyter tutorial notebooks for two standard text classification datasets, namely SMS and MIT-R. We took LFs on these datasets from [6] and train using approaches implemented in this paper. Figure 4.3 shows performance of various approaches implemented using our package on two datasets.

4.6 Use Cases

SPEAR is employed in project UDAAN[4] for reducing post editing efforts. UDAAN is a post-editing workbench to translate content in native languages. Based on the post editor's patterns of changes to the target language document, candidate labeling functions are generated (based on a combination of heuristics and linguistic patterns) by the UDAAN workbench (*c.f.* Figure 4.4 for examples of LFs). Based on these LFs, SPEAR gets invoked on a combination of the edited (*i.e.*, labeled) data and the not yet edited (*i.e.*, unlabeled)

[4] https://www.udaanproject.org/

data to present consolidated edits to the post-editor. This use case has been presented in the flow chart in Figure 4.4 wherein, we present the appropriate incorporation of SPEAR into the post-editing environment of an ecosystem such as for translation (UDAAN) or even for Optical Character Recognition[5] or Automatic Speech Recognition (ASR).

As a part of third wave preparedness, SPEAR was used by the Municipal Corporation of Greater Mumbai (MCGM)s Health Ward[6] for predicting the COVID-19 status of patients to help in preliminary diagnosis.

4.7 Conclusion

SPEAR is a unified package for semi-supervised data programming that quickly annotates training data and train machine learning models. It eases the use of developing LFs and label aggregation approaches. This allows for better reproducibility, benchmarking and easier ML development in low-resource settings such as textual post-editing.

[5] https://www.cse.iitb.ac.in/~ocr/
[6] https://colab.research.google.com/drive/1tNUObqSDypUos7YNvnqvemALlkrrsB0z

Chapter 5

Semi-Supervised Data Programming

5.1 Semi-Supervised Data Programming with Subset Selection

5.1.1 Motivating Example

We illustrate the LFs on one of the seven tasks on which we experiment with, *viz.*, identifying spam/no-spam comments in the YouTube reviews. For some applications, writing LFs is often as simple as using keyword lookups or a regex expression. In this specific case, the users construct heuristic patterns as LFs for classifying spam/not-spam comments. Each LF takes a comment as an input and provides a binary label as the output; +1 indicates that the comment is spam, -1 indicates that the comment is not spam, and 0 indicates that the LF is unable to assert anything for the comment (referred to as an *abstain*). Table 5.1 presents a few example LFs for spam and non-spam classification.

In isolation, a particular LF may neither be always correct nor complete. Furthermore, the LFs may also produce conflicting labels.

In the past, generative models such as Snorkel [34] and CAGE [10] have been proposed for consensus on the noisy and conflicting labels assigned by the discrete LFs to determine the probability of the correct labels. Labels thus obtained could be used for training any supervised model/classifier and evaluated on a test set. We will next highlight a challenge in doing data programming using only LFs that we attempt to address. For each of the following sentences $S_1 \ldots S_6$ that can constitute *an observed set of training instances*, we state the value of the true label (± 1). While the candidates in S_1 and S_4 are instances of

Id	Description
LF1	If `http` or `https` in comment text, then return +1 otherwise ABSTAIN (return 0)
LF2	If length of comment is less than 5 words, then return -1 otherwise ABSTAIN (return 0).(Non spam comments are often short)
LF3	If comment contains `my channel` or `my video`, then return +1 otherwise ABSTAIN (return 0).

Table 5.1: Three LFs based on keyword lookups or regex expression for the *YouTube* spam classification task

Training data		LF outputs		Features	
id	Label	LF1(+1)	LF2(-1)	F1	F2
S_1	+1	1	0	1	0
S_2	-1	0	1	0	1
S_3	-1	0	0	0	1
S_4	+1	1	1	1	0
Test data					
S_5	-1	0	1	0	1
S_6	+1	0	0	1	0

Table 5.2: Example illustrating the insufficiency of using data programming using only LFs.

a spam comment, the ones in S_2 and S_3 are not.

1. $\langle \boldsymbol{S_1}, +1 \rangle$: Please help me go to college guys! Thanks from the bottom of my heart. https://www.indiegogo.com/projects/
2. $\langle \boldsymbol{S_2}, -1 \rangle$: I love this song
3. $\langle \boldsymbol{S_3}, -1 \rangle$: This song is very good... but the video makes no sense...
4. $\langle \boldsymbol{S_4}, +1 \rangle$: https://www.facebook.com/teeLaLaLa Further, let us say we have a completely *unseen set of test instances*, S_5 and S_6, whose labels we would also like to predict effectively:
5. $\langle \boldsymbol{S_5}, -1 \rangle$: This song is prehistoric
6. $\langle \boldsymbol{S_6}, +1 \rangle$: Watch Maroon 5's latest ... www.youtube.com/watch?v=TQ046FuAu00

In Table 5.2, we present the outputs of the LFs as well as some n-gram features F1 ('.com') and F2 ('This song') on the observed training examples S_1, S_2, S_3 and S_4 as well as on the unseen test examples S_5 and S_6. For S_1, the correct consensus can easily be performed to output the true label +1, since LF1 (designed for class +1) gets triggered, whereas LF2 (designed for class -1) is not triggered. Similarly, for S_2, LF2 gets triggered whereas LF1 is not, making it possible to easily perform the correct consensus. Hence, we have treated S_1 and S_2 as unlabelled, indicating that we could learn a model based on

LFs alone without supervision if all we observed were these two examples and the outputs of LF1 and LF2. However, the correct consensus on S_3 and S_4 is challenging since LF1 and LF2 either fire or both do not. While the (n-gram based) features F1 and F2 appear to be informative and could potentially complement LF1 and LF2, we can easily see that correlating feature values with LF outputs is tricky in a completely unsupervised setup. To address this issue, we ask the following questions:

(A) What if we are provided access to the true labels of a small subset of instances - in this case, only S_3 and S_4? Could the (i) correlation of features values (*eg.* F1 and F2) with labels (*eg.* +1 and -1 respectively), modelled via a small set of labelled instances (*eg.* S_3 and S_4), in conjunction with (ii) the correlation of feature values (*eg.* F1 and F2) with LFs (*eg.* LF1 and LF2) modelled via a potentially larger set of unlabelled instances (*eg.* S_1, S_2), help improved prediction of labels for hitherto unseen test instances S_5 and S_6?

(B) Can we precisely determine the subset of the unlabelled data that, when labelling would help us train a model (in conjunction with the labelling functions) that is most effective on the test set? In other words, instead of randomly choosing the labelled dataset for doing semi-supervised learning (part A), can we intelligently select the labelled subset? In the above example, choosing the labelled set as S_3, S_4 would be much more useful than choosing the labelled set as S_1, S_2. As a solution to (A), in Section 5.1.6, we present a new formulation, SPEAR, in which the parameters over features and LFs are jointly trained in a semi-supervised manner. SPEAR expands as **S**emi-su**P**ervis**E**d d**A**ta p**R**ogramming. As for (B), we present a subset selection recipe, SPEAR-SS (in Section 5.1.7), that recommends the sub-set of the data (*e.g.* S_3 and S_4), which, after labelling, would most benefit the joint learning framework.

5.1.2 Our Contributions

We summarise our main contributions as follows: To address (A), we present SPEAR (*c.f.*, Section 5.1.6), which is a novel paradigm for jointly learning the parameters over features and labelling functions in a semi-supervised manner. We jointly learn a parameterized graphical model and a classifier model to learn our overall objective. To address (B), we present SPEAR-SS (*c.f.*, Section 5.1.7), which is a subset selection approach to *select* the set of examples which can be used as the labelled set by SPEAR. We show, in particular, that through a principled data selection approach, we can achieve significantly

higher accuracies than just randomly selecting the seed labelled set for semi-supervised learning with labelling functions. Moreover, we also show that the automatically selected subset performs comparably or even better than the hand-picked subset by humans as in [6], further emphasising the benefit of subset selection for semi-supervised data programming. Our framework is agnostic to the underlying network architecture and can be applied using different underlying techniques without a change in the meta-approach. Finally, we evaluate our model on seven publicly available datasets from domains such as spam detection, record classification, and genre prediction and demonstrate significant improvement over state-of-the-art techniques. We also draw insights from experiments in synthetic settings

5.1.3 Methodology

5.1.4 Problem Description

Let $\mathcal{X}$ and $\mathcal{Y} \in \{1...K\}$ be the feature space and label space, respectively. We also have access to m labelling functions (LF) λ_1 to λ_m. As mentioned in Section 5.1.1, each LF λ_j is designed to record some class; let us denote[1] by $k_j \in \{1...K\}$, the class associated with λ_j. The dataset consists of 2 components, *viz.*,
1. $\mathcal{L} = \{(\mathbf{x}_1, y_1, \mathbf{l}_1), (\mathbf{x}_2, y_2, \mathbf{l}_2), \ldots, (\mathbf{x}_N, y_N, \mathbf{l}_N)\}$, which denotes the labelled dataset and
2. $\mathcal{U} = \{(\mathbf{x}_{N+1}, \mathbf{l}_{N+1}), (\mathbf{x}_{N+2}, \mathbf{l}_{N+2}), \ldots, (\mathbf{x}_M, \mathbf{l}_M)\}$, which denotes the unlabelled dataset wherein $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$.
Here, the vector $\mathbf{l}_i = (l_{i1}, l_{i2}, \ldots, l_{im})$ denotes the firings of all the LFs on instance $\mathbf{x}_i$. Each l_{ij} can be either 1 or 0. $l_{ij} = 1$ indicates that the LF λ_j has fired on the instance i and 0 indicates it has not. All the labelling functions are discrete; hence, no continuous scores are associated with them.

5.1.5 Classification and Labelling Function Models

SPEAR has a feature-based classification model $f_\phi(\mathbf{x})$ which takes the features as input and predicts the class label. Examples of $f_\phi(\mathbf{x})$ we consider in this paper are logistic regression and neural network models. The output of this model is $P^f_\phi(y|\mathbf{x})$, *i.e.*, the probability of the classes given the input features. This model can be a simple classification model such

[1]We use the association of LF λ_j with some class k_j only in the quality guide component (QG) of the loss in eqn. 5.3

as a logistic regression model or a simple neural network model.

We also use an LF-based graphical model $P_\theta(\mathbf{l}_i, y)$ which, as specified in equation (5.1) for an example $\mathbf{x}_i$, is a generative model on the LF outputs and class label y.

$$P_\theta(\mathbf{l}_i, y) = \frac{1}{Z_\theta} \prod_{j=1}^{m} \psi_\theta(l_{ij}, y) \tag{5.1}$$

$$\psi_\theta(l_{ij}, y) = \begin{cases} \exp(\theta_{jy}) & \text{if } l_{ij} \neq 0 \\ 1 & \text{otherwise.} \end{cases} \tag{5.2}$$

There are K parameters $\theta_{j1}, \theta_{j2} ... \theta_{jK}$ for each LF λ_j, where K is the number of classes. The model makes the simple assumption that each LF λ_j independently acts on an instance $\mathbf{x}_i$ to produce outputs $l_{i1}, l_{1i} ... l_{im}$. The potentials ψ_θ invoked in equation (5.1) are defined in equation (5.2). Z_θ is the normalization factor. We propose a joint learning algorithm with semi-supervision to employ both features and LF predictions in an end-to-end manner.

5.1.6 Joint Learning in Spear

We first specify the objective of Spear and thereafter explain each of its components in greater detail:

$$\begin{aligned} \min_{\theta,\phi} & \sum_{i \in \mathcal{L}} L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), y_i\right) + \sum_{i \in \mathcal{U}} H\left(P_\phi^f(y|\mathbf{x}_i)\right) \\ & + \sum_{i \in \mathcal{U}} L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), g(\mathbf{l}_i)\right) + LL_s(\theta|\mathcal{L}) \\ & + LL_u(\theta|\mathcal{U}) + \sum_{i \in \mathcal{U} \cup \mathcal{L}} KL\left(P_\phi^f(y|\mathbf{x}_i), P_\theta(y|\mathbf{l}_i)\right) \\ & + R(\theta|\{q_j\}) \end{aligned} \tag{5.3}$$

Before we proceed further, we refer the reader to Table 5.3 in which we summarise the notation built so far as well as the notation that we will soon be introducing.

First Component (L1): The first component (L1) of the loss $L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), y_i\right) = -\log\left(P_\phi^f(y = y_i|\mathbf{x}_i)\right)$ is the standard cross-entropy loss on the labelled dataset $\mathcal{L}$ for the model P_ϕ^f.

Second Component (L2): The second component L2 is the semi-supervised loss on the unlabelled data $\mathcal{U}$. In our framework, we can use any unsupervised loss function. However, for this paper, we use the Entropy minimisation [16] approach. Thus, our second

Notation	Description
f_ϕ	The feature-based Model
P_ϕ^f	The label probabilities as per the feature-based model f_ϕ
P_θ	The label probabilities as per the LF-based Graphical Model
L_{CE}	Cross Entropy Loss: $L_{CE}\left(P_\phi^f(y\|\mathbf{x}), \tilde{y}\right) = -\log\left(P_\phi^f(y=\tilde{y}\|\mathbf{x})\right)$
H	Entropy function : $H(P_\phi^f(y\|\mathbf{x})) = -\sum_{\hat{y}} P_\phi^f(y=\hat{y}\|\mathbf{x}) \log P_\phi^f(y=\hat{y}\|\mathbf{x})$
g	Label Prediction from the LF-based graphical model
LL_s	Supervised negative log likelihood
LL_u	Unsupervised negative log likelihood summed over labels
KL	KL Divergence between two probability models
R	Quality Guide based loss

Table 5.3: Summary of notation used.

component $H\left(P_\phi^f(y|\mathbf{x}_i)\right)$ is the entropy of the predictions on the unlabelled dataset. It acts as a form of semi-supervision by trying to increase the confidence of the predictions made by the model on the unlabelled dataset.

Third Component (L3): The third component $L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), g(\mathbf{l}_i)\right)$ is the cross-entropy of the classification model using the hypothesised labels from CAGE [10] on $\mathcal{U}$. Given that $\mathbf{l}_i$ is the output vector of all labelling functions for any $\mathbf{x}_i \in \mathcal{U}$, we specify the predicted label for $\mathbf{x}_i$ using the LF-based graphical model $P_\theta(\mathbf{l}_i, y)$ from eqn. (5.1) as:

$$g(\mathbf{l}_i) = \text{argmax} P_\theta(\mathbf{l}_i, y)$$

. **Fourth Component (L4):** The fourth component $LL_s(\theta|\mathcal{L})$ is the (supervised) negative log likelihood loss on the labelled dataset $\mathcal{L}$ as per eqn. (5.3): $LL_s(\theta|\mathcal{L}) = -\sum_{i=1}^{N} \log P_\theta(\mathbf{l}_i, y_i)$

Fifth Component (L5): The fifth component $LL_u(\theta|\mathcal{U})$ is the negative log likelihood loss for the unlabelled dataset $\mathcal{U}$ as per eqn. (5.3). Since the true label information is not available, the probabilities need to be summed over y: $LL_u(\theta|\mathcal{U}) = -\sum_{i=N+1}^{M} \log \sum_{y\in\mathcal{Y}} P_\theta(\mathbf{l}_i, y)$

Sixth Component (L6): The sixth component $KL(P_\phi^f(y|\mathbf{x}_i), P_\theta(y|\mathbf{l}_i))$ is the Kullback-Leibler (KL) divergence between the predictions of both the models, *viz.*, feature-based model f_ϕ and the LF-based graphical model P_θ summed over every example $\mathbf{x}_i \in \mathcal{U} \cup \mathcal{L}$. Through this term, we try and make the models agree in their predictions over the union of the labelled and unlabelled datasets.

Quality Guides (QG): As the last component in our objective, we use quality guides $R(\theta|\{q_j\})$ on LFs, which have been shown in [10] to stabilise the unsupervised likelihood training while using labelling functions. Let q_j be the fraction of cases where λ_j correctly triggered, and let q_j^t be the user's belief on the fraction of examples $\mathbf{x}_i$ where y_i and l_{ij} agree. If the user's beliefs were not available, we consider the precision of the LFs on the validation set as the user's beliefs. Except for the SMS dataset, we take the precision of the LFs on the validation set as the quality guides. If $P_\theta(y_i = k_j|l_{ij} = 1)$ is the model-based precision over the LFs, the quality guide based loss can be expressed as $R(\theta|\{q_j^t\}) = \sum_j q_j^t \log P_\theta(y_i = k_j|l_{ij} = 1) + (1-q_j^t)\log(1-P_\theta(y_i = k_j|l_{ij} = 1))$. Throughout the paper, we consider QG always in conjunction with Loss **L5**.

In summary, the first three components (L1, L2 and L3) invoke losses on the supervised model f_ϕ. While L1 compares the output f_ϕ against the ground truth in the labelled set $\mathcal{L}$, L2 and L3 operate on the unlabelled data $\mathcal{U}$ by minimizing the entropy of f_ϕ (L2) and by calibrating the f_ϕ output against the noisy predictions $g(\mathbf{l}_i)$ of the graphical model $P_\theta(\mathbf{l}_i, y)$ for each $\mathbf{x}_i \in \mathcal{U}$ (L3). The next two components L4 and L5 focus on maximizing the likelihood of the parameters θ of $P_\theta(\mathbf{l}_i, y)$ over labelled $\mathbf{x}_i \in \mathcal{L}$ and unlabelled $\mathbf{x}_i \in \mathcal{U}$ datasets respectively. Finally, in L6, we compare the probabilistic outputs from the supervised model f_ϕ against those from the graphical model $P_\theta(\mathbf{l}, y)$ through a KL divergence based loss. We use the ADAM (stochastic gradient descent) optimizer to train the non-convex loss objective.

Erstwhile data programming approaches [7, 10] adopt a cascaded approach in which they first optimise a variant of L5 to learn the θ parameters associated with the LFs and thereafter use the noisily generated labels using $g(\mathbf{l})$ to learn the supervised model f_ϕ using a variant of L3. In contrast, our approach learns the LF's θ parameters and the model's ϕ parameters jointly in the context of the unlabelled data $\mathcal{U}$.

We present synthetic experiments to illustrate the effect of Spear for data programming and semi-supervision in a controlled setting in which (i) the overlap between classes in the data is controlled and (ii) the labelling functions are accurate. The details of the synthetic experiments are provided in the appendix.

5.1.7 Spear-SS: Subset Selection with Spear

Suppose we are given an unlabelled data set $\mathcal{U}$ and a limited budget for data labelling because of the costs involved in it. It is essential for us to choose the data points that need

to be labelled properly. We explore two strategies for selecting a subset of data points from the unlabelled set. We then obtain the labels for this subset, and run SPEAR on the combination of this labelled and unlabelled set. The two approaches given below try and maximise the diversity of the selected subset in the feature space. We complement both the approaches with Entropy Filtering (also described below).

Unsupervised Facility Location: In this approach, given an unlabelled data-set $\mathcal{U}$, we want to select a subset $\mathcal{S}$ such that the selected subset has maximum diversity with respect to the features. Inherently, we are trying to maximise the information gained by a machine learning model when trained on the subset selected. The objective function for unsupervised facility location is $f_{\text{unsup}}(\mathcal{S}) = \sum_{i \in \mathcal{U}} \max_{j \in \mathcal{S}} \sigma_{ij}$ where σ_{ij} denotes the similarity score (in the feature space $\mathcal{X}$) between data instance $\mathbf{x}_i$ in unlabelled set $\mathcal{U}$ and data instance $\mathbf{x}_j$ in selected subset data $\mathcal{S}$. We employ a lazy greedy strategy to select the subset. In conjunction with Entropy Filtering described below, we call this technique *Unsupervised Subset Selection*.

Supervised Facility Location: The objective function for Supervised Facility Location [45] is $f_{\text{sup}}(\mathcal{S}) = \sum_{y \in \mathcal{Y}} \sum_{i \in \mathcal{U}_y} \max_{j \in \mathcal{S} \cap \mathcal{U}_y} \sigma_{ij}$. Here we assume that $\mathcal{U}_y \subseteq \mathcal{U}$ is the subset of data points with hypothesised label y. Simply put, $\mathcal{U}_y$ forms a partition of $\mathcal{U}$ based on the hypothesized labels obtained by performing unsupervised learning with labelling functions. In conjunction with Entropy Filtering, we call this technique *Supervised Subset Selection*.

Entropy Filtering: We also do a filtering based on entropy. In particular, we sort the examples based on maximum entropy and select fB number of data points[2], where B is the data selection budget (which was set to the size of the labelled set $|\mathcal{L}|$ in all our experiments). On the filtered dataset, we perform the subset selection, using either the supervised or unsupervised facility location as described above. Below, we describe the optimisation algorithm for subset selection.

Optimisation Algorithms and Submodularity: Both $f_{\text{unsup}}(\mathcal{S})$ and $f_{\text{sup}}(\mathcal{S})$ are submodular functions. We select a subset $\mathcal{S}$ of the filtered unlabelled data, by maximising these functions under a cardinality budget k (*i.e.*, a labelling budget). For cardinality constrained maximisation, a simple greedy algorithm provides a near-optimal solution [29]. Starting with $\mathcal{S}^0 = \emptyset$, we sequentially update

$$\mathcal{S}^{t+1} = \mathcal{S}^t \cup \underset{j \in \mathcal{U} \setminus \mathcal{S}^t}{\operatorname{argmax}} f(j|\mathcal{S}^t) \tag{5.4}$$

[2]In our experiments, we generously set $f = 5$

where $f(j|\mathcal{S}) = f(\mathcal{S} \cup j) - f(\mathcal{S})$ is the gain of adding element j to set $\mathcal{S}$. We iteratively execute the greedy step (5.4) until $t = k$ and $|\mathcal{S}^t| = k$. It is easy to see that the complexity of the greedy algorithm is $O(nkT_f)$, where T_f is the complexity of evaluating the gain $f(j|\mathcal{S})$ for the supervised and unsupervised facility location functions. We then significantly optimize this simple greedy algorithm via a lazy greedy algorithm [26] via memoization and precompute statistics [21].

Dataset	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{U}\rvert$	#Rules/LFs	Precision	%Cover	|Test|
MIT-R	1842	64888	15	80.7	14	14256
YouTube	100	1586	10	78.6	87	250
SMS	69	4502	73	97.3	40	500
Census	83	10000	83	84.1	100	16281
Ionosphere	73	98	64	65	100	106
Audit	162	218	52	87.5	100	233
IMDB	284	852	25	80	58.1	500

Table 5.4: Statistics of datasets and their rules/LFs. Precision refers to micro precision of rules. %Cover is the fraction of instances in U covered by at least one LF. Size of Validation set is equal to $|\mathcal{L}|$.

In this section, we (1) evaluate our joint learning against state-of-the-art approaches and (2) demonstrate the importance of subset selection over random subset selection. We present evaluations on seven datasets on tasks such as text classification, record classification and sequence labelling.

5.1.8 Datasets

We adopt the same experimental setting as in [6] for the dataset split and the labelling functions. However (for the sake of fairness), we set the validation data size to be equal to the size of the labelled data-set unlike [6] in which the size of the validation set was assumed to be much larger. We use the following datasets: **(1) YouTube**: A spam classification on YouTube comments; **(2) SMS Spam Classification** [5], which is a binary spam classification dataset containing 5574 documents; **(3) MIT-R** [23], is a sequence labelling task on each token with following labels: *Amenity, Prices, Cuisine, Dish, Location, Hours, Others*; **(4) IMDB**, which is a plot summary based movie genre binary classification dataset, and the LFs (and the labelled set) are obtained from the approach followed by [43]; **(5) Census** [13], **(6) Ionosphere**, and **(7) Audit**, which are

all UCI datasets. The task in the Census is to predict whether a person earns more than $ 50K or not. Ionosphere is radar binary classification task given a list of 32 features. The task in the Audit is to classify suspicious firms based on the present and historical risk factors.

Statistics pertaining to these datasets are presented in Table 5.4.

We employ the Wilcoxon signed-rank test [46] to determine whether there is a significant difference between SPEAR and Imply Loss (current state-of-the-art). Our null hypothesis is that there is not significant difference between SPEAR and Imply loss. For $n = 7$ instances, we observe that the one-tailed hypothesis is significant at $p < .05$, so we reject the null hypothesis. Clearly, SPEAR significantly outperforms Imply loss and, therefore, all previous baselines.

Similarly, we perform the significance test to assess the difference between SPEAR-SS and Imply Loss. As expected, the one-tailed hypothesis is significant at $p < 0.05$, which implies that our SPEAR-SS approach significantly outperforms Imply Loss, and thus all other approaches.

Bibliography

[1] Guttu Sai Abhishek, Harshad Ingole, Parth Laturia, Vineeth Dorna, Ayush Maheshwari, Ganesh Ramakrishnan, and Rishabh Iyer. Spear : Semi-supervised data programming in python, 2021.

[2] Vicent Alabau, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Ulrich Germann, Jesús González-Rubio, Robin Hill, Philipp Koehn, Luis Leiva, Bartolomé Mesa-Lao, Daniel Ortiz-Martínez, Herve Saint-Amand, Germán Sanchis Trilles, and Chara Tsoukala. CASMACAT: A computer-assisted translation workbench. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 25–28, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[3] Tamer Alkhouli, Gabriel Bretschner, and Hermann Ney. On the alignment problem in multi-head attention-based neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 177–185, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[4] I Elaine Allen and Christopher A Seaman. Likert scales and data analyses. *Quality progress*, 40(7):64–65, 2007.

[5] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262, 2011.

[6] Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. Learning from rules generalizing labeled exemplars. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[7] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*, pages 362–375, 2019.

[8] Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, and Matthias andigital Humanities Huck. Findings of the 2019 conference on machine translation (WMT19). volume 5, page 9. Frontiers, 2018.

[9] Claude Bédard. Mémoire de traduction cherche traducteur de phrases. *Traduire*, 186:41–49, 2000.

[10] Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. Robust data programming with precision-guided labeling functions. In *AAAI*, 2020.

[11] Guanhua Chen, Yun Chen, and Victor OK Li. Lexically constrained neural machine translation with explicit alignment guidance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12630–12638, 2021.

[12] Stephen Doherty. Translations| the impact of translation technologies on the process and product of translation. *International journal of communication*, 10:23, 2016.

[13] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[14] Marcello Federico, Nicola Bertoldi, Mauro Cettolo, Matteo Negri, Marco Turchi, Marco Trombetti, Alessandro Cattelan, Antonio Farina, Domenico Lupinetti, Andrea Martines, et al. The matecat tool. In *COLING (Demos)*, pages 129–132, 2014.

[15] Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *arXiv preprint arXiv:2104.14478*, 2021.

[16] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[17] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[18] Nico Herbig, Tim Düwel, Santanu Pal, Kalliopi Meladaki, Mahsa Monshizadeh, Antonio Krüger, and Josef van Genabith. Mmpe: A multi-modal interface for post-editing machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1691–1702, 2020.

[19] Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[20] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

[21] Rishabh Iyer and Jeff Bilmes. A memoization framework for scaling submodular optimization to large scale problems. *arXiv preprint arXiv:1902.10176*, 2019.

[22] Dongjun Lee, Junhyeong Ahn, Heesoo Park, and Jaemin Jo. Intellicat: Intelligent machine translation post-editing with quality estimation and translation suggestion. *arXiv preprint arXiv:2105.12172*, 2021.

[23] Jingjing Liu, Panupong Pasupat, Yining Wang, Scott Cyphers, and Jim Glass. Query understanding enhanced by hierarchical parsing structures. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 72–77. IEEE, 2013.

[24] Ayush Maheshwari, Oishik Chatterjee, KrishnaTeja Killamsetty, Rishabh K. Iyer, and Ganesh Ramakrishnan. Data programming using semi-supervision and subset selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.

[25] Ayush Maheshwari, Krishnateja Killamsetty, Ganesh Ramakrishnan, Rishabh Iyer, Marina Danilevsky, and Lucian Popa. Learning to robustly aggregate labeling functions for semi-supervised data programming, 2021.

[26] Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer, 1978.

[27] Joss Moorkens, Stephen Doherty, Dorothy Kenny, and Sharon O'Brien. A virtuous circle: laundering translation memory data using statistical machine translation. *Perspectives*, 22(3):291–303, 2014.

[28] Masaaki Nagata, Chousa Katsuki, and Masaaki Nishino. A supervised word alignment method based on cross-language span prediction using multilingual bert. *arXiv preprint arXiv:2004.14516*, 2020.

[29] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical programming*, 14(1):265–294, 1978.

[30] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[31] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[32] Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Kumar Deepak, Vivek Raghavan, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Shantadevi Khapra. Samanantar: The largest publicly available parallel corpora collection for 11 indic languages, 2021.

[33] Alexander J Ratner, Stephen H Bach, Henry R Ehrenberg, and Chris Ré. Snorkel: Fast training set generation for information extraction. In *Proceedings of the 2017 ACM international conference on management of data*, pages 1683–1686, 2017.

[34] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575, 2016.

[35] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343, 2018.

[36] Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. Simalign: High quality word alignments without parallel training data using static and contex-

tualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, 2020.

[37] Sebastin Santy, Sandipan Dandapat, Monojit Choudhury, and Kalika Bali. Inmt: Interactive neural machine translation prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 103–108, 2019.

[38] Lane Schwartz, Isabel Lacruz, and Tatyana Bystrova. Effects of word alignment visualization on post-editing quality & speed. In *Proceedings of Machine Translation Summit XV: Papers*, 2015.

[39] Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[40] Antonio Toral, Martijn Wieling, and Andy Way. Post-editing effort of a novel with statistical and neural machine translation. *Frontiers in Digital Humanities*, 5:9, 2018.

[41] Jan Van den Bergh, Eva Geurts, Donald Degraen, Mieke Haesen, Iulianna Van der Lek-Ciudin, and Karin Coninx. Recommendations for translation environments to improve translators workflows. In *Proceedings of Translating and the Computer 37*, 2015.

[42] Vincent Vandeghinste, Tom Vanallemeersch, Liesbeth Augustinus, Bram Bulté, Frank Van Eynde, Joris Pelemans, Lyan Verwimp, Patrick Wambacq, Geert Heyman, Marie-Francine Moens, et al. Improving the translation environment for professional translators. In *Informatics*, volume 6, page 24. Multidisciplinary Digital Publishing Institute, 2019.

[43] Paroma Varma and Christopher Ré. Snuba: Automating weak supervision to label training data. *Proc. VLDB Endow.*, 12(3):223236, November 2018.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[45] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963, 2015.

[46] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[47] Marcos Zampieri and Mihaela Vela. Quantifying the influence of mt output in the translators performance: A case study in technical translation. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 93–98, 2014.