Infrequent Discourse Relation Identification Using Data Programming

by

Xing Zeng

H.B.Sc., University of Toronto, 2016

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

December 2018

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Infrequent Discourse Relation Identification Using Data Programming

submitted by Xing Zeng in partial fulfillment of the requirements for
the degree of Master of Science
in Computer Science

Examining Committee:

Giuseppe Carenini, Computer Science
Co-supervisor

Raymond Ng, Computer Science
Co-supervisor

Laks Lakshmanan, Computer Science
Additional Examiner

Abstract

Discourse parsing is an important task in natural language processing as it supports a wide range of downstream NLP tasks [2, 5, 14, 15, 21, 33]. While the overall performance of discourse parsing has been recently improved considerably [32, 44], the performance on identifying relatively infrequent discourse relations is still rather low ($\sim$ 20 in terms of F1 score).

To resolve the gap between the performance of infrequent and frequent relations, we propose a novel method for discourse relation identification that is centered around "a paradigm for the programmatic creation of training datasets," called Data Programming (DP) [35]. The main idea in our approach is to overcome the issue of limited labeled data for infrequent relations by leveraging unlabeled data in addition to labeled data. Our experiments show that our method improves the performance on most of the infrequent relations with minimal negative effect on frequent relations.

Lay Summary

Sentences appear in a coherent paragraph usually have some relations between them. Some past research has been done on automatically identifying these relations with a primary focus on improving the overall correctness. However, the overall correctness hides a lot of detail. The reality is, there are several "types" of relations. The past work has been having high correctness on some types only, while for the rest of the types the correctness rate could be meager. It has been shown that the type of relations that usually has low correctness rate are those that have only a limited amount of training data. To solve this problem, we would like to apply a framework called "Data Programming" that supports adding more training data. Our experiments show that our approach made some improvements to the correctness rate on the relations that only have a limited amount of training data.

Preface

This thesis is an original intellectual product of the author, Xing Zeng. All specifications in approach, experiments, and writings, are done by the Xing Zeng with help from Dr. Hyeju Jang under the supervision of Dr. Giuseppe Carenini and Dr. Raymond Ng. Certain parts of this thesis is based on [45]. Xing Zeng is also the primary author of that publication.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my supervisors, Dr. Giuseppe Carenini and Dr. Raymond Ng. They have been very professional in the field of research, and I have learned so much from them.

I would like to thank Dr. Hyeju Jiang for providing extra help on improving the language used by this thesis.

I would like to thank Dr. Laks Lakshmanan for his additional suggestions for my thesis.

I would like to thank my parents, Jianguo Xiang and Xian Zeng, for supporting my studies in Canada.

Chapter 1

Introduction

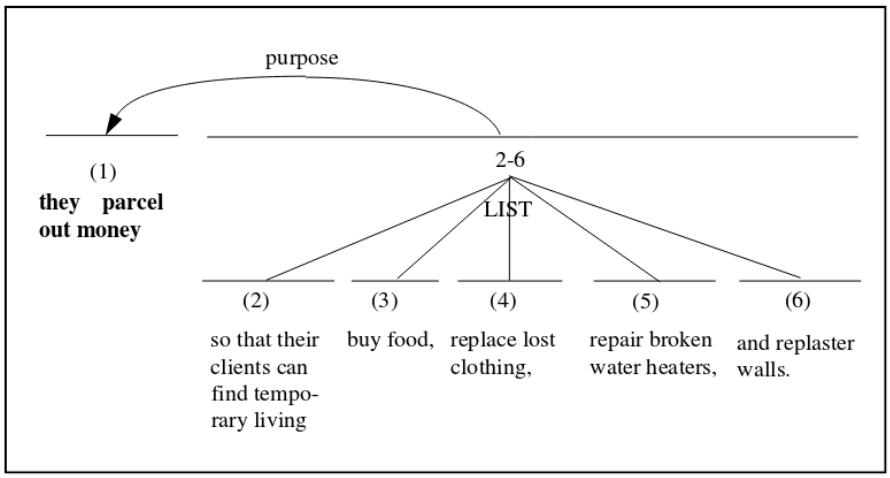

Figure 1.1: Example Discourse Tree from [8]

Discourse parsing is the task of parsing a span of text into its rhetorical structure. The rhetorical structure can be seen as consisting of two parts. The first part is about how each portion of the text connects to each other. This would require us parsing the whole text into a tree structure. The other part is about what the rhetoric is concerning each connection in the previous part. This would require us knowing what the actual relation on each node in the tree structure is. The latter part of the task is also called discourse relation identification, which is the primary task of our thesis. Figure 1.1 shows an example discourse structure for sentence "They parcel out money so that their clients can find temporary living, buy food, replace lost clothing, repair broken water heaters, and replaster walls", extracted from [8]. Here, the links and arrows between every text span refer to the tree structure of the text, where "purpose" and "list" are two discourse relations.

Discourse parsing provides a lot of understanding on the structure of the text. Its sub-task, discourse relation identification, provides knowledge about the actual relation between each part of the structure. Because of this, discourse parsing can support a wide range of downstream NLP tasks. This includes sentiment analysis [5, 21, 33], text understanding [2], summarization [14, 15], and vote prediction [21]. There are multiple ways of bringing discourse parsing into NLP tasks. (Allen et al., 2014)[2] utilized certain aspects of the result from a discourse parser as

features to detect disagreement. (Gerani et al., 2014, 2016) [14, 15] combined a pre-defined template with the hierarchy of entities created using the result from a discourse parser for text generation. (Bhatia et al., 2015; Ji & Smith, 2017; Nejat et al., 2017) [5, 21, 33] constructed a recursive neural network based on the discourse structure as well as the relation generated by the parser to learn a discourse informed representation for text spans.

The primary focus of our thesis is on discourse relation identification. Currently, most existing discourse parsers generally have good overall performance on this problem (close to 60 in terms of F_1 score). However, this "overall performanc" hides a lot of detail. If we look at the performance of each individual relation, the result is less exciting. While we have some relations that have excellent performance (> 80 in terms of F_1 score), there are a lot of relations that are barely working (< 20 in terms of F_1 score). The per-relation performance of the state-of-the-art discourse parser from [44] is listed in Table 1.1 and a more detailed definition of the per-relation performance metrics is described in Section 4.2.

(Jiang et al., 2016)[22] identified what might cause some of these relations to be poorly performed. Not surprisingly, the per relation performance is highly correlated with the size of the training data for that specific relation. Jiang's work is done using two discourse parsers from [23] and [20] as examples. This observation is still the same for some newer models, including the state of the art. In Figure 1.2, we plot the graph of per relation performance versus per relation training data size, on the state of the art discourse parser [44], in together with the fitted regression line. The fitted regression line does not directly match with all the data points. This suggests us that for the state of the art model there are also other factors that are also affecting each relation's performance other than the size of the training data for that specific relation. However, the fitted regression line still have a $r = 0.4826$ as the Pearson coefficient of correlation [37] and $p = 0.043$. This suggest us that the per relation performance is still highly correlated with the size of the training data for that specific relation.

Relation	F_1 **score**
Attribution	86.5
Same-Unit	82.8
Condition	62.5
Enablement	56.4
Elaboration	51.9
Joint	48.9
Summary	45.7
Manner-Means	42.0
Textual-Organization	33.2
Contrast	30.4
Background	30.1
Explanation	24.8
Comparison	14.0
Temporal	10.7
Cause	9.7
Topic-Change	2.6
Topic-Comment	0.0
Evaluation	0.0

Table 1.1: The performance across all 18 relations for the state-of-the-art discourse parser [44], ordered by their F_1 score

1.1 Motivation

Following the observation that the per-relation performance is highly correlated with the frequency, (Jiang et al., 2016) [22] divided these relations into two groups: the top 10 *most frequent relations* and the 8 *infrequent relations*. Both are shown in Table 1.2 with a more detailed analysis of this division described later in Section 4.1. Then, Jiang et al. apply the co-training [6] technique on two discourse parsers, and use the results from a weaker discourse parser [20] to help add more training data for the stronger discourse parser [23]. Jiang's approach shows some improvements in the performance of infrequent relations. However, it suffers from several issues. Firstly, Jiang's approach has negative effects on the per-relation performance of many frequent relations. Secondly, to conduct co-training effectively, their approach requires two discourse parsers to be operated on two feature sets

Figure 1.2: F1 v.s. Size of Training Data, per Relation, for the state-of-the-art discourse parser [44]

Relation	Counts
Elaboration	7106
Attribution	2727
Joint	1775
Same-Unit	1277
Contrast	984
Explanation	876
Background	826
Cause	611
Enablement	522
Temporal	457
Total	17161

Relation	Counts
Evaluation	519
Comparison	280
Condition	274
Topic-Change	199
Manner-Means	192
Summary	191
Textual-Organization	148
Topic-Comment	132
Total	1935

Table 1.2: Top 10 Frequent relations (left) and the 8 infrequent relations (right), ordered by number of occurrences

that are different from each other. However, this is not necessarily achievable if we would like to apply newer classes of discourse parsers. In fact, the state of the art discourse parser [44] have features from both [23] and [20], making it virtually

impossible to find an alternative classifier to operate on. Lastly, co-training is less systematic because there are no explicit guidelines over how to deal with conflict labels.

In order to tackle the problems in Jiang's work, we are interested in applying a framework called "Data Programming", introduced by [35], into discourse relation identification. Data Programming shares some similarity with the work of Jiang but is more general beyond just co-training alone. It enables us to learn the accuracies of each individual labeling source systematically. Good results have been shown utilizing this framework on some classification tasks that only have limited labels like relation extraction.

Ideally, we hope this framework would be a more systematical approach to replace co-training. It may also solve the issue of requiring two different feature sets expected by co-training. This is because Data Programming only requires a more relaxed assumption: outputs from all labeling sources should be conditionally independent of each other given the true label. Moreover, in previous empirical study [4], it has been shown that even if the independence assumption is not carefully followed, good results can still be sometime achievable.

However, Data Programming cannot be directly applied to our problem. There are two reasons for this. First is because Data Programming requires using high accuracy labeling functions. However, obtaining high accuracy labeling functions for discourse relation identification is not a trivial task. Second is despite the fact that the Data Programming framework is a general framework which supports both binary classification and multi-class classification, most of the previous empirical results are done towards binary classification problems and we have found it to be non-trivial to apply it to a multi-class scenario.

1.2 Approach and Contributions

The primary objective of this thesis is to find a workflow that incorporates Data Programming into the discourse relation identification problem, in order to boost the performance of infrequent relations. In particular, we are aiming at two issues. First, we are aiming at creating high-performance labeling functions that work well with Data Programming. Second, we are looking for methods that can utilize the

output of Data Programming for discourse relation identification.

In order to create labeling functions, we utilize ideas from Bootstrap Aggregating method [7] to create labeling functions using sub-samples of training data. However, doing this would not be enough since Data Programming requires labeling functions to have higher accuracies which are not achievable even if we use all the training data. We consider two approaches to tackle this issue. One is to use filtering based on labeling functions' confidence scores. This is common in semi-supervised learning [38] and is also used by [12, 22] for discourse parsing. This would require us to filter out those that have lower confidence score before passing the predictions to Data Programming framework. Besides filtering, we are considering applying techniques from Few-Shot Learning [39] into labeling functions. These models would ideally work better if there are only a few training samples in presence, which is exactly the scenario we have when creating labeling functions using Bootstrap Aggregating. After we create the labeling functions, we invoke the Data Programming framework. After this process, Data Programming would output a probabilistic distribution of labels for each unsupervised instance. Then, we train another classifier as the final model utilizing both ground truth label and output from Data Programming using a loss function designed by us. A more detailed description of our approach is listed in Chapter 3.

Our experiments have shown some performance improvements on infrequent relations on some scenarios when incorporating Data Programming framework into discourse parsing. More specifically, we show that our way of creating labeling functions is effective for providing labels for the Data Programming framework. We also show an effective way of training neural network models with output from Data Programming. However, we also find several issues in our approach. We find the result of utilizing techniques from Few-Shot Learning to be not significantly more effective in terms of providing better performance on infrequent relations. We also find that Data Programming does not reach significant improvement on the ablation test over majority voting. A more detailed description of our results is listed in Chapter 4.

More specifically, the contributions we make in this thesis are:

- Present a workflow for applying Data Programming framework to the task

of discourse relation identification.

- Develop an ensemble method based mechanism for creating labeling functions for Data Programming framework.
- Experiment with a new filtering mechanism that works well with the workflow we presented.
- Develop a loss function for utilizing the output of Data Programming for training a final discourse relation identification model.
- Explore one Few-Shot learning technique on discourse relation identification problem for both creating labeling functions for Data Programming and used as the standalone prediction model.

The advantages of our approach are:

- It does not require any expert to label additional documents or provide handwritten labeling functions.
- It is effective in boosting most of the infrequent relations.
- It is effective across a wide range of settings for neural networks.
- The negative effect on the performance of frequent relations is negligible.

The disadvantage of our approach are:

- It is unable to boost the performance of all infrequent relations.
- It is not effective on linear models.
- In our workflow Data Programming itself does not significantly outperform majority voting.
- We did not evaluate if the techniques we proposed can be generalized to other problems.

Chapter 2

Related Work

2.1 Existing Discourse Parsers

A lot of the research has been done on creating discourse parsers. In this section, we will introduce several of them, grouped by the machine learning model they use.

Some of the earliest attempts to discourse parsing focus on graphical models, for their convenience in measuring dependencies. (Soricut et al., 2003) [40] were among the first with their development of the SPADE system. In SPADE, a simple graphical model is used. This graphical model only has one dependency assumption: each prediction made on a text span should only be dependent on the prediction previously made on some of its children. SPADE only works for parsing a sentence and does not support document-level parsing, however. (Joty et al., 2015) [23] introduced CODRA later on. CODRA has a more sophisticated Conditional Random Field (CRF) based model. This model involves one linear chain CRF for each of the possible configurations at all levels. The final predictions are made based on a Cocke-Kasami-Younger (CKY) style optimal parsing algorithm [10] using predictions gathered from all possible configurations at all levels. On the other hand, (Feng et al., 2014) [11] introduced another model which is similar to CODRA as both models utilize linear-chain CRF. However, Feng use a greedy algorithm for parsing. The greedy algorithm enables it to have a linear time parsing performance instead of the $O(n^3)$ runtime CODRA has.

Support Vector Machine (SVM) based models are also popular in discourse parsing. This is partly because historically a majority part of the features used in discourse parsing is Bag of Words. The use of Bag of Words features results in a sparse feature space where a max-margin classifier like SVM can work very well. (Hernault et al., 2010) [19] were among the first with their development of the HILDA system. (Ji et al., 2014) [20] introduced DPLP later which is also an SVM based approach. Different from HILDA, DPLP uses parameter tying on the SVM side. Moreover, (Wang et al., 2017) [44] utilized the observation that discourse parsing can be separated into the task of intra-sentential parsing, sentential parsing, and paragraph parsing. The three tasks are very different from each other, both from the distribution of labels and from the features they should use. Based on this observation, they train one SVM for each of the components. Currently, Wang's

approach is the state of the art result in discourse parsing.

In recent years, as neural networks and deep learning methods start to become a major theme in natural language processing, more interests have been shifted to using neural networks for discourse parsing. (Li et al., 2014) [27] were among the first who introduced deep learning techniques in discourse parsing. They implement one recursive neural network structure based on the binary representation for each discourse tree. Every node in this recursive neural network represents a node in the original discourse tree. If the node is a leaf, the representation of this node will come from an LSTM running on the text span of this node only. If not, the representation of this node would be based on the representation of its direct child nodes. (Li et al., 2016) [28] further refined the idea of creating representation based on child nodes. However, they replace the deep binary discourse tree with sentence-level attention mechanism. The use of attention mechanism provides a shallower structure compared to the previous work and are thus easier to train. As both utilize CKY style parsing algorithm which has an $O(n^3)$ complexity, (Liu and Lapata, 2017) [30] designed another neural network based model for discourse parsing called CIDER. CIDER has a linear time performance by utilizing a CRF similar to [11] for parsing, with features taken by the CRF coming from the deep feature learned by the neural network. CIDER also includes contextual information in prediction by running a LSTM over the whole document.

Nevertheless, all the models above face issues in predicting infrequent relations, which is the main issue we address in this thesis.

2.2 Learning with Limited Labeled Data

Infrequent discourse relation identification can be essentially seen as a type of learning with limited labeled data problem. Recently there has been more interests in this problem. This phenomenon occurs because of the limitation of deep learning. Deep learning methods have provided numerous performance improvement on various problems in machine learning. However, most of their successes rely on the abundance of labeled training data. Deep learning related methods usually do not work very well when there is only a limited amount of training data. When the training data is small, they often can only perform no better than simple

machine learning methods like Support Vector Machine. Learning with Limited Labeled Data is still a much immature field with various different approaches from different angles have been proposed. We will only focus on those that are more related to our work.

2.2.1 Self-training and Co-training

Self-training and co-training are among the earliest attempts in semi-supervised learning because of their simplicity.

Self-training [18] [38] is a simple heuristic based approach. In this approach, a machine learning model is first trained in a supervised way on a possibly small set of labeled data. Then this model is used to make predictions on the unlabeled data. Next, it adds some of the predictions, filtered based on certain pre-defined rules, to the training set. Another new classifier is trained on the new training set. This process can be repeated several times.

Co-training [6] [38] is an extension of self-training. It relies on two "views" of data which are usually implemented on the same dataset but leverage two different feature sets. Ideally, the two feature sets should have the following two properties. Firstly, they should be independent of each other if the label is known. Secondly, each one of them should be sufficient enough to infer the true label. Under the two settings listed above, two classifiers are trained on each of the feature sets. Then, for each of the classifier, new data are added to the training set for the current classifier using information from the other classifier. This method suffers less from the problem of the inherent bias of only using one classifier, a problem of self-training.

These approaches have been applied to Discourse Parsing problem, with some successful result. (Fisher and Simmons, 2015) [12] explored self-training technique to shallow discourse parsing. They adopt an HMM-like model which is common for shallow discourse parsing. Then, they use this model to predict the relation label of the unlabeled instances. It is also used to calculate the density estimation of each unlabeled instance. The density estimation, when weighted with the entropy of the currently predicted relation, is being used to decide if the current prediction should be filtered or not. Their approach shows a 9% improvement on

the total performance across all relations.

(Jiang et al., 2016) [22] applied the co-training technique to discourse parsing. They use two discourse parsers: one is CODRA [23] and another is a simplified version of DPLP [20]. The two parsers are very different from each other. CODRA uses various features from structure information, lexical chains, text organization and dominance set, and uses a CKY style parsing algorithm to construct the label probability for each possible tree structure. On the other hand, the simplified version of DPLP uses Bag of Words as features, and uses a shift-reduce style parsing algorithm to construct the discourse tree greedily. Jiang's approach shows improvement in the performance of infrequent relations. However, there are some negative effects on some frequent relations like Elaboration, Attribution, Joint, and Explanation.

2.2.2 Heterogeneous Supervision

Heterogeneous supervision refers to the scenario where multiple sources of weak supervision are applied for unlabeled data. In this type of situation, for each unlabeled instance, different sources of weak supervision may provide conflicting supervision labels. A trivial way to deal with the conflict is taking majority voting or randomly selecting one. But recently more systematical approaches have been suggested where we could use machine learning method to model the dynamics of supervision from different sources.

(Ratner et al., 2016) [35] introduced a way that employs "a paradigm for the programmatic creation of training datasets" which they called "Data Programming". In this framework, heterogeneous sources of automatic labeling functions and unlabeled data are required as inputs. The labeling functions are required to have high precisions ($> 80\%$ is recommended), but are not expected to be always correct. Moreover, these labeling functions are not required to always output a label when seeing an unlabeled data instance. After receiving all the inputs, the framework learns the accuracies for each of the labeling functions by looking at how well the labeling functions would label the unlabeled data by comparing them against each other. These accuracies are then being used to generate a distribution over all possible labels for each unsupervised data point.

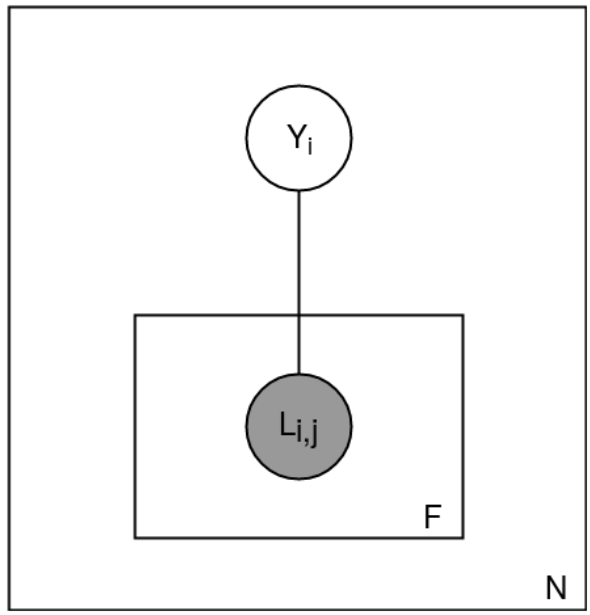

Figure 2.1: The generative model structure of Data Programming.

More specifically, in the following part we will use these notations:

- F: number of labeling functions
- N: number of unsupervised data instances
- Y_i: the true label on instance i which without the loss of generality we will force it to be non zero
- α_j: the probability of labeling function j gives the correct label across all possible instances
- β_j: the probability of labeling function j successfully gives a label
- L_{ij}: the label given by labeling function j on instance i where $L_{ij} = 0$ means that it fails to infer a label
- O_i: output of the final classification model on instance i
- S: any standard loss function, including but not limited to hinge loss, softmax cross entropy loss, and least square errors.

Data Programming aims at learning a generative model shown in Figure 2.1, with the following distribution:

$$\forall i \in \{1,...,N\}, P(Y_i, L_i) = \prod_{j=1}^{F} (\beta_j(\alpha_j \times \mathbf{1}_{[Y_i = L_{ij}]} + (1-\alpha_j)\mathbf{1}_{[Y_i \neq L_{ij}]}) + (1-\beta_j)\mathbf{1}_{L_{ij}=0})$$

In order to learn the optimal α, β, we maximize the probability given by our observations, aggregating over all possible hidden values:

$$\alpha, \beta = \arg\max_{\alpha,\beta} \sum_{i=1}^{N} \log(\sum_{Y_i} P(Y_i, L_i))$$

This optimization can be done easily using Stochastic Gradient Descent.

After we successfully gather the value of α, β, for each data point, given by the observation of the labels coming out from the labeling functions, we have:

$$P(Y_i|L_{ij}) = \frac{P(Y_i, L_{ij})}{P(L_{ij})} \propto P(Y_i, L_{ij})$$

After that, we train a final machine learning model that utilize a noise-aware empirical loss function:

$$\text{loss} = \sum_{i=1}^{N} \mathbf{E}_{Y_i \sim P(Y_i|L_i)}[S(Y_i, O_i)]$$

The Data Programming approach has been applied to NLP tasks, with success in relation extraction task and sentiment analysis task [3].

The Data Programming approach is similar to the work of (Liu et al., 2017) [29]. However, Liu use a more sophisticated model. One of the weaknesses in Data Programming is that for each labeling function, the accuracy is the same across all possible data instances. However, this assumption usually does not hold. In reality, for all the instances that a specific weak supervision approach fails to provide a correct label, they may have a similar representation. This is because intuitively each weak supervision approach would make similar mistakes for similar cases. Because of this, they use a logistic regression from the representation of the data to model how accurately can a labeling function generate a correct label for a specific instance. Besides, instead of separating the accuracy training of labeling functions and the final machine learning model training, they combine these two as well as

representation learning together into one model, and train it from end to end.

2.2.3 Few Shot Learning

Few shot learning refers to the classification task on a training set where there are only a few training samples per class. This type of task is usually required to be done without using any external data source for transfer learning. Despite the hardness of this type of problem, there have been numerous recent works on this problem on multiple computer vision datasets like omniglot [26] or mini-imagenet [43] with successful results. In most of the recent works, "few" usually refer to 5 [39] or even 1 [25] samples per class. However, in our case, the number of samples for each infrequent relation can usually reach more than 100. While this number is much larger, since Natural Language Processing is still a much different problem from Computer Vision, and we may create labeling functions using less than 100 samples for some labels, our task is still related to Few Shot learning.

Most of these works focus on learning an algorithm that map the input, usually can be seen as in $\mathbb{R}^n$, into a task-specific metric space in $\mathbb{R}^m$, with $m << n$. In the final metrics space where the dimension is small, the classification can be done in a more efficient way [13, 25, 39, 43].

For instance, in Prototypical Network [39], for each class, one learns a "prototype" vector for each class, and the prediction is made base on closeness to the prototype.

More specifically, in the following part we will use these notations:

- $f : \mathbb{R}^n \to \mathbb{R}^m$: a learnable transformation function which is usually a neural network
- k: the id of the class
- S_k: the set of all the input training data that are of class k, which by itself is a subset of $\mathbb{R}^n$

Then the prototype vector p_k is given as:

$$p_k = \frac{1}{|S_k|} \sum_{x \in S_k} f(x)$$

where this is learned iteratively whenever $f(x)$ is updated.

The class prediction is given as:

$$p(y = k|x) = \frac{\exp(-d(p_k, f(x)))}{\sum_{k'} \exp(-d(p_{k'}, f(x)))}$$

where d could be any distance metrics function. In the original work for Prototypical Network [39], they choose squared Euclidean distance.

2.2.4 Summary

In this section, we list some previous approaches to the generic learning with limited labeled data problem. Self-training and co-training have been applied to discourse parsing problem before, with some successful results but are also limited in certain aspects. On the other hand, Heterogeneous supervision approaches, most notably Data Programming, have not been applied to discourse parsing yet. This is the main goal of our thesis. However, applying Data Programming to discourse relation identification is not trivial. One issue we need to tackle is to provide the sources of supervision. In order to build these sources, standard machine learning methods can be used but we are also curious to see if Few-Shot learning techniques may be also helpful. Our workflow is described in detail in the next Chapter.

Chapter 3

Our Approach

Our approach is centered around Data Programming. However, just as we previously mentioned, the approaches in Data Programming cannot be directly applied to our problem. There are two reasons for this. First is because of how the labeling functions are constructed in Data Programming's original work. Second is because Data Programming's original way of training a final model is not ideal for our problem.

```
def lambda_1(x):
    return 1 if (x.gene,x.pheno) in KNOWN_RELATIONS_1 else 0

def lambda_2(x):
    return -1 if re.match(r'.*not␣cause.*', x.text_between) else 0

def lambda_3(x):
    return 1 if re.match(r'.*associated.*', x.text_between)
            and (x.gene,x.pheno) in KNOWN_RELATIONS_2 else 0
```

Figure 3.1: Example of hand-written labeling function from [35]

In Data Programming's original work, the labeling functions are usually heuristic based and are hand-written by experts. This allows them to have a precision close to 80% while having a reasonable coverage across multiple cases. Figure 3.1 shows an example of what some of the labeling functions look like for the Genomes Extraction task in Data Programming's original work [35]. However, this is unfeasible in discourse relation identification since there are too many variations that are too hard to capture by simple hand-written functions [31].

Besides labeling functions, we can not directly follow Data Programming's approach in training a final model. We will have both data with ground truth label and data with noisy label provided by Data Programming during the final training. Ideally, we need to take both into account when training a final model. Besides, Data Programming's final loss function is optimized for binary classification problems. In a multi-class scenario, while one could directly borrow the loss function from the binary case, we find it to be not ideal.

In the following of this Chapter, we describe our ways to tackle the problems listed above. In Section 3.1.1 we introduce an alternative method of creating labeling functions without the involvement of human experts. In Section 3.1.2 we describe how we conduct the final multi-class training using both outputs from Data

Programming and ground truth labels. In Section 3.2 we list the classifiers we use for the previous two sections, including those that are based on Few-Shot Learning techniques, which are introduced with the hope that Few-Shot Learning techniques could create better labeling functions compared to traditional classifiers.

3.1 Our workflow

3.1.1 Training Labeling Functions

Our aim is to create labeling functions for discourse relation identification. We need these labeling functions to have high precisions, ideally close to the 80% recommendation set up by Data Programming [35]. We also hope the classifier we created would have reasonable coverage across different cases. In our work, we focus on creating labeling functions from some ensemble machine learning methods. Our idea is similar to the Bootstrap Aggregating method [7]. We randomly sampled half of the labeled training data four times. This creates four different but possibly overlapping dataset. Then, on each of the dataset, we create a single classifier on top of it. The schematic graph of this process is shown in Figure 3.2.

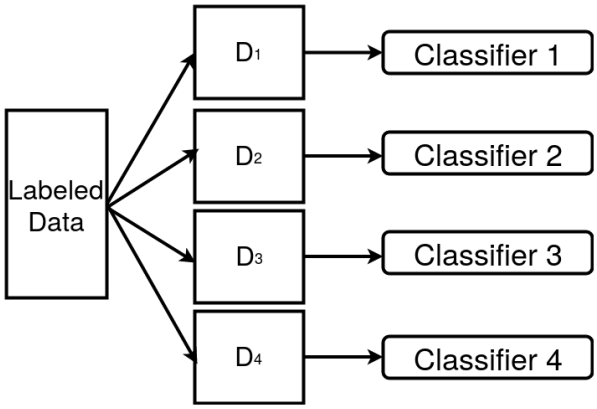

Figure 3.2: Schematic graph of the creation of labeling functions.

However, these classifiers do not meet the precision requirement of Data Programming yet. Theoretically, Data Programming only work well if each labeling function has an precision of more than 80% [35]. In order to tackle this issue, we

applied an idea that is commonly used in most self-training methods [38]. The idea is to utilize the internal confidence scores of classifiers for filtering. Ideally, if we filter out predictions that have relatively low confidence scores according to the classifier, then we would boost the precision of our labeling functions. Moreover, for the instances where the confidence score is too low, it can be just treated as unknown, which is an acceptable type of input to Data Programming framework. The schematic graph of this process is shown in Figure 3.3.

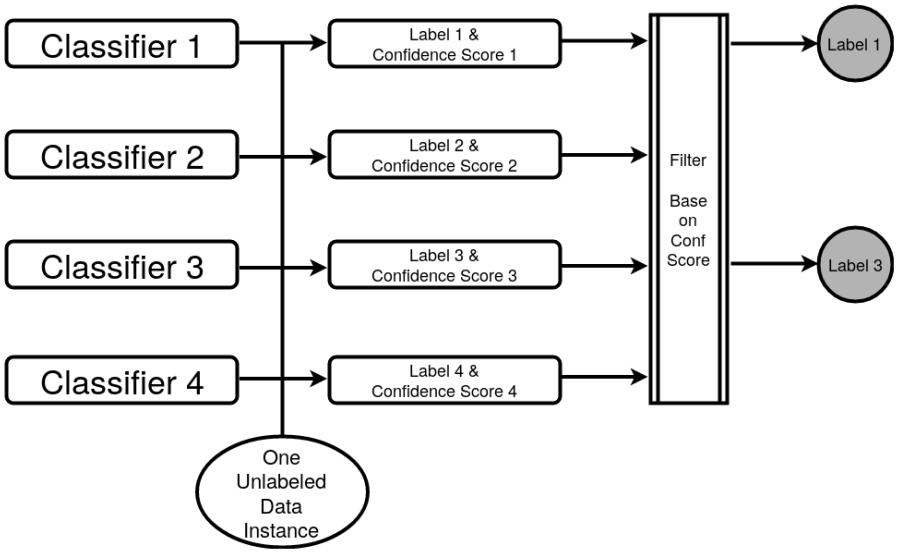

Figure 3.3: Schematic graph of how the filtering is done

We suggest two ways of filtering. One way is to set up a uniform boundary for all labels. This is a somewhat naive approach that had previously been used by [22]. In this method, as long as the confidence score is higher than a fixed pre-defined boundary, regardless of what the prediction is, it is kept as input to Data Programming. Otherwise, it is removed. Another way of filtering is we allow each of the possible classes predicted by labeling functions to have different pre-defined boundaries, and dynamically adjust the boundaries such that the distribution of labels selected is the same as the distribution of labels in the training set. This idea of having different boundaries for each label is inspired by (Jiang et al., 2016) [22] and ideologically similar to the filtering techniques used by (Fisher and Simmons,

2015) [12]. Ideally, having different boundaries may tackle one issue in uniformed boundary: classifiers are very likely to fit more towards commonly appeared labels, so for a specific confidence score, there might be much more frequent labels than infrequent labels, which cause the final model to be even more likely to fit more towards common labels instead of infrequent labels.

After we conduct filtering and only keep the predictions that have higher confidence scores, we could invoke Data Programming in the standard way as previously described in Section 2.2.2. In this process, the precisions of each of the labeling functions are learned.

3.1.2 Final Training

In this section, we describe how to create a final model for discourse relation identification. After learning the precision for each labeling function, for each unsupervised instance, we could use the output of labeling functions on that instance to give rise to a probabilistic distribution across all possible labels. Then, we could train another classifier that utilizes these probabilistic distributions. This classifier is being used as the final classification model.

Here, we again conduct slightly different approaches from the standard Data Programming. In standard Data Programming, data with ground truth label is usually not present so the loss function should only take into account the weakly labeled data [35]. However, in our case, we have both data with ground truth label from the training set and the weakly labeled data. So our loss function could take both into account.

Besides, we are not facing a binary task here, and the original noise-aware loss function is optimized for binary tasks. So we slightly alter the loss function for the weakly labeled part. In our work, for each unsupervised instance, we only keep the probability estimations of labels that show up in the results of the labeling functions operating on that instance. For the probability estimations of labels not shown up in the labeling functions on that instance, we remove them. This is because in a binary classification problem, if we assume that one labeling function is giving the wrong label on an instance, we could directly conclude that the true label for that instance is precisely the opposite of the one offered by the labeling function. However, in

a multi-class scenario, if we assume that one labeling function is giving the wrong label on an instance, we only know that the true label is among one of the rest possible labels while not knowing which one actually is. So we have to average the probabilities to all other labels. This would be way too noisy for a final classifier to learn.

More specifically, in the following part we will use these notations:

- O_i: output of our final model
- $L_{wl,i}$: the output of all labeling functions on unlabeled instance i
- $Y_{wl,i}$: the label from Data Programming on unlabeled instance i
- $Y_{gt,i}$: the ground truth label on labeled instance i
- S: any loss function.

Then the loss function takes the following form:

$$\begin{aligned}
\text{loss from ground truth} &= \sum_{i=1}^{N_{gt}} (S(Y_{gt,i}, O_{gt,i})) \\
\text{loss from weakly labeled} &= \sum_{i=1}^{N_{wl}} \sum_{Y_{wl,i}} \mathbf{1}_{Y_{wl,i} \in L_{wl,i}} P(Y_{wl,i} | L_{wl,i}) S(Y_{wl,i}, O_{wl,i}) \\
\text{final loss} &= \text{loss from ground truth} + \text{loss from weakly labeled}
\end{aligned} \tag{3.1}$$

3.2 Classifiers Used in Our Approach

In our approach, we need classifiers both as labeling functions and as the final method. We mostly focus on two types of models: simple neural network based classifiers and Few-Shot Learning based classifiers. The use of simple neural network based classifiers is because usually they benefit more from adding more data compared to most other machine learning methods while they are still capable of learning an acceptable labeling function when only half of the training data are in presence. The use of Few-Shot Learning based classifiers is because usually they are even better in terms of providing higher accuracies for infrequent relations during the training of labeling functions where we only have half of the data. We also

incorporate some linear methods for comparison.

More specifically, we combined the following models in our approach:

NN0: A simple feed-forward neural network with one hidden layer and softmax layer for prediction.

NN1: Same as **NN0**, except the learning rate is not fine-tuned. This is to simulate the scenarios where in some neural networks the learning rate is very hard to tune right.

NN2: Same as **NN1**, except added dropout [41] to the first hidden layer. Dropout is a common approach in neural networks, and we would like to see if adding dropout may bring any differences.

NN3: This is a neural network where we design a Prototypical Network [39] for discourse relation identification. We hope this technique from Few Shot Learning could further reduce the issues in creating labeling functions. It has the same hidden layer as **NN1** has, but the final softmax layer is replaced by calculating the distance to the prototype and make predictions based on the distance metric. Besides, different from [39], instead of training in an iterative way where the prototype is fixed in the training iteration, since the number of samples we have is much more than what they had in [39], we conduct a slightly different way of learning the class prototype. Our model essentially treats prototype as a variable that we need to update in Stochastic Gradient Descent, and train the whole model from end to end using Stochastic Gradient Descent.

NN4: Same as **NN3** except we add dropout [41] to the first hidden layer. Dropout is a common approach in neural networks, and we would like to see if adding dropout may bring any differences.

SVM: This is a standard Linear SVM model. It is the same as the one used in the state of the art from [44]. It is being used for two purposes. First, we would like to see if our approach can help linear methods. Second is we would like to know if we could improve over the state of the art.

LR: This is a standard Logistic Regression model. It is also being used to see if our approach can help linear methods.

3.3 Summary

In this chapter, we describe various design choices we proposed to incorporate Data Programming into discourse relation identification. We first describe our way of creating labeling functions using an ensemble method. We next describe two different filtering mechanisms used to filtering out the output from each of the individual classifier created through the ensemble method. We then describe our alternative loss function for training a final model. Lastly, we show various different classifiers in different configurations we would like to incorporate in the previous two steps. In the next chapter, we are going to empirically evaluate these design choices.

Chapter 4

Experiments

Relation	Counts
Elaboration	7106
Attribution	2727
Joint	1775
Same-Unit	1277
Contrast	984
Explanation	876
Background	826
Cause	611
Enablement	522
Temporal	457
Total	17161

Relation	Counts
Evaluation	519
Comparison	280
Condition	274
Topic-Change	199
Manner-Means	192
Summary	191
Textual-Organization	148
Topic-Comment	132
Total	1935

Table 4.1: Top 10 Frequent relations (left) and the 8 infrequent relations (right), ordered by number of occurrences. The switch between Temporal and Evaluation is explained in the text.

In this chapter, we describe the experiments we conducted to evaluate our approach. We first describe the setups of our experiments, in Section 4.1, 4.2, 4.3. We then describe the first series of experiments we conducted for evaluating the performance of all our models over all aggregated metrics in Section 4.4. A selected amount of our models were also being evaluated with the per-relation performances. This per-relation analysis is shown in Section 4.5. At last, in order to find how did Data Programming actually help improve the performance of infrequent relation, we conducted an error analysis, which we describe in Section 4.6.

4.1 Datasets

We conducted our experiments on the Rhetorical Structure Theory Discourse Treebank (RST-DT) [9]. In this dataset, there are 18 coarse-grained discourse relations, as listed in Table 4.1. With respect to frequencies, (Jiang et al., 2016) [22] divided these relations into two groups. The top 10 *most frequent relations*, shown on the left side of Table 4.1, in total take up 90% of the training data. While the rest 8 *infrequent relations*, shown on the right side of Table 4.1, take up 10% of the training data. For consistency, we will follow their definition. One might note that Tem-

poral, a frequent relation, has a lower number of occurrences than an infrequent relation, Evaluation. This is because compared to [22] we use a slightly different definition of the number of occurrences. Our number of occurrences is generated based on the count on a right-heavy binarized version of the RST-DT tree with label attachment [32]. This makes more sense for our purpose since this number is exactly equal to the number of training samples used by all of the machine learning methods utilized in our approach. Although whether it is a good measure or not is subject to debate [32].

For the unsupervised dataset leveraged in training data expansion, we used the New York Times Annotated Corpus (NYT) [36], consisting of news documents, which is the same genre as RST-DT is.

Our task is to label discourse relations given discourse trees. Whereas RST-DT already has human annotated discourse trees for the documents, NYT does not. On NYT, therefore, we used the state of the art rhetorical structure classifier from [44] to create discourse trees. We used each node in a discourse tree as an unlabeled data instance for prediction. Note that we used only part of the NYT (the first 60,000 rhetorical tree nodes) for shorter computing time. However, adding more data made no significant difference to performance when we investigated this issue re-running our experiments with 90,000 tree nodes.

4.2 Evaluation Metrics

(Morey et al., 2017) [32] have recently shown that there are inconsistencies across the research community on how discourse parsing performance was evaluated on the RST-DT dataset. The main discrepancy comes with if the evaluation should be done using the micro-averaged F_1 or the macro-averaged F_1. Since the discourse parsing community has not reached an agreement on how evaluation should be done, and all metrics listed by Morey are not sensitive to poorly performed infrequent relations, we list the detail of the evaluation metrics we applied to our model below.

We define our notations first:

- $i \in D$: the index of a document in the testing dataset D
- $r \in \{0, 1, ..., 17\}$: the index of a specific relation

- h_{ir}: the number of correct predictions the discourse parser made on document i and relation r, and for "correct", we only require the discourse relation label but not the nuclearity prediction to be correct (this would be equivalent of the "R" metrics used by [32])
- p_{ir}: the number of predictions made by an discourse parser on document i and relation r
- g_{ir}: the number of gold relations in the evaluation set on document i and relation r.

Then, we define **micro-averaged** F_1 score (Micro) as:

$$\frac{2 \times \frac{\sum_i \sum_r h_{ir}}{\sum_i \sum_r g_{ir}} \times \frac{\sum_i \sum_r h_{ir}}{\sum_i \sum_r p_{ir}}}{\frac{\sum_i \sum_r h_{ir}}{\sum_i \sum_r g_{ir}} + \frac{\sum_i \sum_r h_{ir}}{\sum_i \sum_r p_{ir}}}$$

We also define **document macro-averaged** F_1 score (Doc Macro) as:

$$\text{average}_{i \in D}\left(\frac{2 \times \frac{\sum_r h_{ir}}{\sum_r g_{ir}} \times \frac{\sum_r h_{ir}}{\sum_r p_{ir}}}{\frac{\sum_r h_{ir}}{\sum_r g_{ir}} + \frac{\sum_r h_{ir}}{\sum_r p_{ir}}}\right)$$

The first definition, micro-averaged F_1 score, is the most standard and commonly used evaluation metrics in the past literature. The second definition, document macro-averaged F_1, which appeared in more recent publications [32], aims at measuring the average performance of the discourse parser on each document. Both of the two metrics, however, do not suffer if only the infrequent relations may perform extremely badly. In order to be more sensitive to the performance of infrequent relations, we need to evaluate the **per-relation** F_1 score for each of the relation, defined as:

$$\frac{2 \times \frac{\sum_i h_{ir}}{\sum_i g_{ir}} \times \frac{\sum_i h_{ir}}{\sum_i p_{ir}}}{\frac{\sum_i h_{ir}}{\sum_i g_{ir}} + \frac{\sum_i h_{ir}}{\sum_i p_{ir}}}$$

Based on the above, we further define another metrics, the **relation macro-averaged** F_1 score (Rel Macro), as:

$$\text{average}_{r \in \{0,1,\ldots,17\}}\left(\frac{2 \times \frac{\sum_i h_{ir}}{\sum_i g_{ir}} \times \frac{\sum_i h_{ir}}{\sum_i p_{ir}}}{\frac{\sum_i h_{ir}}{\sum_i g_{ir}} + \frac{\sum_i h_{ir}}{\sum_i p_{ir}}}\right)$$

These two scores are more sensitive to potentially extremely poorly performed relations.

To separately see the effects of our model on frequent relations and infrequent relations, we also define two variants of the relation macro-averaged F_1 score. The first variant only takes the average of the per-relation performance on the Top 10 frequent relations (Freq Macro). The second variant only takes the average of per-relation performance on the 8 infrequent relations (Inf Macro).

4.3 Extra Experimental Details

For features used by our models, we mostly based on features compiled by the state-of-the-art in discourse relation identification [44]. For neural network based models, the most common 600 features were extracted and combined with the word2vec representation of the first two words and the last two words on each discourse unit. For linear models, all features were used. We also adopt the state-of-the-art's approach of training one classifier for each of the 3 level 0(intra-sentential, sentential, paragraph) [44].

Data Programming was trained using the Snorkel implementation [3] which internally uses Stochastic Gradient Descent. All neural network based models were implemented using Tensorflow [1] and trained using Adam [24]. All linear models were implemented using Scikit-learn [34]. All hyperparameters were tuned on an eight fold cross-validation set before applied on to the test set unless otherwise specified.

For all our experiments, we forced the labeling function and final method to be the same type of model with the same hyperparameter. In this way, it is easier to claim if Data Programming actually brings improvement to performance. Otherwise one may argue that the performance improvement is gained by using a labeling function that works better than the final method.

All result below are reported as the average of 10 different runs with different random initialization points on the official test set of RST-DT using standard RST-Parseval method [31] [32].

Model	Micro	Doc Macro	Rel Macro	Freq Macro	Inf Macro
MV	58.1	61.6	34.9	41.0	27.4
DP	58.2	62.0	35.1	41.2	27.5
Increase with DP	0.17%	0.65%	0.57%	0.48%	0.36%

Table 4.2: Majority Voting v.s. Data Programming. All results are based on NN0 model

4.4 Overall Results

In this section we describe the overall results on all aggregated metrics defined in Section 4.2 across various configurations previously described in Chapter 3.

We first conducted two ablation tests on Data Programming. The first ablation test aims at evaluating if the Data Programming process actually helps over Majority Voting. The second ablation test aim at evaluating if the loss function we defined in Section 3.1.2 is better than the original loss function in Data Programming previously described in Section 2.2.2. The results are shown in Section 4.4.1.

Then, we experimented with using two different filtering mechanism previously described in Section 3.1.1. The results are shown in Section 4.4.2.

Next, we experimented with combining Data Programming and Prototypical Network, which was listed as **NN3** and **NN4** in Section 3.2. The results are shown in Section 4.4.3.

In the end, we experimented with combining Data Programming with linear models, which were listed as **SVM** and **LR** in Section 3.2. The results are shown in Section 4.4.4.

4.4.1 Ablation Test on Design Choice related to Data Programming

Our first task is to show if the design choice we made over using Data Programming actually brings us benefit. This involves determining if using the probabilistic distribution from Data Programming actually helps over just using a simple ensemble using majority vote. It also involves experimenting whether we should add the probabilistic distribution of labels that never shows up in our labeling functions in a multi-class environment.

Model	Micro	Doc Macro	Rel Macro	Freq Macro	Inf Macro
All	58.4	62.5	33.0	40.0	24.3
LF Only	58.2	62.0	35.1	41.2	27.5
Increase with LF only	-0.34%	-0.80%	6.36%	3.00%	13.17%

Table 4.3: Using all probability in Data Programming v.s. Only keep those that show up in Labeling Functions. All results are based on NN0 model

In Table 4.2 we show the performance of using the probabilistic distribution from Data Programming over using the Majority Voting mechanism. Unfortunately, using Data Programming is not significantly better than doing Majority Voting. However, it does appear that there is a small positive trend when Data Programming is applied.

In Table 4.3 we show the performance of using the probabilistic distribution from Data Programming in two different way. As described previously in Chapter 3, the original Data Programming approach would keep a probability over all possible labels. This is reasonable in their original binary task since the alternative of a wrong positive prediction is precisely negative. However in our task, as we are in a multi-class environment, the alternative of an incorrect prediction on Evaluation does not directly give us any information about which label it would be other than Evaluation. So using the probabilistic distribution here would involve too much noise. The adverse effect of the extra noise is also reflected in the result we have. While the version of NN2 with all the probabilities have higher performance on the metrics of both Micro F_1 score and Document Macro F_1 score, the Relation Macro F_1 score is consistently lower. This means that this extra amount of noise cause the infrequent relations to perform less well in the version of NN0 with all the probabilities. Since our primary task is to improve the infrequent relations, we prefer the variant that only keep the probabilities that show up in labeling functions.

Model	Micro	Doc Macro	Rel Macro	Freq Macro	Inf Macro
NN0	58.0	61.8	34.3	40.9	26.1
NN0 + DP	58.1	62.1	35.1	40.7	27.1
Increase with DP	0.17%	0.49%	2.33%	-0.49%	3.83%
NN1	56.9	60.6	31.4	39.1	21.8
NN1 + DP	57.5	61.2	32.6	40.0	23.4
Increase with DP	1.5%	0.99%	3.82%	2.30%	7.34%
NN2	58.5	62.4	31.4	39.3	21.5
NN2 + DP	57.9	61.7	27.3	37.3	15.2
Increase with DP	-1.03%	-1.12%	-13.06%	-5.09%	-29.30%

Table 4.4: Filtering using Uniform Boundary

4.4.2 Effects of Data Programming with Different Filtering Techniques

We then show the result of our model under scenarios using different types of filtering techniques, previously described in Section 3.1.1. This was done on several simple neural network models we proposed for discourse relation identification.

Table 4.4 shows the performance of using the uniform boundary technique. We can see here that we have performance improvement on both of NN0 and NN1 in terms of infrequent relations. Moreover, we are able to show a significant leap in Micro-averaged performance on NN1, a not well tuned neural network. This indicates that the Data Programming techniques with simple boundary would work exceptionally well if the neural network was too hard to tune right, and even if it is tuned right it may still provide some performance improvements. However, this is not the case for NN2, the model with dropout. Under a uniform boundary filtering, the performance was actually decreasing after Data Programming.

Table 4.5 shows the performance of using the dynamic boundary technique. Here, we get a somewhat different result. We are able to improve the performance of NN2 for infrequent relations with Data Programming by 4.19%. This is important because dropout is a standard technique in neural networks now, as dropout-

Model	Micro	Doc Macro	Rel Macro	Freq Macro	Inf Macro
NN0	58.0	61.8	34.3	40.9	26.1
NN0 + DP	58.2	62.0	35.1	41.2	27.5
Increase with DP	0.34%	0.32%	2.33%	0.73%	5.36%
NN1	56.9	60.6	31.4	39.1	21.8
NN1 + DP	57.5	61.2	32.5	40.0	23.1
Increase with DP	1.05%	0.99%	3.50%	2.30%	5.96%
NN2	58.5	62.4	31.4	39.3	21.5
NN2 + DP	58.4	62.3	32.1	39.9	22.4
Increase with DP	-0.17%	-0.16%	2.23%	1.53%	4.19%

Table 4.5: Filtering using Variable Boundary

enabled models usually have a higher performance. Indeed, given by the results we present, dropout itself does improve the Micro-average performance compared to the one without dropout. So ideally we hope our approach could boost a model that is generically more powerful. However, our Micro-averaged performance on the NN2 with Data Programming was still not as good as the one without Data Programming. Also, we observed that when dropout is enabled, the baseline performance of NN2 on infrequent relations is already much worse than NN0.

The results on a dropout enabled model partially coincide with the phenomenon discovered in [16], where they reported that when dropout is enabled, the probability output from the softmax layer started to drift from the true distribution of accuracy. This could be the reason why switching filtering techniques only have a substantial effect on the model with dropout.

4.4.3 Effects of Data Programming on Prototypical Networks

We also experimented with Prototypical Network on the problem of discourse relation identification. As previously described in Chapter 2, this model has been working well on some other tasks with only limited labeled data. We are hoping that this model could be a better labeling function compared to the standard neural

Model	Micro	Doc Macro	Rel Macro	Freq Macro	Inf Macro
NN0	58.0	61.8	34.3	40.9	26.1
NN0 + DP	58.2	62.0	35.1	41.2	27.5
Increase with DP	0.34%	0.32%	2.33%	0.73%	5.36%
NN3	58.6	62.3	34.8	41.3	26.6
NN3 + DP	58.5	62.3	35.4	41.6	27.6
Increase with DP	-0.17%	0.00%	1.72%	0.73%	3.76%

Table 4.6: Effects of Data Programming on Prototypical Networks without Dropout (NN3), NN0 is listed here as baseline

network model. The experiments were done for both the variant with or without Data Programming, following the conventions described in Section 4.3. Starting from here we are only experimenting with filtering using variable boundary, which is the most effective filtering method we found according to the previous section. The performance is shown in Table 4.6 and Table 4.7.

The regular Prototypical Network, denotes here the NN3, does show some benefit of predicting infrequent relations over NN0. However, if we apply Data Programming on NN3, while we are still able to get improvement on infrequent relations over the vanilla NN3, the performance we get is mostly the same as applying Data Programming on NN0. We also get a decrease in Micro-averaged performance, which is not seen in NN0.

On the other hand, when comparing the variants with dropout, which are denoted as NN4 and NN2, the Prototypical Network get much higher performance on infrequent relations. One may also observe that the performance gap between NN4 and NN3 are lower than the ones between NN2 and NN0. We are not sure why a Prototypical Network type of loss would reduce the problem of dropout on fitting over infrequent relations. The performance on infrequent relations was lower than NN3, nevertheless. When combining with Data Programming, we get some improvements, but the magnitude of improvements are smaller than the ones seen on NN2 and NN3.

In summary, we find the results of combining Prototypical Network with Data

Model	Micro	Doc Macro	Rel Macro	Freq Macro	Inf Macro
NN2	58.5	62.4	31.4	39.3	21.5
NN2 + DP	58.4	62.3	32.1	39.9	22.4
Increase with DP	-0.17%	-0.16%	2.23%	1.53%	4.19%
NN4	58.8	62.6	34.3	40.9	26.0
NN4 + DP	58.7	62.8	34.5	41.0	26.4
Increase with DP	-0.17%	0.32%	0.58%	-0.24%	1.54%

Table 4.7: Effects of Data Programming on Prototypical Networks with Dropout (NN4), NN2 is listed here as baseline

Programming to be less effective compared to the ones previously seen in Section 4.4.2. Although considering that Prototypical Network based models have a better performance than the regular neural network based models during the scenarios where Data Programming is not enabled, we can still conclude that we reached our original goal of creating better labeling functions using Few-Shot Learning techniques. They might not scale well when more data is added, however.

4.4.4 Effects of Data Programming on Linear Models

We also experimented with combining Linear models with Data Programming. Historically these models, especially SVM, have been providing excellent results in the field of discourse parsing.

The performance is shown in Table 4.8. We realize linear model hardly had any benefit from Data Programming, regardless of using a maximum margin loss like in SVM or using a probabilistic loss like in Logistic Regression. This is probably caused by the internal limitation of linear models.

4.5 Per Relation Performance Analysis

In order to show that our approach actually helps infrequent relations without hindering too much the performance of frequent relations, we conduct a detailed per-relation performance analysis. This is done for two types of models. One is NN2 +

Model	Micro	Doc Macro	Rel Macro	Freq Macro	Inf Macro
LR	57.8	62.0	30.6	38.1	21.2
LR + DP	57.6	61.8	29.5	37.9	19.0
Increase with DP	-0.35%	-0.32%	-3.59%	-0.52%	-10.38%
SVM	59.5	63.6	35.1	43.2	25.1
SVM + DP	59.5	63.6	35.1	43.4	24.9
Increase with DP	0.00%	0.00%	0.00%	0.46%	-0.80%

Table 4.8: Effects of Data Programming on Linear Models

DP using variable boundary filtering; another is NN4 + DP using variable boundary filtering. They are selected because they have the best Micro-averaged performance.

Table 4.9 shows the per-relation performance analysis for NN2, using the version with Data Programming against the one without Data Programming. We can see that most relation indeed have a positive increase in performance. And for those whose performance is lower after Data Programming, the decrease is mostly small except for Comparison. Based on this observation, we mostly reached the goal of improving infrequent relations without harming the performance of frequent relations.

Table 4.10 shows the per-relation performance analysis for NN4, using the version with Data Programming against the one without Data Programming. The improvement over using Data Programming we have here is much worse than the one we previously described in Table 4.9. Here, in fact, we have more infrequent relations that have their pre-relation performance decreased after Data Programming. Nevertheless, the decrease in the performance of the frequent ones is still limited.

We compare our performance results with the ones from (Jiang et al., 2016)[22], which is a previous work on improving infrequent relation using co-training (more detail described in Chapter 2). We can see that for both of the models we have, we are able to make sure that at least most of the frequent relations' performances are not being hindered too much, which is something that is not achievable from their

Relation	Count	F_1, before	F_1, + DP	Changes in %
Elaboration	7106	50.30	50.73	0.86
Attribution	2727	86.60	86.17	-0.50
Joint	1775	45.88	46.42	1.20
Same-Unit	1277	79.65	79.78	0.16
Contrast	984	30.87	30.66	-0.69
Explanation	876	16.03	16.58	3.45
Background	826	23.47	24.10	2.68
Cause	611	5.14	6.07	17.95
Enablement	522	52.56	52.83	0.52
Temporal	457	2.55	5.74	125.44
Evaluation	519	1.05	2.38	126.94
Comparison	280	11.83	8.44	-28.63
Condition	274	55.59	57.07	2.66
Topic-Change	199	1.64	2.51	52.57
Manner-Means	192	37.93	39.58	4.37
Summary	191	45.05	45.85	1.7
Textual-Organization	148	18.53	23.63	27.56
Topic-Comment	132	0.00	0.00	0.00

Table 4.9: Per-Relation Performance Analysis for NN2, before and after Data Programming. Relations that benefit from Data Programming are colored in green, those that are not are colored in Gold

result. However, Jiang use a different baseline method based on CODRA [23], so the result may be different if it was done with neural networks.

4.6 Error Analysis

In order to see how exactly does our approach affect the performance of infrequent relations, we did an error analysis on our model to see what actually causes our model to perform differently on infrequent relations. The results are shown with NN2 using variable boundary, which is the most successful model we have when Data Programming is applied. The confusion matrix for the one without Data Programming is shown in Table 4.11, and the one with Data Programming is shown in Table 4.12. In both tables, each row represents the actual label, and each column

Relation	Count	F_1, before	F_1, + DP	Changes in %
Elaboration	7106	51.71	51.71	0.00
Attribution	2727	86.52	86.55	0.03
Joint	1775	47.37	47.59	0.46
Same-Unit	1277	80.89	80.29	-0.74
Contrast	984	30.21	30.78	1.89
Explanation	876	15.58	16.87	8.28
Background	826	26.08	25.34	-2.84
Cause	611	7.56	8.40	11.11
Enablement	522	54.01	52.59	-2.63
Temporal	457	9.16	9.58	4.59
Evaluation	519	2.29	2.60	13.54
Comparison	280	17.44	15.94	-8.60
Condition	274	61.25	58.86	-3.90
Topic-Change	199	2.79	2.48	-11.11
Manner-Means	192	38.54	39.47	2.41
Summary	191	47.98	47.54	-0.92
Textual-Organization	148	35.89	35.71	-0.50
Topic-Comment	132	1.82	8.76	381.32

Table 4.10: Per-Relation Performance Analysis for NN4, before and after Data Programming. Relations that benefit from Data Programming are colored in green, those that are not are colored in Gold

represents the predicted label.

The two tables explain why the performance on infrequent relations is still low. It is mostly caused by the recall rates of infrequent relations. They are proportional to the number of predicted infrequent relations which are highlighted in green in Table 4.11 and Table 4.12. The recall rates of infrequent relations, in general, are much lower than the recall rates of frequent relations. The precisions of infrequent relations, while also not good, are still usually much higher than the recall rate and might be sometimes acceptable. Moreover, one would observe that most of the actual infrequent relations are being predicted as Elaboration, the most common relation in RST-DT. All predictions that are made to Elaboration are highlighted in pink in Table 4.11 and Table 4.12. This happens because the machine learning model could easily learn the Elaboration is the most common label and have a

	AT	BA	CA	CMP	CND	CO	EL	EN	EV	EX	JO	M-M	S-U	SR	TE	T-O	T-C	T-CM
AT	308	1	0	0	0	0	8	0	0	0	0	0	1	0	0	0	0	0
BA	7	21	1	0	2	3	20	0	0	1	7	1	5	0	1	0	0	0
CA	0	1	2	0	0	3	29	2	0	6	16	1	1	0	0	0	0	0
CMP	0	2	0	2	0	4	17	0	0	1	0	0	0	0	0	0	0	0
CND	3	5	0	0	22	2	4	0	0	0	2	0	1	0	0	0	0	0
CO	2	2	0	0	2	51	24	0	0	0	12	0	3	0	0	0	0	0
EL	7	2	1	0	0	2	511	14	0	4	31	0	4	0	1	0	4	0
EN	2	0	0	0	0	0	11	25	0	0	0	0	1	0	0	0	0	0
EV	2	1	0	0	0	2	31	0	0	1	8	0	0	0	0	0	1	0
EX	0	0	2	0	0	2	41	1	0	13	6	0	0	0	0	0	1	0
JO	1	1	2	0	0	5	77	0	0	4	203	0	8	0	0	0	0	0
M-M	1	2	0	0	0	1	11	0	0	0	0	7	0	0	0	0	0	0
S-U	4	0	0	0	0	1	4	0	0	0	7	0	205	0	0	0	0	0
SR	0	0	0	0	0	0	13	0	0	0	1	0	0	9	0	0	0	0
TE	0	19	0	0	1	2	20	1	0	0	28	0	4	0	2	0	0	0
T-O	0	0	0	0	0	0	4	0	0	0	3	0	0	0	0	2	0	0
T-C	0	0	0	0	0	0	3	0	0	0	3	0	0	0	0	0	0	0
T-CM	0	0	0	0	0	0	10	0	0	0	9	0	0	0	0	0	0	0

Table 4.11: Error Analysis, NN2 without Data Programming. Each row represents the actual label and each column represents the predicted label. The abbreviations of the relations are Topic-Change (T-C), Topic-Comment (T-CM), TextualOrganization (T-O), Manner-Means (M-M), Comparison (CMP), Evaluation (EV), Summary (SR), Condition (CND), Enablement (EN), Cause (CA), Temporal (TE), Explanation (EX), Background (BA), Contrast (CO), Joint (JO), SameUnit (S-U), Attribution (AT), and Elaboration (EL), followed from the conventions used by [23]

stronger bias towards predicting everything it does not know as Elaboration.

The Data Programming essentially helps in tackling the issue above. As we could see, most of the improvements come from a higher recall rate, where the values from the green boxes in Table 4.12 are consistently either higher or stay the same compared to the green boxes in Table 4.11. The number of infrequent relations that are being predicted as Elaboration is also reduced, where the values in the pink column of Table 4.12 are usually lower than the ones in table 4.11. This does provide an adverse effect on Elaboration's recall. However, the improvement in its precision has a larger effect. This contributes to the slight improvement in the F_1 score of Elaboration we saw in the previous section.

	AT	BA	CA	CMP	CND	CO	EL	EN	EV	EX	JO	M-M	S-U	SR	TE	T-O	T-C	T-CM
AT	308	1	0	0	0	0	8	0	0	0	0	0	1	0	0	0	0	0
BA	7	22	1	0	2	3	18	0	0	1	7	0	6	0	1	0	0	0
CA	0	1	4	0	0	3	25	2	0	7	16	1	1	0	0	0	0	0
CMP	0	4	0	2	0	3	16	0	0	1	0	0	0	0	1	0	0	0
CND	3	4	0	0	23	2	3	0	0	0	2	0	1	0	0	0	0	0
CO	2	2	0	0	2	53	24	0	0	0	11	0	2	0	0	0	0	0
EL	7	3	2	0	0	2	502	17	2	8	29	0	4	0	1	0	5	0
EN	2	0	0	0	0	0	10	26	0	0	0	0	1	0	0	0	0	0
EV	3	0	0	0	0	1	28	0	1	3	8	0	0	0	0	0	1	0
EX	0	1	6	0	0	2	35	1	1	15	6	0	0	0	0	0	1	0
JO	2	2	3	0	0	6	71	0	0	8	203	0	7	0	0	0	0	0
M-M	1	2	0	0	0	1	10	0	0	1	0	7	0	0	0	0	0	0
S-U	4	0	1	0	0	1	4	0	0	0	7	0	203	0	0	0	0	0
SR	0	0	0	0	0	0	13	0	0	0	1	0	0	10	0	0	0	0
TE	0	19	1	0	0	1	19	1	0	0	29	0	4	0	4	0	0	0
T-O	0	0	0	0	0	0	4	0	0	0	2	0	0	0	0	3	1	0
T-C	0	0	0	0	0	0	3	0	0	0	2	0	0	0	0	0	0	0
T-CM	0	0	0	0	0	0	9	0	0	0	9	0	0	0	0	0	0	0

Table 4.12: Error Analysis, NN2 with Data Programming. Each row represents the actual label and each column represents the predicted label. The abbreviations of the relations are Topic-Change (T-C), Topic-Comment (T-CM), TextualOrganization (T-O), Manner-Means (M-M), Comparison (CMP), Evaluation (EV), Summary (SR), Condition (CND), Enablement (EN), Cause (CA), Temporal (TE), Explanation (EX), Background (BA), Contrast (CO), Joint (JO), SameUnit (S-U), Attribution (AT), and Elaboration (EL), followed from the conventions used by [23]

Chapter 5

Conclusion and Future Directions

In our thesis, we show an alternative approach to improve the performance of discourse parser on infrequent discourse relations. We present a workflow for applying Data Programming framework to discourse relation identification task, and use this workflow to add more training data for a discourse relation identification model. When applying a new variable boundary filtering mechanism proposed by us, our empirical results show improvement on the performance of infrequent relations across different configurations of neural networks, with only limited negative effect on the performance of frequent relations.

There is still plenty of room for improvement in our approach. First, since the variable boundary strategy had a strong effect on the performance, meta-learning strategies that can learn boundaries automatically might bring higher performance improvement on our task. Such strategies may especially provide a better path to integrate Data Programming with few-shot learning mechanism as meta-learning has been shown to be helpful for Prototypical Network in semi-supervised learning settings [42]. Additionally, we are also considering using a deep learning model as the final model. This is because the benefit of more data would be more evident in a deep learning model, compared to the simple feed-forward neural network we used. Besides, in our current approach, the performance improvement between Data Programming and Majority Voting is negligible. We think that might be caused by the fact that all our labeling functions are created using the same method, whereas Data Programming's strength really comes from tacking scenarios in which labeling functions are entirely different. Under this assumption, alternative ideas from ensemble methods like Adaboost [17] where each classifier is created for different purposes are worth experimenting. Moreover, we only incorporated one type of few-shot learning mechanism into our approach. Since there are some other few-shot learning models available, they might bring a better result.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. → page 30

[2] K. Allen, G. Carenini, and R. Ng. Detecting disagreement in conversations using pseudo-monologic rhetorical structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1169–1180, 2014. → pages iii, 2

[3] A. R. S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3), 2017. → pages 15, 30

[4] S. H. Bach, B. He, A. Ratner, and C. Ré. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning*, pages 273–282, 2017. → page 6

[5] P. Bhatia, Y. Ji, and J. Eisenstein. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218, 2015. → pages iii, 2, 3

[6] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998. → pages 4, 12

[7] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. → pages 7, 20

[8] L. Carlson and D. Marcu. Discourse tagging reference manual. *ISI Technical Report*, page 56, 2001. → pages x, 2

[9] L. Carlson, D. Marcu, and M. E. Okurowski. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer, 2003. → page 27

[10] J. Cocke. Programming languages and their compilers: Preliminary notes. 1970. → page 10

[11] V. W. Feng and G. Hirst. A linear-time bottom-up discourse parser with constraints and post-editing. In *ACL (1)*, pages 511–521, 2014. → pages 10, 11

[12] R. Fisher and R. Simmons. Spectral semi-supervised discourse relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 89–93, 2015. → pages 7, 12, 22

[13] V. Garcia and J. Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018. → page 16

[14] S. Gerani, Y. Mehdad, G. Carenini, R. T. Ng, and B. Nejat. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1602–1613, 2014. → pages iii, 2, 3

[15] S. Gerani, G. Carenini, and R. T. Ng. Modeling content and structure for abstractive review summarization. *Computer Speech & Language*, 2016. → pages iii, 2, 3

[16] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017. → page 34

[17] T. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009. → page 43

[18] M. Hearst. Noun homograph disambiguation using local context in large text corpora. → page 12

[19] H. Hernault, H. Prendinger, M. Ishizuka, et al. Hilda: A discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3), 2010. → page 10

[20] Y. Ji and J. Eisenstein. Representation learning for text-level discourse parsing. In *ACL (1)*, pages 13–24, 2014. → pages 3, 4, 5, 10, 13

[21] Y. Ji and N. A. Smith. Neural discourse structure for text categorization. In *ACL (1)*, volume 1, pages 996–1005, 2017. → pages iii, 2, 3

[22] K. Jiang, G. Carenini, and R. Ng. Training data enrichment for infrequent discourse relations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2603–2614, 2016. → pages 3, 4, 7, 13, 21, 27, 28, 37

[23] S. Joty, G. Carenini, and R. T. Ng. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435, 2015. → pages ix, 3, 4, 5, 10, 13, 38, 40, 41

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014. → page 30

[25] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015. → page 16

[26] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011. → page 16

[27] J. Li, R. Li, and E. Hovy. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 2061–2069, 2014. → page 11

[28] Q. Li, T. Li, and B. Chang. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 362–371, 2016. → page 11

[29] L. Liu, X. Ren, Q. Zhu, S. Zhi, H. Gui, H. Ji, and J. Han. Heterogeneous supervision for relation extraction: A representation learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 46–56, 2017. → page 15

[30] Y. Liu and M. Lapata. Learning contextually informed representations for linear-time discourse parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1289–1298, 2017. → page 11

[31] D. Marcu. The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational linguistics*, 26(3):395–448, 2000. → pages 19, 30

[32] M. Morey, P. Muller, and N. Asher. How much progress have we made on rst discourse parsing? a replication study of recent results on the rst-dt. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1319–1324, 2017. → pages iii, 28, 29, 30

[33] B. Nejat, G. Carenini, and R. Ng. Exploring joint neural model for sentence level discourse parsing and sentiment analysis. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 289–298, 2017. → pages iii, 2, 3

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct): 2825–2830, 2011. → page 30

[35] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575, 2016. → pages iii, x, 6, 13, 19, 20, 22

[36] E. Sandhaus. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752, 2008. → page 28

[37] M. C. Seiler and F. A. Seiler. Numerical recipes in c: the art of scientific computing. *Risk Analysis*, 9(3):415–416, 1989. → page 3

[38] R. Shams. Semi-supervised classification for natural language processing. *arXiv preprint arXiv:1409.7612*, 2014. → pages 7, 12, 21

[39] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090, 2017. → pages 7, 16, 17, 24

[40] R. Soricut and D. Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics, 2003. → page 10

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. → page 24

[42] E. Triantafillou, H. Larochelle, J. Snell, J. Tenenbaum, K. J. Swersky, M. Ren, R. Zemel, and S. Ravi. Meta-learning for semi-supervised few-shot classification. 2018. → page 43

[43] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016. → page 16

[44] Y. Wang, S. Li, and H. Wang. A two-stage parsing method for text-level discourse analysis. In *ACL (2)*, pages 184–188, 2017. → pages iii, viii, x, 3, 4, 5, 10, 24, 28, 30

[45] X. Zeng, G. Carenini, R. Ng, and H. Jang. Infrequent discourse relation identification using data programming. In *Pacific Northwest Regional NLP Workshop*, 2018. → page v