Improving Digital Ink Interpretation through Expected Type Prediction and Dynamic Dispatch

by

Kah Seng Tay

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2008

Author ____
Department of Electrical Engineering and Computer Science
May 8, 2008

Certified by ____
Kimberle Koile, Ph.D.
Research Scientist, MIT CSAIL
Thesis Supervisor

Accepted by ____
Arthur C. Smith, Ph.D.
Professor of Electrical Engineering
Chairman, Department Committee on Graduate Theses

Improving Digital Ink Interpretation through Expected Type Prediction and Dynamic Dispatch

by

Kah Seng Tay

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Interpretation accuracy of current applications dependent on interpretation of handwritten "digital ink" can be improved by providing contextual information about an ink sample's expected type. This expected type, however, has to be known or provided *a priori*, and poses several challenges if unknown or ambiguous. We have developed a novel approach that uses a classic machine learning technique to predict this expected type from an ink sample. By extracting many relevant features from the ink, and performing generic dimensionality reduction, we can obtain a minimum prediction accuracy of 89% for experiments involving up to five different expected types. With this approach, we can create a "dynamic dispatch interpreter" by biasing interpretation differently according to the predicted expected types of the ink samples. When evaluated in the domain of introductory computer science, our interpreter achieves high interpretation accuracy (87%), an improvement from Microsoft's default interpreter (62%), and comparable with other previous interpreters (87-89%), which, unlike ours, require additional expected type information for each ink sample.

Thesis Supervisor: Kimberle Koile, Ph.D.
Title: Research Scientist

Acknowledgements

I would like to thank my thesis advisor, Kimberle Koile, without whom this research would not have been possible. She has provided me funding over the years and has guided me in tablet PCs and ink interpretation. She is willing to listen to my overly wild and ambitious ideas, and encourages me to continuously seek improvement. She has been extremely helpful with editing and improving my papers and this thesis, helping me articulate my ideas clearly and succinctly.

I would like to thank professors Martin Rinard, Sivan Toledo, Michael Ernst, and Saman Amarasinghe. They were the lecturers with whom I have had the privilege to teach the last two terms of 6.170 *Laboratory in Software Engineering* offered at MIT. 6.170 was my favorite class taken at MIT, and it was an honor to be a TA (and subsequently head TA) for the class. I have learned a lot from them through this experience, and am grateful for the opportunity and leadership.

I would like to thank the group members of Classroom Learning Partner: Adam Rogal, for starting out in the group with me from the very beginning and always being a helping hand; Capen Low, David Chen, and Curtis Liu, without whom I could not have done much many improvements to the new version of CLP. I would also like to thank some past members of the group: Michel Rbeiz, for first introducing me to the group, and whose work I have continued; and Sanjukta Pal, Kevin Chevalier and Kenneth Wu, for the days and nights spent in the lab together developing and debugging CLP.

I would like to thank Sung Kim from the Program Analysis Group and Tom Ouyang from the Sketch Understanding Group. Sung, whom I worked with briefly on automatic bug detection, introduced me to machine learning and feature extraction, allowing me to come up with original idea for the work in my thesis. Tom created the chemical diagram interpreter, and is collaborating with my group for joint deployment.

I would like to thank the many members of Asian Baptist Student Koinonia, my Christian fellowship group, for being here with me at MIT these four years. Special thanks go to members of my class: Brandon Yoshimoto (also my wonderful roommate), Jill Rowehl, Tiffany Lee, Tami Shinkawa, Sophia Lee and Diana Wang, for weathering thick and thin with me year after year over Christian Festival, the bible studies and the IM games; and to staff members Austin Kim, Donald Choi and David Um, for their guidance, support, food and welcoming me to their homes.

I would like to thank my parents, Ming Chee Tay and Lai Ngoh Lam, and family in Singapore, for giving me the opportunities to learn and excel in school, and allowing me to go overseas for my studies without a scholarship. I would also like to thank my twin brother, Kah Keng, for the friendly competition and the many collaborative projects and ventures we have done together.

I would like to thank the Siebel Scholars Foundation, Microsoft iCampus, the Office of Educational Innovation and Technology for their financial support in my graduate education and research.

I would especially like to thank Serene Lee, my fiancée (and wife, in a month), for accepting me for what I am and enduring the time I have spent at work and on research. Thank you for the support, love and encouragement throughout these few years at MIT.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Ink interpretation systems play a critical role in enabling more “intelligent” computers that are capable of understanding what a user has written, beyond mere digital dots on a plane. Such interpretation systems need to be highly accurate [Giudice & Mottershead, 1999], [LaLomia, 1994] in parsing a variety of handwritten text and diagrams into a digitized semantic representation in order to be useful for higher-order processing by other applications. Digital ink interpretation has grown increasingly important as tablet PCs become more pervasive in today’s society, especially in classrooms. Tablet PCs offer users the ability to transcribe notes digitally in the users' own handwriting, using a stylus and screen as easily and naturally as pen and paper.

This thesis reports a new method that uses ink type prediction and dynamic dispatch as the basis for an ink interpretation system capable of high ink interpretation accuracy over multiple domains. Our novel approach uses machine learning techniques to extract features from ink strokes to predict the type of the ink, thus identifying its domain, then dispatches interpretation to well-suited domain-specialized interpreters based on the particular type. This approach is able to achieve higher overall interpretation accuracy than existing systems, and allows scaling of our interpretation system, something currently not possible with domain-specialized interpreters.

1.1 Motivation

There are many domain-specialized interpreters that are capable of producing highly accurate interpretations, but only of ink samples within their own domains. These domain-specialized interpreters are developed concurrently by many researchers and are difficult to integrate into systems that could benefit from using them. Ink interpretation systems are thus often plagued with problems of poor accuracy because they are limited in scope or cannot accurately identify the best interpreter to choose from a set of interpreters. Our goal, which resulted in the work described in this thesis, was to deploy an ink interpretation system capable of high interpretation accuracy over several domains. The scenario is this one: We have a digital ink sample that belongs to a particular domain, e.g., Scheme expressions, but we do not know, or want to have to specify *a priori*, which of the interpreters in our system should be used to interpret the ink. Some approaches choose upfront the interpreter to use, with information provided externally by a user, for example. Others choose the best interpreter based on the highest ranked confidence measure. Our novel approach uses machine learning, on ink stroke features of various possible ink types, to predict the correct interpreter for a particular ink sample, before dispatching interpretation calls to that interpreter.

1.2 Overview

We have created a common `Interpreter` framework to support a variety of interpreters for different domains. To evaluate our novel idea, we create an ink type prediction module that uses machine learning to differentiate between different ink answer types and to predict the most suitable type based on extracted features from the ink. We then build upon the `Interpreter` framework by creating dynamic dispatch interpreters that utilize information from ink type prediction to improve interpretation accuracy. This entire interpretation system is writer-independent, and operates synchronously on a completed ink sample, making full use of the rich dynamic features found in digital ink.

We tested our prototype in an application developed by our group, which depends on highly accurate ink interpretation. The application, called Classroom Learning Partner (CLP), consists of a network of tablet PCs that run software for posing in-class questions to students, interpreting their handwritten answers, and aggregating the answers into equivalence classes. We have shown that such systems hold great promise for improving student interaction and learning in classrooms [Koile & Singer, 2006], [Koile et al, 2007a], [Koile et al, 2007b]. For ink interpretation systems to be used in the classroom, however, high ink interpretation accuracy rates are necessary for instructor and student confidence in the system. A limitation of the original Microsoft interpreter, used in our first prototype of CLP, was its inability to accurately interpret ink samples beyond the domain for which it was trained—cursive English text. Early work on CLP [Rbeiz, 2006] improved interpretation accuracy for the domain of introductory computer science by introducing instructor-specified expected types for answers to questions; different interpretation methods were used for each type. This improvement, however, was not easily scalable to include more domain-specialized interpretation, e.g., chemical diagrams.

Using CLP as our test environment, we conducted experiments in which students were instructed to write on the tablet PCs as they normally would write on paper, without needing to follow any special gesture-based recognition schemes such as Graffiti for the original Palm Pilot [Rubine, 1991]. Such gesture-based schemes have a high learning curve which we believe would affect a student's ability to write as he or she normally would, impeding regular writing and note-taking. We required no individualized handwriting training in our experiments, as the nature of coursework presents very little time for students to train handwriting recognition systems to learn individual handwriting. Students may choose to drop the class, wasting early effort, or the instructor may come up with new material after training is done. No real-time feedback of the interpretation result was provided, allowing students to write freely without becoming distracted by worrying about inaccurate interpretation. With sufficiently high ink interpretation rates, a few interpretation errors can be tolerated by the instructor, who is the only one able to view these errors.

The hypothesis investigated in this thesis is the following: Ink interpretation accuracy of an interpreter that dynamically dispatches to a specialized interpreter based on a predicted ink sample type will be close in accuracy to an interpreter that requires *a priori* expected type information. This hypothesis is illustrated visually in Figure 1-1. In addition, we expect our proposed ink interpretation method to alleviate limitations of our current interpreter that depends on *a priori* type information, namely, low accuracy when expected types are unknown, or when ink samples representing student answers are incorrect and of an unexpected type.

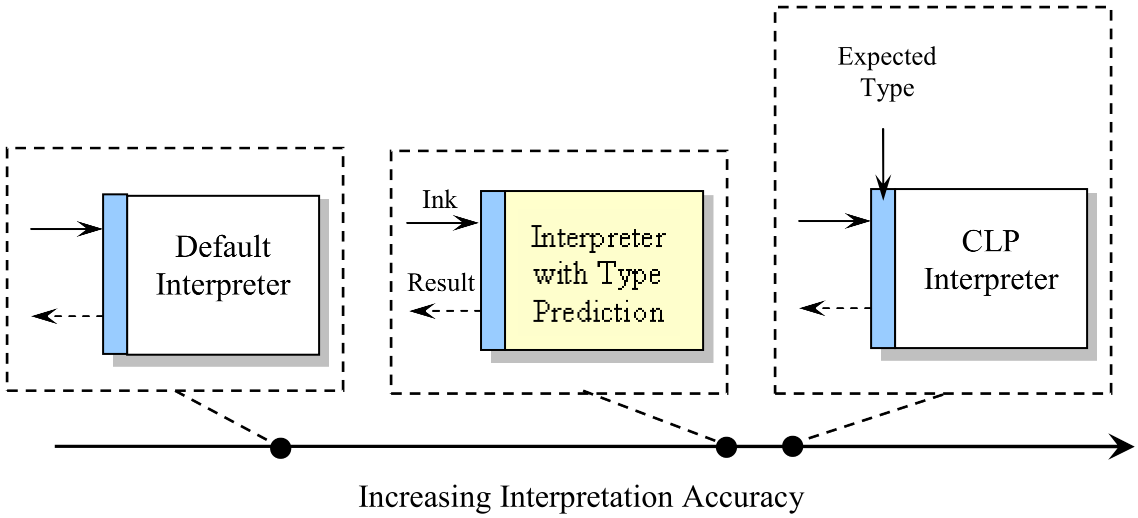


Figure 1-1. Our hypothesis: We expect an interpreter that predicts expected ink sample type and dispatches to appropriate specialized interpreters to be close in accuracy to an interpreter with user-supplied *a priori* knowledge of expected type. This new interpreter also will be far more accurate than a default interpreter that uses no ink sample type information.

1.3 Thesis Outline

We describe background on domain-specialized interpreters and biasing with expected types in Chapter 2. Chapter 3 describes our experimental approach and implementation. We go into details and results of ink type prediction in Chapter 4, and dynamic dispatch interpretation in Chapter 5. Chapter 6 describes related work in the field of ink interpretation. Finally, Chapter 7 summarizes our main contributions and describes future work beyond the scope of this thesis.

Chapter 2
Background

In this chapter we describe relevant background on handwriting recognition so that our work can be placed in the context of current and past research. We discuss example domain-specialized interpreters and how biasing interpreters improves interpretation accuracy. Related work and alternative approaches to handwriting recognition are discussed in Chapter 6.

2.1 Domain-Specialized Interpreters

There has been much recent interest in advanced sketch interpretation systems. Many of these systems have demonstrated that domain knowledge can be used to overcome ambiguities and hence improve interpretation accuracy (e.g., [Sezgin & Davis, 2005], [Calhoun et al, 2002], [Shilman et al, 2002, 2004], [Gennari et al, 2005], [Kara & Stahovich, 2004]).

Research on domain-specialized interpreters for CLP has been conducted, and these interpreters can recognize a variety of ink types with varying degrees of success: boolean, numbers, sequences, Scheme expressions, box-and-pointer diagrams, and diagram markings. [Rbeiz, 2006] [Chevalier, 2007] [Wu, 2008] [Koile et al, 2007b] Figures 2-1 (a) and (b) show, respectively, an example of a box-and-pointer diagram and its CLP interpretation.

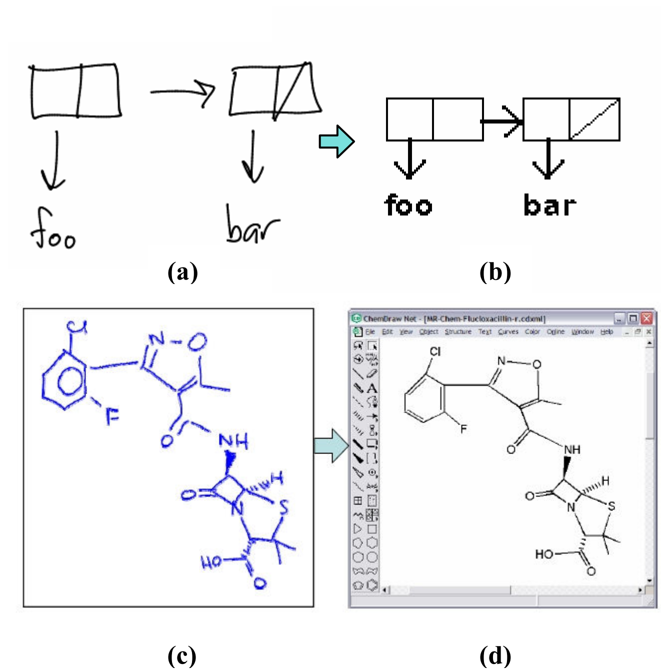
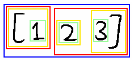
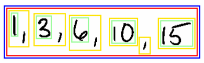
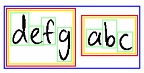
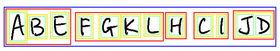


Figure 2-1. (a) Hand-drawn box-and-pointer diagram, **(b)** CLP's interpretation [Chevalier, 2007] **(c)** Hand-drawn chemical structure, **(d)** Interpretation re-rendered [Ouyang & Davis, 2007]

A prototype chemical structure interpretation system also has been developed by T. Ouyang and Prof. R. Davis of the Sketch Understanding Group at MIT [Ouyang & Davis, 2007]; it is capable of interpreting hand-drawn diagrams of organic chemistry compounds, using the graphical vocabulary and drawing conventions routinely employed by chemists. Figures 2-1 (c) and (d) show a chemical structure and its rendered interpretation in that system.

With a restricted domain, researchers can make assumptions about the possible ink inputs and obtain higher interpretation accuracy as a result. Table 2.1, for example, shows how we improved sequence interpretation for CLP over several iterations of the ink segmentation and interpretation algorithm, which we call INK. The latest version of our sequence interpreter uses a mixture of *sequence subtypes* (number, single character or string), and several *flag*s (e.g., whether commas, brackets, or ampersands are present) as heuristics for interpreting the ink more accurately than ordinary English interpreters. This higher accuracy, however, is conditioned on obtaining *a priori* information about the expected domain (or equivalently, expected type and expected flags) of the ink input.

Table 2.1: Interpretation results for four ink samples of sequences and overall accuracies

Ink	INKv2.2[1]		INKv1.5[2]		INKv1[3]		Microsoft	
	Interpreted	**%**	**Interpreted**	**%**	**Interpreted**	**%**	**Interpreted**	**%**
	[1,2,3]	100.00	TI,2,3]	71.43	->,23]	57.14	[I,23]	71.43
	[1,3,6,10,15]	100.00	[1,3,6,10I15]	92.31	[li3,6,10,15]	84.62	[1,3,6,10115]	92.31
	[d,e,f,g,a,b,C]	100.00	[defy,abc]	60.00	[defy,abc]	60.00	[defog,abc]	66.67
	[A,B,E,F,G,k, L,H,C,I,J,D]	100.00	[A,B,E,F,G,k, L,H,C,I,JD]	96.00	[ABE,F,Gk,H, ->,JD]	64.00	[ABE,Fatal,H, CI,JD]	64.00
All Sequence Accuracy		**89.33**		**73.48**		**79.58**		**70.92**

2.2 Biasing With Expected Type Information

Recognition systems on handwritten mailing addresses have specific templates and restricted dictionaries to interpret state abbreviations and zip codes more accurately [Plamondon & Srihari, 2000]. The form-design tool of Scribble [O' Boyle et al, 2000] allows a known field within a form template to be annotated with markup indicating the field input type from a range of possibilities such as dates, emails, credit card numbers, etc. This approach improves accuracy during interpretation of the ink on the form.

As mentioned in our introduction, CLP also uses *expected type*s to bias interpretation of the ink for better accuracy [Rbeiz, 2006]. When the instructor knows that the students' answers should be of a particular type, a number, for example, an expected type is defined for that exercise question using an authoring tool [Chen, 2006] that we developed for use in preparing class presentation material. During class, all student ink sample inputs for that exercise, in turn, are annotated with that expected type. Each ink input sample is then dispatched to the best domain-specialized interpreter for the expected type, and the interpretation results are passed on to the next component (CLP's aggregator) [Smith, 2006].

[1] INKv2.2 is this author's work as published in [Koile et al, 2007b].
[2] INKv1.5 is a result of Rbeiz's unpublished research in 2006 after his thesis.
[3] INKv1 is Rbeiz's interpreter as published in [Rbeiz, 2006].

We illustrate this technique with a simple example—applying biasing to numerical strings that are easily misinterpreted as characters of the Roman alphabet (e.g., the ink strokes that a user writes for “11” may be interpreted as two lowercase-Ls of the alphabet). When we performed the experiments with this example, an accuracy of 99% was obtained compared to 89% without biasing (see breakdown in Table 2.2). Rbeiz’s earlier study of 21 representative examples of student answers across 5 expected types also showed that interpretation with this biasing approach achieved a higher accuracy (87% compared to 73%).

Table 2.2: Interpretation accuracy results showing improvement by number biasing

Number	Possibly Confused As	Number Biasing (%)	No Biasing (%)
0	O	100.00	53.85
1	I or l	100.00	36.36
2	Z	100.00	100.00
5	S	100.00	100.00
6	G	100.00	100.00
7	T or >	100.00	100.00
9	g	100.00	90.91
10	IO or lo	100.00	100.00
11	II or ll	95.45	95.45
50	so	90.91	81.82
55	SS	100.00	100.00
100	loo	100.00	100.00
101	IOI or lol	96.67	96.67
Total Accuracy		**98.70**	**88.86**

The use of expected types can be extended beyond the interpretation of regular English strings. With expected types, CLP can differentiate the possibilities of domain-specialized ink inputs from students: whether they are box-and-pointer diagrams, Scheme expressions, markings, and in future, chemical structures or circuit diagrams.

Thus, we have shown in this previous work of ours that biasing an ink interpreter with information about expected types improves interpretation accuracy. Our next challenge, addressed in this thesis, was to extend this idea to decrease dependency on explicit *a priori* labeling of expected type information.

Chapter 3
Approach

In this chapter, we describe the design of an interpretation system that *automatically* takes advantage of the idea that biasing ink samples with type information improves interpretation accuracy. The interpretation system employs machine learning techniques to predict the ink sample type, and then dispatches interpretation calls to an appropriate ink interpreter specialized for that type. The system is writer-independent and operates synchronously on a completed ink sample, a method that has proven advantageous for our classroom application [Rbeiz, 2006]. Unlike scanned handwritten images or optical character recognition (OCR), we make full use of the dynamic nature of digital ink for our interpretation system. Our interpretation framework is designed for online digital ink interpretation, and allows different interpreters to be added with relative ease. This chapter describes this framework and presents a high-level overview of our ink type prediction using machine learning and our dynamic dispatch method. Our system has been integrated with CLP, allowing us to easily deploy this approach in the classroom. We describe an evaluation of our idea using ink samples collected in a user study.

3.1 Dynamic Ink Strokes

The dynamic nature of ink strokes plays an important role in our work. Digital ink samples captured through pen-based input, e.g., using a tablet PC, contain a myriad of information not present in static scanned images of user handwriting. Examples of such information are the number of strokes written or drawn, the individual stroke order over the entire ink sample, and the positions of sampled points in each stroke. This information can aid recognition, e.g., overlapping strokes of different characters that may have been grouped inaccurately when rasterized in a scanned image can be easily identified as disjoint using stroke information. The information, unfortunately, also can mislead interpreters, e.g., two different user-written samples may look the same visually, but may have been written in different stroke orders. Machine learning with feature selection, however, as described in Chapter 4, allows us to use dynamic stroke information effectively. In this thesis, we focus on improving the interpretation accuracy of digital ink, for which this information can be captured with tablet PCs.

3.2 The Interpretation Framework

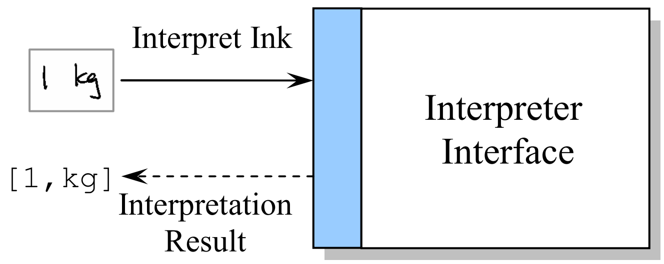


Figure 3-1. The common interpreter interface that we use within CLP and for our experiments.

We have created a common `Interpreter` interface, where "common" refers to the ability to "plug in" various interpreters for use in our CLP prototyping environment. Figure 3-1 depicts a simple diagram of this `Interpreter` interface. With this framework, we allow the interpretation module of CLP originally created by Rbeiz to be extended easily as we develop newer interpreters. We also have as a goal, the ability to plug in interpreters developed by researchers working in other domains.

Examples of deployed interpreters that have taken advantage of our framework are the box-and-pointer diagram interpreter [Chevalier, 2007], a marking interpreter [Wu, 2008], our specialized sequence interpreters and post-2006 versions of our CLP general interpreters. Using this same `Interpreter` interface, we also have been able to run experiments comparing the accuracies of newer versions of the same interpreters and the accuracies of different algorithms. Details of how our new ink interpreter fits into this general interpretation framework are discussed in Chapters 4 and 5.

3.3 Representative Examples

For this thesis, we selected a total of 181 different representative examples of possible student answers. Some of the examples are based on actual tutorial answers from past recitations at MIT, while the others are chosen because they are highly representative of the domain and the answer types we have seen in the classroom. Eighty-eight of these examples lie within the domain of introductory computer science (including the 21 from Rbeiz's thesis) and 93 within introductory chemistry, since these are the two domains in which CLP is being used. Figure 3-2 shows several of these representative examples and their types. We list our full set of representative examples in Appendix A.

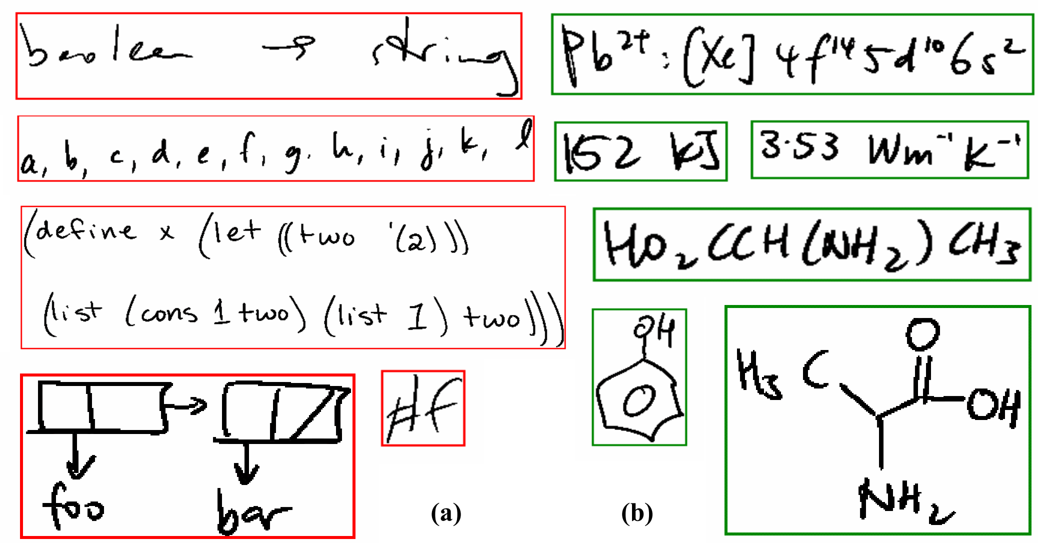


Figure 3-2. Representative examples selected from the field of **(a)** introductory computer science; **(b)** introductory chemistry, for training and evaluating our interpretation system.

3.4 Improving Ink Interpretation Accuracy

As stated earlier, the main idea explored in this thesis is that of using ink type prediction and the dynamic dispatch to specialized interpreters to improve ink interpretation accuracy. A problem faced by most ink interpretation systems is that many domain-specialized interpreters exist, and the systems cannot identify the best interpreter for interpreting specific samples of ink. Many interpretation systems address this issue by relying on confidence measures, which rank output results from candidate interpreters, often qualitatively. Our novel approach differs significantly from these confidence-based systems: Instead of performing potentially costly recognition procedures on many different domain-specialized interpreters to determine the confidence of the interpreted result, we predict the correct interpreter to which to dispatch the ink sample.

Our approach is similar to having an instructor provide *a priori* information about the interpreter to be chosen based on a given *expected type*, except that we use machine learning to predict this expected type purely from the ink sample and a list of available interpreters and their associated ink sample types. In the following two chapters, we describe in detail the two components to our approach: ink type prediction and using dynamic dispatch. Below we give a justification and preview for each of these components.

- **Ink Type Prediction.** Type prediction has two important benefits: (1) it avoids the inefficiency of having to choose a candidate interpreter by running all possible interpreters and ranking their outputs, and (2) it does not require *a priori* specification of an expected answer type for each ink sample. We accomplish type prediction by using machine learning classification techniques, described in Chapter 4: Our machine learning algorithms select relevant features for many different types of ink samples, then, in turn, use those features to identify the types of unseen ink samples.

- **Dynamic Dispatch.** After our machine learning component has predicted an ink sample's type, our system dispatches interpretation calls to an interpreter appropriate for that particular type. Our previous results indicate that using specialized interpreters improves overall accuracy, and our dispatch mechanism provides an efficient way to take advantage of several interpreters, as described in detail in Chapter 5.

3.5 Implementation

In order to conduct user study experiments and evaluate our ink interpretation system, we created the following modules:[4]

- **Ink Collector.** We created this ink collection application to perform experiments on user-provided samples of digital ink. This stand-alone application displays either a string of type-written text or computer-generated images of our above-mentioned representative examples, and asks users to write or draw what they see. We displayed our example text with a standard default typeface (in order not to introduce any bias in using a person's handwriting), but asked users to write on the tablet PC as they normally would on a piece of paper. The user's order of strokes, scale and speed in the ink sample were preserved in the collection. No feedback was provided to the user at each step in order to simulate writing on a piece of paper, and to avoid worrying the user with poor intermediate recognition.

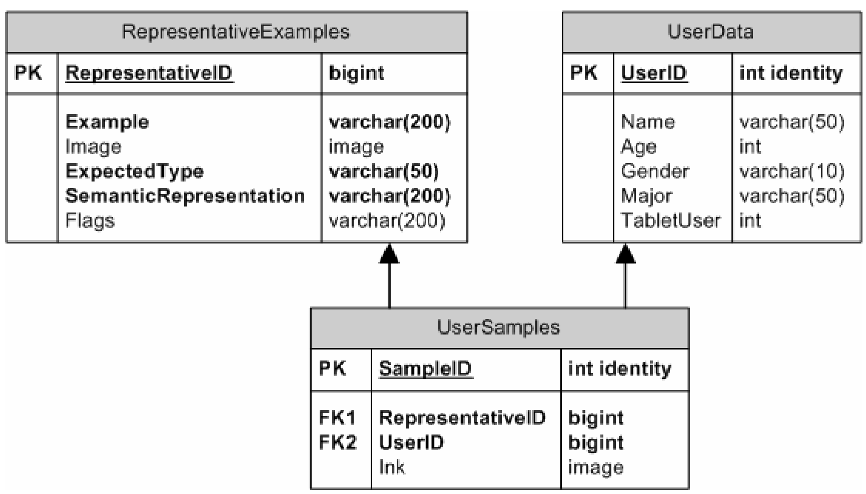


Figure 3-3. A simplified ink database schematic used in our system.

[4] Our system is implemented in C#, which allows easy access to the Microsoft tablet PC software development kit, and easy integration with CLP, which also is implemented in C#.

- **Ink Database.** We collected all user ink samples for training and testing prior to the conduction of our experiments and stored them in this database. This database allowed us to have a consistent dataset for all our experiments, so that we could compare results of different interpreters and type prediction algorithms without bias. After creating representative examples in the database in a single table, we linked all samples thereafter collected to their `RepresentativeID`s as foreign keys and stored them in a user samples table with `SampleID` as the primary key. Throughout our system and this thesis, we use `RepresentativeID` (or `RepID` in short) as a symbolic reference to a specific representative example, and `SampleID` as a symbolic reference to a specific user-provided sample. Figure 3-3 shows a simplified database diagram of our implementation of the database in Microsoft SQL Server 2005.

- **Ink Recognition Accuracy Evaluator.** We created this simple evaluator module to generate tables of recognition results. This evaluator allows us to use the same dataset to compare several interpreters that implement our `Interpreter` interface. Accuracy is measured by the edit distance [Atallah, 1998] between what was interpreted and the original example string used for input.

- **Ink Type Predictor.** Our ink type predictor is the module that carries out the process of ink type prediction (described in detail in Chapter 4). We wrote the feature extraction and data mining code that took an input of digital ink objects, which we represented using the tablet PC software development kit. We utilized the Java implementation of Waikato Environment for Knowledge Analysis (WEKA) [Witten & Frank, 2005] for running our machine learning and feature selection experiments. We created several utility classes in C# that interact with WEKA libraries using IKVM.NET[5], which allows Java-C# interoperability. Accuracy results were stored in text result files for easy viewing, together with

[5] http://www.ikvm.net/index.html

evaluation summaries. We generated all graphs and visualizations in Python using matplotlib[6] and the Python Imaging Library (PIL)[7].

- **Domain-Specialized Interpreters.** We created most of our domain-specialized interpreters in C# to allow for easy integration. For interpreters that make use of external recognition systems, we created special wrapper classes in C# that act as an intermediary layer between our system and the external modules. Communication between our system and the external modules took place either through socket connections (like when connecting to LADDER [Chevalier, 2007]) or through IKVM.NET.

3.6 User Study

We ran two user studies to collect ink samples for all the representative examples we had: twelve students provided ink samples for computer science and ten students provided ink samples for chemistry. All the students had varying backgrounds and majors (computer science, chemistry, among others) with different levels of tablet PC experience. Students were allowed to stop providing ink samples at any point in time of the study. A total of 1958 samples of ink were obtained for our type prediction and dynamic dispatch experiments described in Chapters 4 and 5, with evaluations covered in Sections 4.7 and 5.5 respectively.

---

[6] http://matplotlib.sourceforge.net/
[7] http://www.pythonware.com/products/pil/

Chapter 4
Ink Type Prediction

We describe the details of our approach to ink type prediction in this chapter. We examine in more detail the motivation for doing type prediction in the first place, and describe what features are extracted from ink samples and used as input to our machine learning algorithms. Since we want to perform type prediction across many different types of scenarios and experiments, we show how we use feature selection algorithms to generalize the ink interpretation problem and select the relevant extracted features that are useful for different scenarios. Finally, we evaluate how well we can predict ink types for our experimental data set.

4.1 Motivation

Our motivation in using ink type prediction is based on the superiority of this approach when compared to other approaches that use confidence measures or supply *a priori* contextual information.

Using confidence measures for selecting the best domain-specialized interpreters has several limitations. First, not all interpreters can accurately measure a confidence value for their interpretation result. Some simple interpreters that are based on heuristics do not have confidence measures at all. Second, using a confidence-based ranking scheme requires that a system interpret the ink using all interpreters, a potentially computationally costly process. If an interpreter is known to use many resources for its domain of interpretation, e.g., using an exponential brute-force approach, and the ink to be interpreted does not belong to that domain at all, we will have wasted resources. As such, we aim to predict the domain-specialized interpreters by determining the expected type of the ink, so that only one interpreter does the interpretation work that is required.

Ink type prediction is also beneficial when we do not know the expected type of an ink sample and thus cannot determine the single correct interpreter to use beforehand. In a classroom, for example, we would expect a student's answer to the simple question "three + one = ?" to be "four." There may be students who write "4" instead, however, which may be an equally valid answer, depending on the lesson (math vs. spelling, for example). The answer to a simple yet ambiguous question "What follows in this sequence: 1, 4, 9?" may not be just "16" but a sequence such as "16, 25, 36."

CLP removes the ambiguity in student answers such as "4" vs. "four" with an *aggregator* module. Before passing the representations to a smart aggregator that groups semantically equivalent results, however, we still need a robust interpreter that can interpret both "four" and "4" accurately, and convert each to the desired semantic representation. Thus, it would be beneficial for an interpreter to achieve a high level of accuracy without knowledge of the expected type information, so that it can correctly interpret the different types of answers that may be supplied for the same question. We show that we can achieve this accuracy by predicting the expected type using machine learning.

4.2 Approach

In this section, we cover the general steps taken to obtain maximum accuracy in ink type prediction and to evaluate our methodology. We describe a high level overview of how we use machine learning to predict ink types, what features we extract, what feature selection algorithms we use to choose important features, and how accurately we can predict ink types with different machine learning algorithms. We then detail each of the critical steps in individual sections of this chapter.

- **The Intuition.** Ink type prediction is a classic class prediction problem for which machine learning is well-suited. The problem can be formulated as such: We have a new ink sample of a student's answer that could potentially be any of several expected types (e.g., number, string, Scheme code, etc.). Given a classifier that has been trained with many other previously obtained and correctly classified answers, we ask: Can we predict the expected type of the new ink sample? We hypothesize, and show, that we can.

- **Features to Extract.** The dynamic nature of digital ink strokes provides many possible features to extract for machine learning. We consider both temporal and spatial features of the ink samples. We also extract information about individual strokes as well as the vector of all strokes in each ink sample. We choose some distinct features using domain knowledge to differentiate some of the classes; others are generic features that we feel might be useful based on related work.

- **Dimensionality Reduction.** There are many features that we may extract from the digital ink strokes, but not all of them are critical to helping us in ink type prediction. To prevent overfitting of our class predictors over many useless and counter-effective features, we use feature selection algorithms, also known as dimensionality reduction algorithms, such as information gain or principal components analysis, to prune away unimportant features. We evaluate the effectiveness of several feature selection algorithms to determine those that increase prediction accuracy over the baseline of using all features.

- **Machine Learning Algorithms.** In the absence of prior domain knowledge for our classification problem, we evaluate prediction accuracy using several machine learning algorithms with distinctive learning methods, such as support vector machines (SVMs), decision trees and probabilistic Bayesian networks. We show how the coupling of different machine learning algorithms with any one of multiple feature selection algorithms can improve prediction accuracy for different sets of type prediction experiments.

4.3 The Intuition

Ink type prediction is a classic class prediction problem in machine learning: using extracted features, we predict the class (type, in our case) of a particular ink sample. We also use binary classification to predict *flags* that are indicative of particular types. These flags can be used to further narrow the scope of type prediction possibilities. If our machine learning component predicts that a sample is a sequence, for example, and also that the sample has a "comma" flag, the sample type can be specialized to a sequence that is comma- or space-delineated, as opposed to just a sequence with elements that could be delimited by anything. This delimiter information is used by the sequence interpreter in its segmentation algorithms [Breuel, 2002], which employ heuristics to section ink samples into smaller parts to simplify and improve interpretation. If, for instance, the presence of commas as delimiters is predicted, then the segmentation algorithm within the sequence interpreter will use this fact to first identify commas, before extracting sequence elements. If the comma flag is not predicted, the sequence interpreter will use the variance in spacing distances to determine segmentation before extracting the elements. Thus, we use machine learning classification to predict types, in some cases further narrowing type possibilities based on the presence of particular ink strokes.

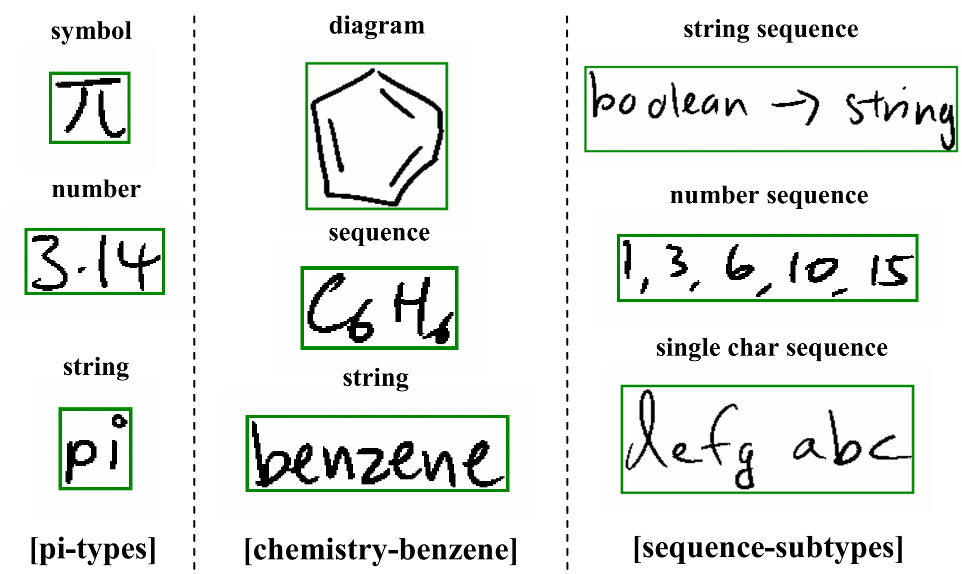


Figure 4-1. Sample ink type prediction experiments that we ran are shown together with their expected type classes.

To observe the ability of classifiers to predict expected types and flags accurately, we ran a number of different experiments over 1958 ink samples that were of different representations and types. Each experiment comprised a subset of the types we wanted to test prediction for. Figure 4-1 shows several ink type prediction experiments that we ran. Our hypothesis was that the correct type can be accurately predicted, and that greater accuracy will be achieved where there are fewer types in the experimental subset.

We obtained some of the types subsets for our experiments from actual questions retrieved from recitation material in the fields of computer science and chemistry. Other subsets that we hypothesized to be useful for our experiments were added to test the limits of the classifiers. Table 4.1 lists the ink type prediction experiments that we conducted and their expected types. In the remainder of this thesis, we will refer to these experiments by the names assigned in the following table.

Table 4.1: The ink type prediction experiments we conducted

No.	Experiment Name	Expected Types (Classes)
1	5-types	Number | String | True-False | Sequence | Scheme Expression
2	no-number	String | True-False | Sequence | Scheme Expression
3	no-string	Number | True-False | Sequence | Scheme Expression
4	no-tf	Number | String | Sequence | Scheme Expression
5	number-scheme	Number | Scheme Expression
6	number-sequence-scheme	Number | Sequence | Scheme Expression
7	number-sequence	Number | Sequence
8	number-string-sequence	Number | String | Sequence
9	number-string-tf	Number | String | True-False
10	number-string	Number | String
11	sequence-commas	Comma | No-Comma
12	sequence-scheme	Sequence | Scheme Expression
13	sequence-subtypes	Single Character | Number | String
14	string-scheme	String | Scheme Expression
15	string-sequence-scheme	String | Sequence | Scheme Expression
16	string-sequence	String | Sequence
17	tf-sequence-scheme	True-False | Sequence | Scheme Expression
18	tf-string-sequence	True-False | String | Sequence
19	tf-string	True-False | String
20	pi-types	Symbol | Number | Fraction
21	scheme-bap	Scheme Expression | Diagram (Box-and-Pointer)
22	chemistry-benzene	Diagram | String | Sequence
23	all-chemistry	Diagram | String | Sequence

4.4 Features to Extract

The dynamic nature of digital ink strokes allows many possible features to be extracted for use by machine learning algorithms. Unlike a rasterized image from a scanner, we can use the time and location information available in the strokes to create feature vectors for each ink sample to use in machine learning. To maximize the information extracted, we considered both temporal and spatial features of the ink samples. We also extracted information about individual strokes as well as the vector of all strokes in each ink sample.

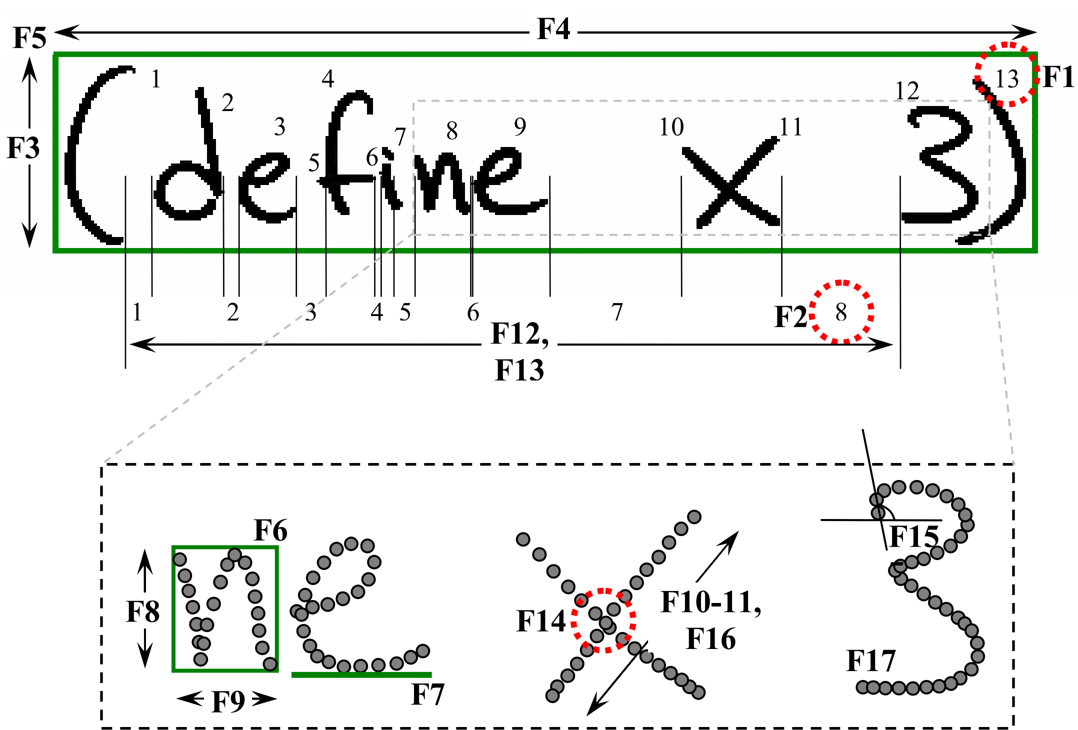


Figure 4-2. Examples of features F1 through F17 are illustrated in this diagram.

With basic knowledge of our domain of expected answer types, we chose several distinct features to differentiate classes; others were generic features that we felt would prove useful to the type domains of short written text or diagrams. Some of the features that we considered are listed in Table 4.2 and illustrated with examples in Figure 4-2. Full descriptions of the features and our hypotheses of their effectiveness in distinguishing types are listed in Appendix C.

Table 4.2: The features we considered

No.	Name
F1	Total number of strokes
F2	Total number of positive inter-stroke adjacent spacing
F3	Sample height span
F4	Sample width span
F5	Sample width-height ratio
F6	Stroke area density of points
F7	Stroke horizontal density of points
F8	Stroke heights
F9	Stroke widths
F10	Stroke lengths
F11	Stroke points count
F12	Stroke adjacent spacing
F13	Stroke adjacent spacing differentials
F14	Number of stroke intersections
F15	Stroke angles
F16	Stroke speeds
F17	Similarity of a stroke to a number

For each feature that applies to individual strokes (F6-F17), we extracted information about the smallest and largest three values, as well as the 25^{th}, 50^{th} and 75^{th} percentiles. We also considered the entire ink sample as a vector of strokes (for each of these features F6-F17) and used this vector as an additional collective feature. For these feature vectors, we calculated their means and variances as additional scalar features.

4.5 Dimensionality Reduction

Not all extractable features are critical to accurate ink type prediction. To prevent overfitting of our type predictors over many useless and counter-effective features, we used feature selection algorithms to prune away the unimportant features.

Using our feature set, we evaluated the effectiveness of several well-known feature selection techniques: information gain (InfoGain), information gain ratio (GainRatio) [Quinlan, 1986], principal components analysis (PCA), Relief-F [Robnik-Sikonja & Kononenko, 1997], and ranking with the square of the weights assigned by an

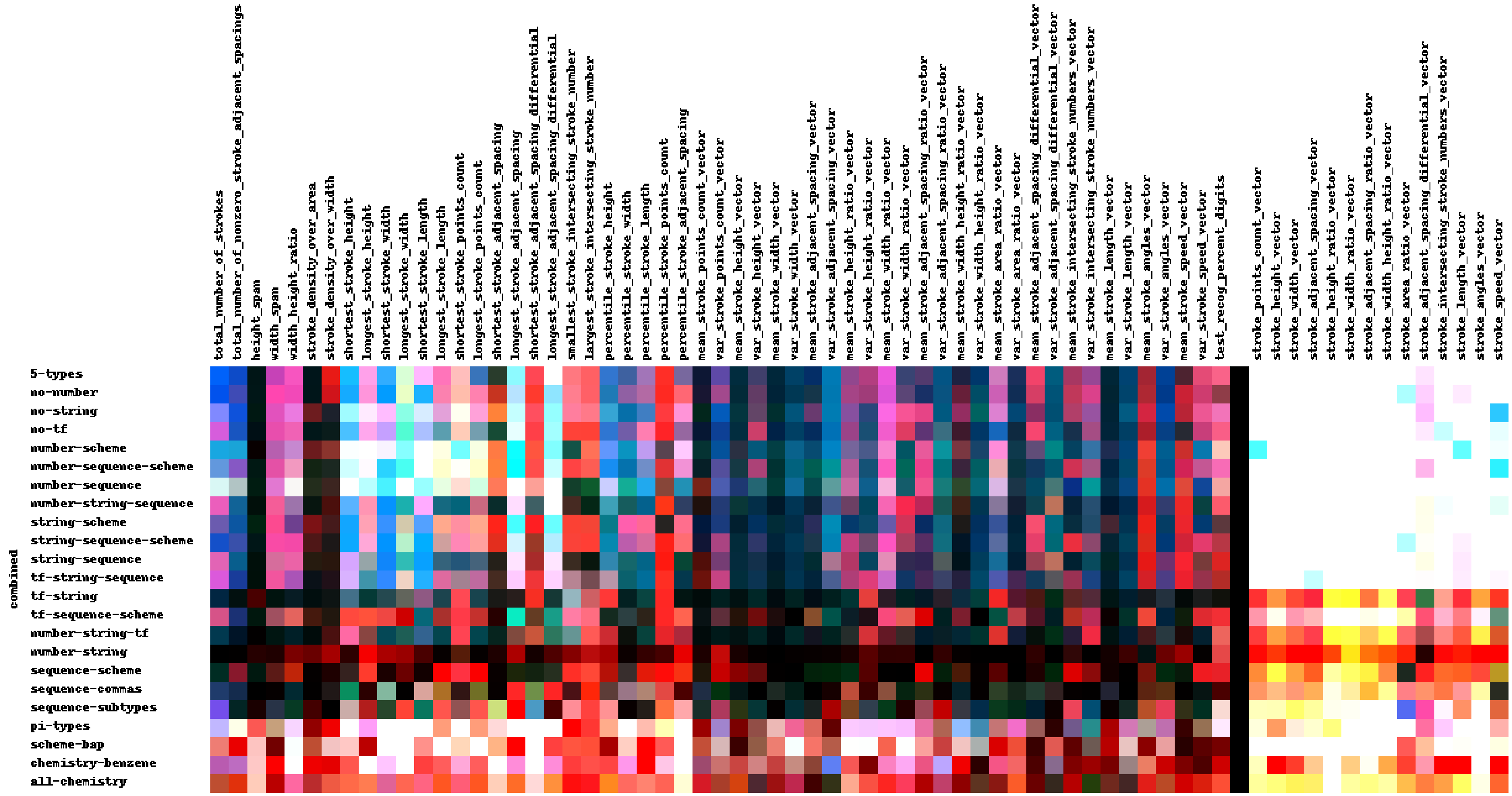


Figure 4-3. This visualization highlights important extracted features. We display extracted features, with many similar ones grouped together for simplicity, on the horizontal axis, and list different experiments on the vertical axis. The colored grid shows a combination of 3 feature selection algorithms (SVM weight, GainRatio and InfoGain) each as individual RGB color channels, with bright colors representing the most important features and dark colors representing the least. For the features grouped together, we used average value of the weight obtained for all features in the group. (A monochrome breakdown is in Appendix D for non-color printing.)

SVM [Guyon et al, 2002]. We wanted to determine if feature selectors would improve prediction accuracy over our baseline of using all features. Figure 4-3 displays a color-coded visualization highlighting important features when we applied our feature selection algorithms to the different experiments.

4.6 Machine Learning Algorithms

Using the WEKA library [Witten & Frank, 2005], we evaluated prediction accuracy with several classification algorithms, each with a distinctive learning method. The algorithms were: an SVM trained with sequential minimal optimization (SMO) [Platt, 1998], a C4.5 decision tree [Quinlan, 1993] (implemented as J48 in WEKA), and a probabilistic Naïve Bayes classifier. We computed the accuracy of our class predictions using stratified cross-validation that was randomized across each of the training and test sets.

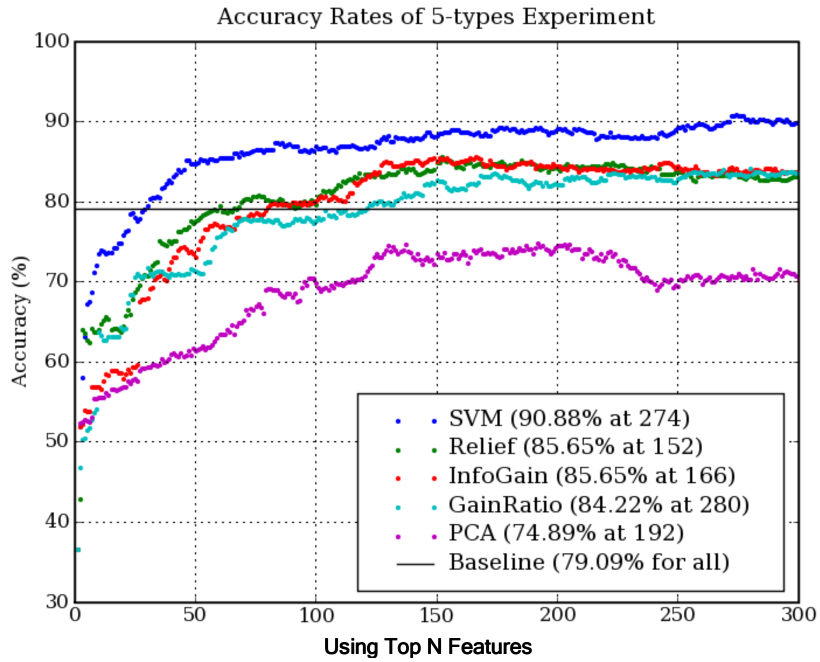
The goal of the evaluation described in this thesis is to highlight the variation in accuracy for a selection of classifiers, instead of finding the perfect classifier for our ink type prediction. We have chosen a representative set of classifiers and feature selection algorithms to show the feasibility of accurate ink type prediction using various methods; other researchers furthering this work may choose to use their preferred classifiers and feature selectors.

4.7 Evaluation

We evaluated ink type prediction with two models: K-fold cross validation and leave-one-out cross validation. Using a uniform distribution, we randomly stratified our ink data sets with $K = 10$ folds across all the representative examples in each experiment. We then selected each fold to be the test set and used the remaining $(K - 1)$ folds for training. The results were then averaged across all K folds.

We performed leave-one-out cross validation by leaving all samples of a single representative example out of the training set each time, and testing classification with each sample of that representative example. The results were then averaged across all representative examples.

(a)



(b)

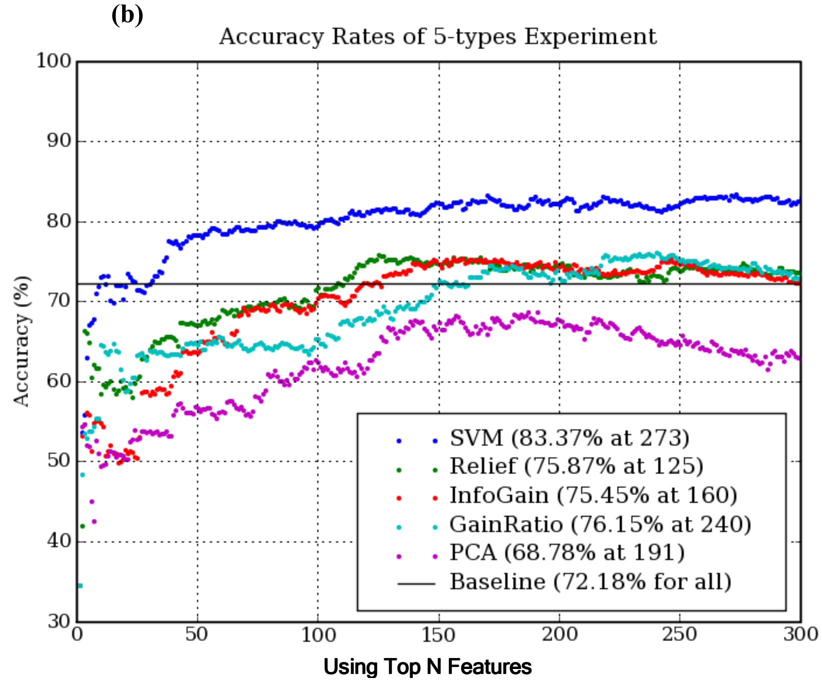


Figure 4-4. Prediction accuracy improves with dimensionality reduction algorithms (such as InfoGain, etc.) over the baseline of using all features with SMO for both **(a)** *K*-fold; and **(b)** leave-one-out cross validation.

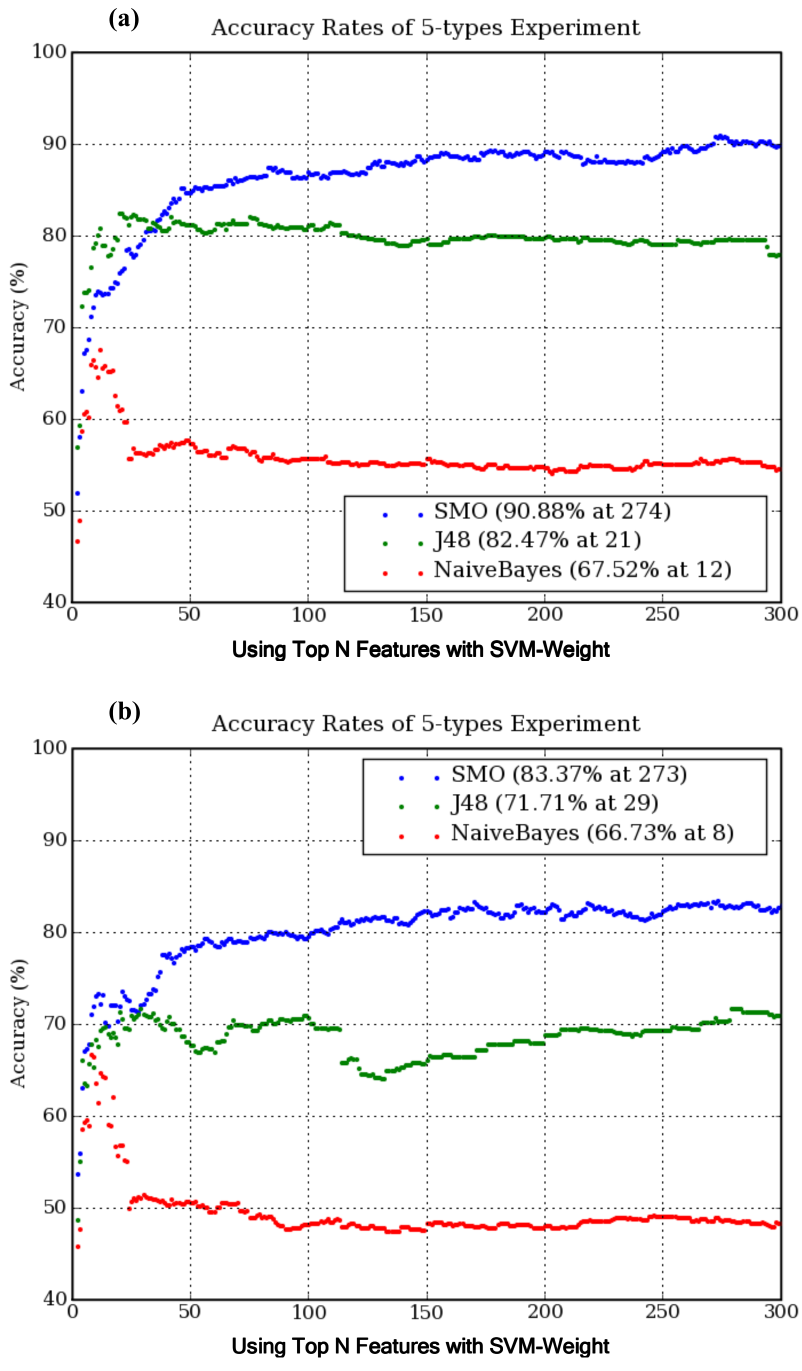


Figure 4-5. These graphs show how prediction accuracy varies for three different machine learning algorithms (SMO, J48 and Naïve Bayes) using SVM-Weight as a feature selector for both **(a)** *K*-fold; and **(b)** leave-one-out cross validation.

4.7.1 *K*-fold Cross Validation Results

Using a *K*-fold cross validation technique allowed us to obtain unbiased accuracy results by preventing testing on the same samples that were used during training.

Figures 4-4 and 4-5 display, for some experiments, the accuracy rates of predicting the correct type according to the number of top features selected. We see that there was no single best classifier, although SMO tended to perform better than the other two learners. Each experiment also required a different optimum number of features to obtain peak accuracy in type prediction. For Tables 4.3 and 4.4, we collected peak accuracies for our five feature selection algorithms using the SMO classifier. Ranking features by SVM weights performed extremely well, increasing prediction accuracy by 10% over the baseline of using all features in an experiment with five types. This feature selector, however, uses a brute-force approach and is time-consuming. Other selectors that employ estimating heuristics or greedy algorithms, such as Relief-F, InfoGain and GainRatio, were able to achieve an improvement of 5% in much less time.

Table 4.3: Expected type prediction accuracy in percent for different groups of experiment classes using 10-fold cross validation with SMO.

Experiment	All Features	SVM Weight	Relief	Info Gain	Gain Ratio	PCA
5-types	79.09	90.88	85.65	85.65	84.22	74.89
no-number	84.27	96.27	90.27	90.87	88.95	81.39
no-string	89.66	98.22	94.99	95.15	95.15	85.78
no-tf	82.23	92.10	86.62	86.73	84.21	79.16
number-scheme	100.00	100.00	100.00	99.73	100.00	100.00
number-sequence-scheme	89.54	99.81	94.95	95.31	95.67	87.74
number-sequence	99.69	100.00	100.00	100.00	99.69	100.00
number-string-sequence	87.11	94.14	88.72	88.57	88.72	86.23
number-string-tf	83.51	93.43	87.23	86.70	87.23	81.20
number-string	78.87	93.31	84.22	83.95	83.68	83.95
sequence-commas	87.97	100.00	93.98	95.08	95.62	90.16
sequence-scheme	86.89	99.75	93.68	92.96	93.44	93.93
sequence-subtypes	96.17	100.00	98.36	98.36	97.81	96.72
string-scheme	95.39	99.65	97.78	98.12	97.44	95.05
string-sequence-scheme	87.64	97.52	92.71	93.75	91.41	86.21

Experiment	All Features	SVM Weight	Relief	Info Gain	Gain Ratio	PCA
string-sequence	97.96	100.00	98.51	97.96	98.33	96.48
tf-sequence-scheme	88.23	99.15	94.95	93.90	94.74	94.53
tf-string-sequence	91.88	99.00	95.69	95.86	94.70	90.39
tf-string	93.82	99.76	96.43	97.38	96.43	95.48
pi-types	95.08	98.36	98.36	98.36	96.72	96.72
scheme-bap	100.00	100.00	100.00	100.00	100.00	100.00
chemistry-benzene	98.00	100.00	100.00	100.00	100.00	100.00
all-chemistry	93.33	100.00	95.66	97.33	95.33	95.00

4.7.2 Leave-One-Out Cross Validation Results

This method of cross validation is important because it allows us to effectively test that our hypothesis works even with our relatively small selection of representative examples. Although we have a total of 181 representative examples presented in this thesis, our individual experiments have ranges spanning only 5 representative examples (e.g., `chemistry-benzene` with 3 types) to 88 representative examples (e.g., `5-types`). If we can show that a high accuracy of predicting types can be obtained without including every representative example in the training set, then our system should be robust enough for a larger universe of possible ink answers beyond the 181 examples we have chosen.

We saw that leave-one-out cross validation still performed relatively well (see Table 4.4), with peak accuracies lower by only 6-10% than those obtained with *K*-fold cross validation. We discuss this observation later in Section 4.7.5.

Table 4.4: Expected type prediction accuracy in percent for different groups of experiment classes using leave-one-out cross validation with SMO.

Experiment	All Features	SVM Weight	Relief	Info Gain	Gain Ratio	PCA
5-types	72.18	83.37	75.87	75.45	76.15	68.78
no-number	78.21	91.23	84.17	83.93	83.51	76.31
no-string	82.98	95.85	89.96	88.72	87.64	79.27
no-tf	74.95	87.75	77.76	77.79	76.51	72.91
number-scheme	99.72	100.00	100.00	99.75	100.00	100.00
number-sequence-scheme	83.58	98.42	90.47	88.58	88.38	81.33
number-sequence	99.07	100.00	99.76	99.30	99.43	99.76

Experiment	All Features	SVM Weight	Relief	Info Gain	Gain Ratio	PCA
number-string-sequence	81.48	90.53	82.19	78.04	78.62	81.26
number-string-tf	73.71	90.51	76.19	75.86	77.22	75.91
number-string	74.61	92.77	79.86	81.73	82.80	78.54
sequence-commas	60.50	100.00	80.37	73.36	75.59	63.45
sequence-scheme	76.63	99.73	88.25	90.75	88.25	89.25
sequence-subtypes	74.05	95.95	85.52	81.93	80.50	76.22
string-scheme	92.56	99.52	96.46	96.79	96.20	93.66
string-sequence-scheme	81.72	94.23	86.48	86.89	84.83	81.98
string-sequence	93.99	99.27	94.83	95.08	95.08	96.04
tf-sequence-scheme	77.44	97.70	87.95	86.48	88.68	88.32
tf-string-sequence	86.61	94.95	88.70	89.04	88.66	87.61
tf-string	87.72	99.43	90.90	92.80	92.71	89.80
pi-types	43.63	66.66	65.15	63.63	65.15	63.63
scheme-bap	75.00	100.00	100.00	100.00	100.00	100.00
chemistry-benzene	30.00	60.00	60.00	60.00	60.00	58.00
all-chemistry	85.66	99.00	92.00	92.66	91.00	90.00

4.7.3 Evaluation by Number of Classes

In order to understand the accuracy and effectiveness of ink type prediction with respect to the number of possible types, we re-arranged the peak results obtained in Tables 4.3 and 4.4 and ranked experiment accuracy by the number of types, as shown in Table 4.5. We also plotted graphs showing the mean peak prediction accuracies, grouped by number of types, for both K-fold and leave-one-out cross validation in Figure 4-6.

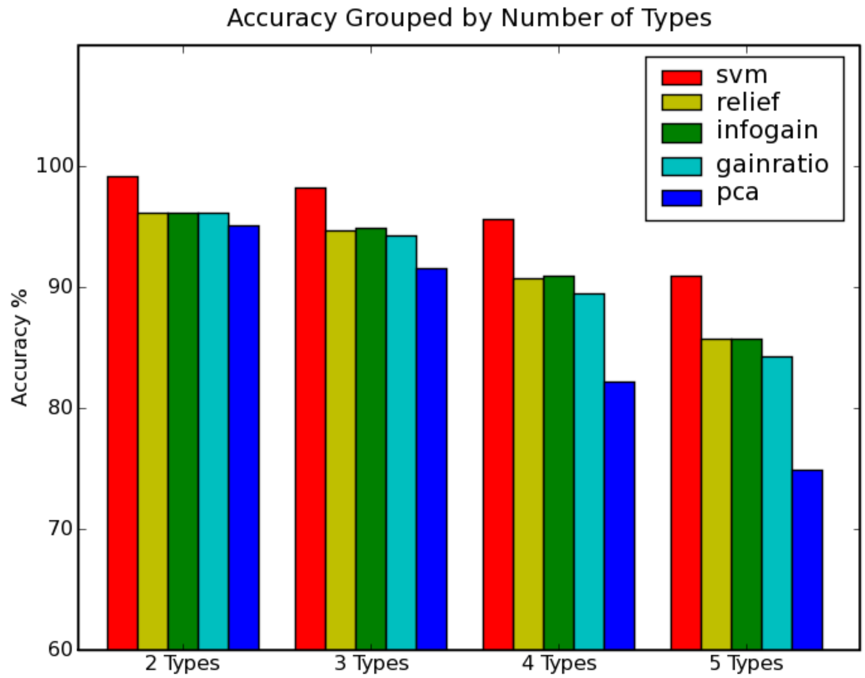
We observed from our experiments that peak prediction accuracy decreases when there are more types from which to predict. This observation is typical of machine learning classification problems. As such, we conclude that the more ambiguous a case we present for ink type prediction, i.e., with more types from which to predict, the harder it is for our type predictor to accurately guess the context of the ink. Not too surprisingly, if we decrease the number of possible types, e.g., by means of more extensive domain knowledge or some context known by the instructor *a priori*, then the system may be able to more accurately guess the context, and use this context, as we later describe in Chapter 5, to improve interpretation accuracy.

This thesis also notes that the correlation between the number of types used in the experiments and the accuracy of prediction depends on which types are actually used, as well as their relative resemblance. The prediction accuracies, for example, in the experiments *number vs. string*, *sequence vs. Scheme expression*, and *sequence vs. number*, exhibit high variance even though the experiments each have only two types. This is because sequences highly resemble Scheme expressions, and our chosen representative strings highly resemble our numbers. The leave-one-out cross validation results for three types show on average a significantly lower accuracy than that of four types because of the poor performance of two experiments with three types: `pi-types` and `chemistry-benzene`. We discuss this anomaly later in Section 4.7.5.

Table 4.5: Peak prediction accuracy ranked by number of types

Experiment	# types	*K*-fold (%)	Leave-one-out (%)
number-scheme	2	100.00	100.00
number-sequence	2	100.00	100.00
sequence-commas	2	100.00	100.00
scheme-bap	2	100.00	100.00
string-sequence	2	100.00	99.27
tf-string	2	99.76	99.43
sequence-scheme	2	99.75	99.73
string-scheme	2	99.65	99.52
number-string	2	93.31	92.77
all-chemistry	3	100.00	99.00
sequence-subtypes	3	100.00	95.95
chemistry-benzene	3	100.00	60.00
number-sequence-scheme	3	99.81	98.42
tf-sequence-scheme	3	99.15	97.70
tf-string-sequence	3	99.00	94.95
pi-types	3	98.36	66.66
string-sequence-scheme	3	97.52	94.23
number-string-sequence	3	94.14	90.53
number-string-tf	3	93.43	90.51
no-string	4	98.22	95.85
no-number	4	96.27	91.23
no-tf	4	92.10	87.75
5-types	5	90.88	83.37

(a)



(b)

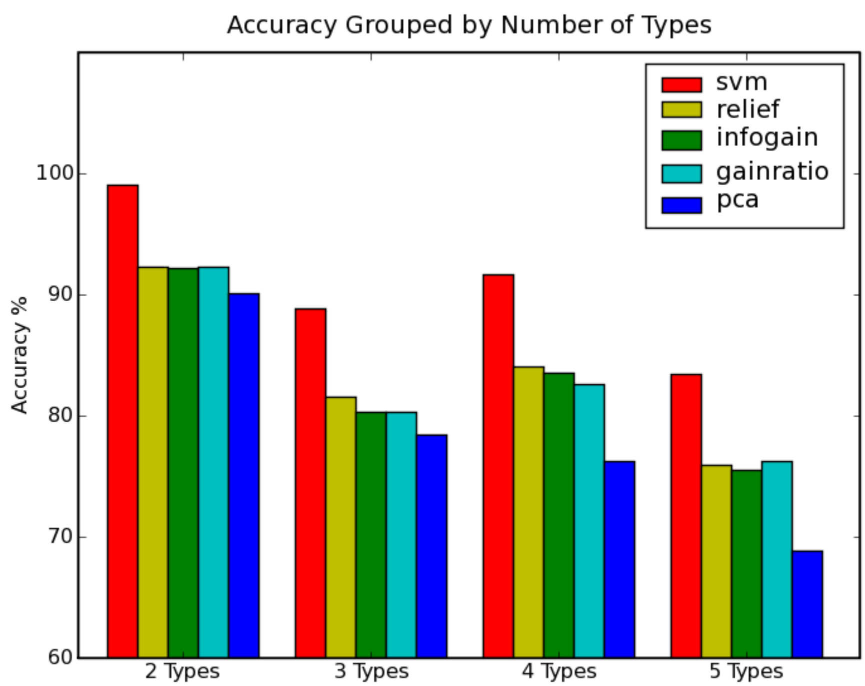


Figure 4-6. Mean prediction accuracy grouped by number of types using **(a)** K-fold; and **(b)** leave-one-out cross validation. The mean accuracies decrease with more types.

4.7.4 Evaluation of Feature Importance

We perform an evaluation of our original hypotheses of feature importance (described in Appendix C) of the features we covered in Table 4.2. It is interesting to assess the validity of our original hypotheses as to which suggested features would improve prediction accuracy. A method for knowing if a feature is relatively important in differentiating type *A* from *B* is to observe the ranking of the feature after feature selection algorithms have been applied to experiments containing type *A* and *B*. A formal investigation is beyond the scope of this thesis, but we looked at our visualization of feature importance in Figure 4-3 to obtain an informal evaluation of our originally chosen feature set. This evaluation is listed in Table 4.6. We could not make conclusions on the effectiveness of several of the features, mainly because they differentiated between different individual character classes (such as complex intersecting characters vs. simple single-stroke ones); our experiments, however, classified many characters in bulk within strings, sequences, Scheme expressions, etc., all of which mixed the different character classes together.

Table 4.6: Features extracted and their effectiveness in distinguishing types

No.	Distinguishes Between	Successful?
F1	Number / String vs. Sequence / Scheme	Yes (*see number-sequence, number-scheme*)
F2	Short / Diagram vs. Long / Sequence	Yes (*see number-sequence, number-scheme*)
F3	Text vs. Diagram	Yes (*see pi-types, scheme-bap, chemistry-benzene*)
F4	String / Number vs. Sequence / Scheme	Yes (*see number-sequence, number-scheme*)
F5	Text vs. Diagrams	Yes (*see pi-types, scheme-bap, chemistry-benzene*)
F6	Text vs. Diagrams	Moderately (*see scheme-bap, chemistry-benzene*)
F7	Text vs. Diagrams	Moderately (*see scheme-bap, chemistry-benzene*)
F12	Character / Number vs. String / Sequence	Yes (*see sequence-subtypes*)
F13	String vs. Sequence	Yes (*see string-sequence, string-scheme*)
F14	Text vs. Diagram	Yes (*see pi-types, scheme-bap, chemistry-benzene*)
F16	Text vs. Diagram	Yes (*see pi-types, scheme-bap*)
F17	Number vs. String	Yes (*see number-string*)
F8, F9, F10, F11, F15		Cannot conclude

We note that different experiments require different features to effectively differentiate the types; features that work in one experiment involving a certain type may not necessarily achieve the same success in another experiment. As such, data-mining and extracting all the features proposed in Section 4.4, and using generic feature selection algorithms to prune away unimportant features dynamically proves to be a viable approach.

4.7.5 Discussion

We observed that the accuracy of predicting the correct class in the number-string experiment was low, despite being a binary classification problem. There is a challenge associated with the distinction between numbers and strings: It is inherently hard to tell whether a simple vertical stroke is a '1' (one), 'I' (capital-i) or 'l' (lowercase-L). If that stroke were to be slightly tilted, we could add either of '/' or '\' to the list. This challenge is the reason that makes biasing with contextual information useful in improving interpretation accuracy, but fails to help us when we are doing ink type prediction. We have many such ambiguous ink stroke samples collected as part of this research, and they lack the contextual information for accurate prediction, thus lowering our prediction accuracy in that experiment.

Leave-one-out cross validation showed poorer prediction accuracy results than *K*-fold cross validation, mainly because the classifiers were not trained with the tested representative samples in the former. The accuracy obtained is still relatively high at greater than 83% for up to five types, however, showing it is possible to accurately predict correct expected types or flags of representative samples that have not been observed before.

We reason that unusually low accuracy in leave-one-out cross validation for both `pi-types` and `chemistry-benzene` experiments was observed because there were too few representative examples present in the training set for such validation. If the classifier had been trained with only "symbol" and "number" classes for pi, for example, it would not be able to predict an unknown "fraction" class when presented with a sample that was a fraction.

To better understand the shortcomings of our ink type predictor system, we also ran an experiment that attempted to classify our eight different expected types with *K*-fold cross validation across all collected samples. There is a low likelihood of a question being so ambiguous that its answer could be any one of eight different types, hence this experiment was conducted purely for additional information. We obtained an 84.22% prediction accuracy using the SMO classifier and InfoGain feature selection algorithm. A full confusion matrix of the classification is listed in Appendix E. We see that misclassification often occurred between any two of strings, sequences, and Scheme expressions when the type predictor was trained across all eight types. As such, we conclude that the features we originally extracted are still relatively insufficient to achieve a full distinction across these very similar types.

Chapter 5
Interpretation using Dynamic Dispatch

In this chapter, we describe the details of our approach to improving ink interpretation using dynamic dispatch. This approach is promising because our past results have shown that *a priori* information about an answer type improves ink interpretation significantly [Rbeiz, 2006]. We also have shown in Chapter 4 how ink type prediction provides an accurate prediction for certain answer types. Combining these two ideas, we can create a system that improves ink interpretation by dynamically dispatching interpretation calls to the best interpreter for a sample's predicted answer type. As stated earlier, we hypothesize that this new interpreter will be close in accuracy to an interpreter requiring explicit *a priori* expected type information, and much more accurate than interpreters that use no expected type information.

5.1 Approach

In this section, we describe the design, implementation, and evaluation of our dynamic dispatch method and variations, which take advantage of predicted ink sample types. We made several iterations in designing such an interpreter for improved accuracy. The next few sections will elaborate on the following in greater detail:

- **The Dynamic Dispatch Interpreter (DDI).** We describe the basic dynamic dispatch interpreter in detail and explain how ink type prediction can be used as a switch to dispatch ink dynamically to static interpreters.

- **Nested Dynamic Dispatch Interpreters (NDDI).** Nested DDIs enable the dynamic dispatching of ink with types and subtypes by having other DDIs as one of their internal interpreters. This is similar to a tree with static interpreters as leaves. These NDDIs make use of a preprocessing stage, which we call *preparation*, which allows us to work with a hierarchy of ink types and subtypes.

- **Cross Validation Interpreters (CVI).** Cross validation interpreters allow us to evaluate interpretation accuracy without mixing our training and test data sets of ink samples. These CVIs are built in with multiple distinct DDIs, and each DDI is trained and tested with different ink sample sets. An equivalent Nested CVI (NCVI) also has been created for NDDIs.

5.2 The Dynamic Dispatch Interpreter (DDI)

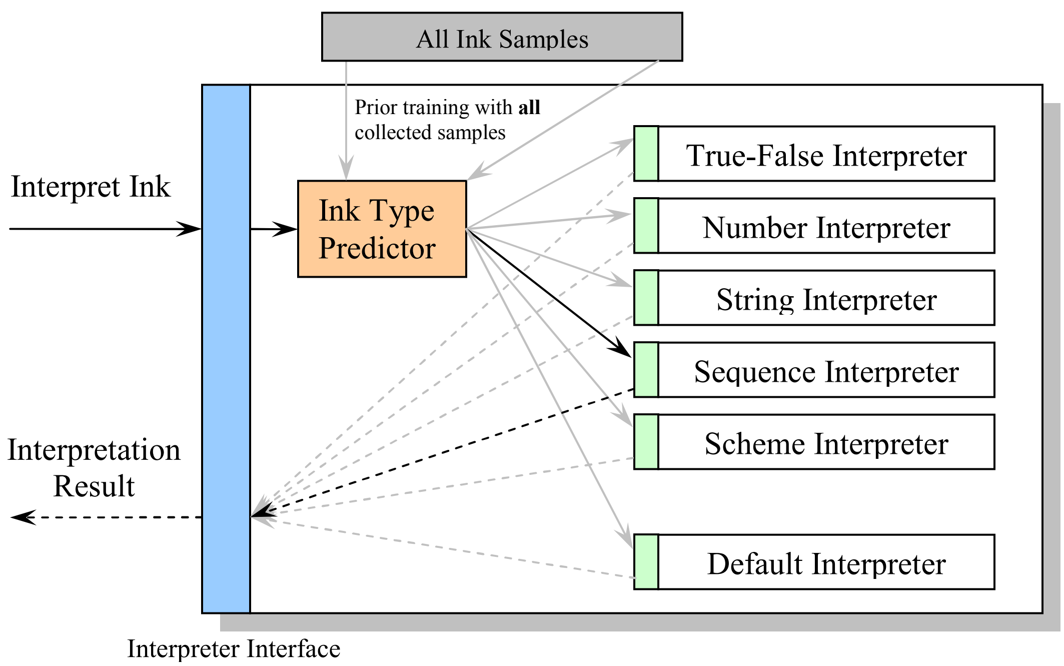


Figure 5-1. A simple schematic demonstrating the Dynamic Dispatch Interpreter at work.

Using the same interpreter interface that we created specially for the CLP system, we can create Dynamic Dispatch Interpreters that use an internal Ink Type Predictor previously trained on our cumulative set of ink samples. (For the rest of this thesis, "training a DDI" will mean "training the Ink Type Predictor inside the DDI."). The interpreter will use its internal Ink Type Predictor module to perform type prediction tests on new ink samples and dynamically dispatch the ink sample to the domain-specialized interpreter of the predicted type for recognition. The dispatching of the ink to be interpreted is illustrated in Figure 5-1.

This Dynamic Dispatch Interpreter demonstrates that we may perform interpretation using domain-specialized interpreters without prior knowledge of expected type information. We have hypothesized that the interpretation accuracy of such a DDI will be close to that of an interpreter provided with expected type information.

5.3 Nested Dynamic Dispatch Interpreters (NDDI)

A single level of type prediction is insufficient for more complex domain-specialized interpreters. We can interpret sequences, for example, with greater accuracy as mentioned in Section 4.3 with more type information, describing the subtypes or flags of the sequence. As we found in Section 4.7.3, however, the more possible types, the lower the prediction accuracy obtained. Adding these sequence subtypes and comma flags as newer expected types from which to predict will result in an "explosion" of combinatorial possibilities—we would need a different class for each combination! Sequences, for example, can be further classified into three different subtypes—number, single character and string—each with two possible flags—comma and bracket. With these additions, we would need up to 12 new types in the place of our original sequence type.

We solved this scalability problem by creating a preprocessing *preparation* stage in our interpreter interface to modify the state of each interpreter and influence subsequent interpretation[8]. An interpreter can be prepared over multiple calls; it can be first alerted to expect a sequence, for example, then prepared to expect a *numbered* sequence, and finally made to expect a *comma-delimited* numbered sequence. This extensible preparation phase allows us to reuse the same specialized interpreters with just

---

[8] Preparing an interpreter is a similar concept to using factoids in Microsoft's ink libraries.

some state-modification to improve accuracy, without having to create entirely different interpretation algorithms.

The Nested Dynamic Dispatch Interpreter (NDDI) uses preparation to allow interpreted ink to virtually traverse a decision tree of type predictors, before the ink is dispatched correctly to the relevant domain-specialized interpreter. Figure 5-2 illustrates the dispatch mechanism of an NDDI.

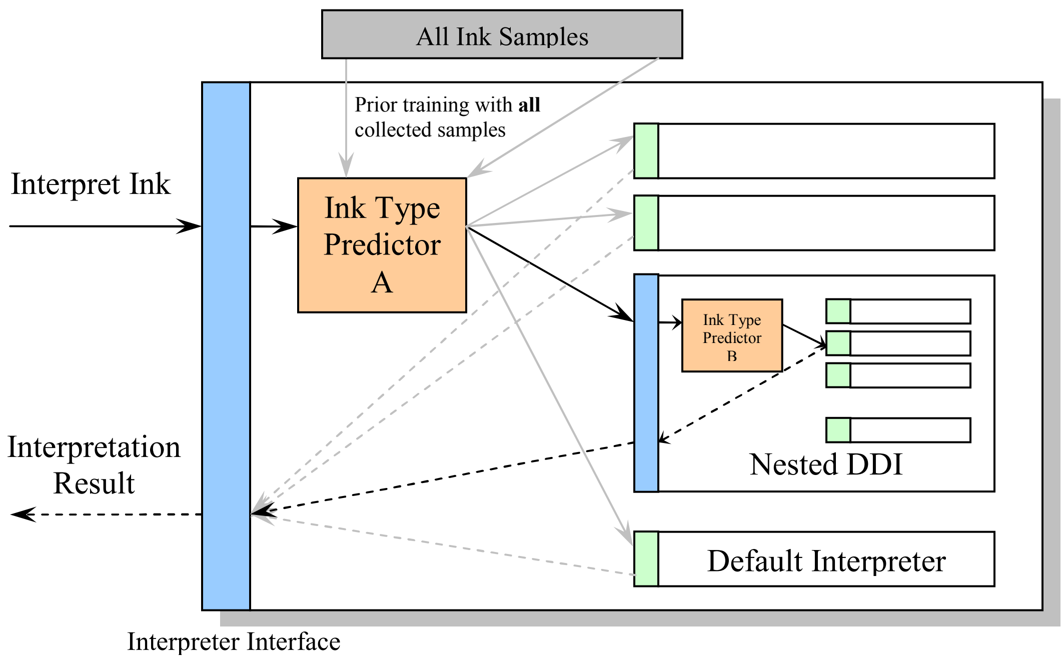


Figure 5-2. This schematic shows how Nested Dynamic Dispatch Interpreters work with one level of nesting.

The NDDI functions like a DDI, with the exception that the internal interpreters (to which ink is dispatched) can be DDIs themselves. These internally nested DDIs may store a different Ink Type Predictor for predicting the different subtype classes of ink, like the sequence subtypes mentioned. We may nest NDDIs recursively and limitlessly for our different flags as well. At each level of dynamic dispatch, the different classes predicted by the Ink Type Predictor would *prepare* the correspondingly predicted NDDI or specialized interpreter. NDDIs transfer this preparation to their internally nested interpreters in addition to their own preparation from their Ink Type Predictor member.

This *chain of preparation* continues down the tree of NDDIs until a specialized interpreter leaf is reached. This leaf interpreter would have received multiple preparatory calls and may thus used the information obtained to interpret the ink more accurately.

The expected type of a sample "1, 2, 3," for example, would be a number sequence, delimited by commas. The best NDDI to interpret this sample will thus have three nested levels: the first to predict that the sample is a sequence (out of the five types we have in total in introductory computer science); the second to predict that the sequence is of numbers; finally, the last level to predict that this number sequence is comma-delimited. Each level of prediction will be passed down in the chain of preparation, and the leaf interpreters would then know to use the predicted contextual information of a comma-delimited number sequence to interpret the ink sample more accurately.

5.4 Cross Validation Interpreters (CVI)

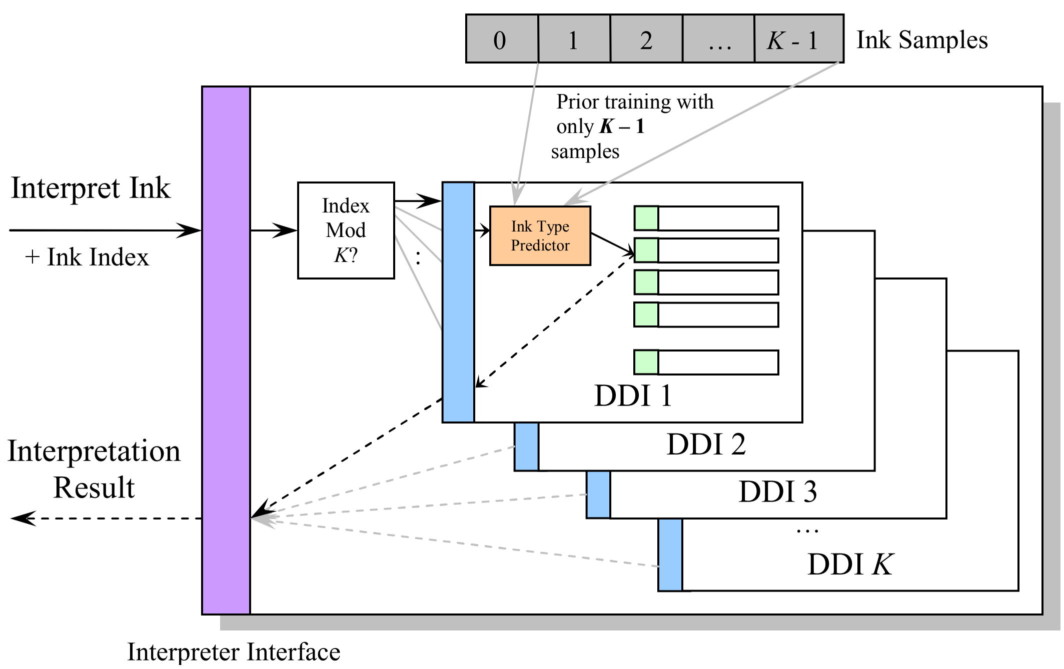


Figure 5-3. A schematic of a simple Cross Validation Interpreter with *K*-folds is shown.

In order to ensure that test ink sample data is never used for training the DDI that we want to evaluate, we created a *K*-fold Cross Validation Interpreter (CVI-K) to wrap the DDI. A schematic of the CVI is shown in Figure 5-3. This CVI encapsulates *K* different copies of the same DDI. Each DDI is specifically designated to test a subset of non-overlapping 1/*K* of the total ink samples in the experiment, and has been trained with the remainder (1 – 1/*K*) of the total number of ink samples.

CVIs differ from DDIs mainly in that CVIs require experiment contextual information and thus cannot be deployed for subsequent use in tightly coupled applications such as CLP, which have no notion of experimental conditions. The CVI makes use of some globally accessible auxiliary data (the index of the ink being interpreted out of all ink samples within the experiment) in order to properly dispatch interpretation to the specific DDI that is meant to "test" the currently inputted ink sample. This technique allows us to evaluate 10-fold cross validation of our DDI's prediction and interpretation accuracy if we set *K* to be 10. We chose to use the ink index modulo *K* to determine the index of DDI copies to which to dispatch the ink, because it provides an easy way to distribute all ink samples equally among the *K* DDI copies, with uniformly distributed test and training sets.

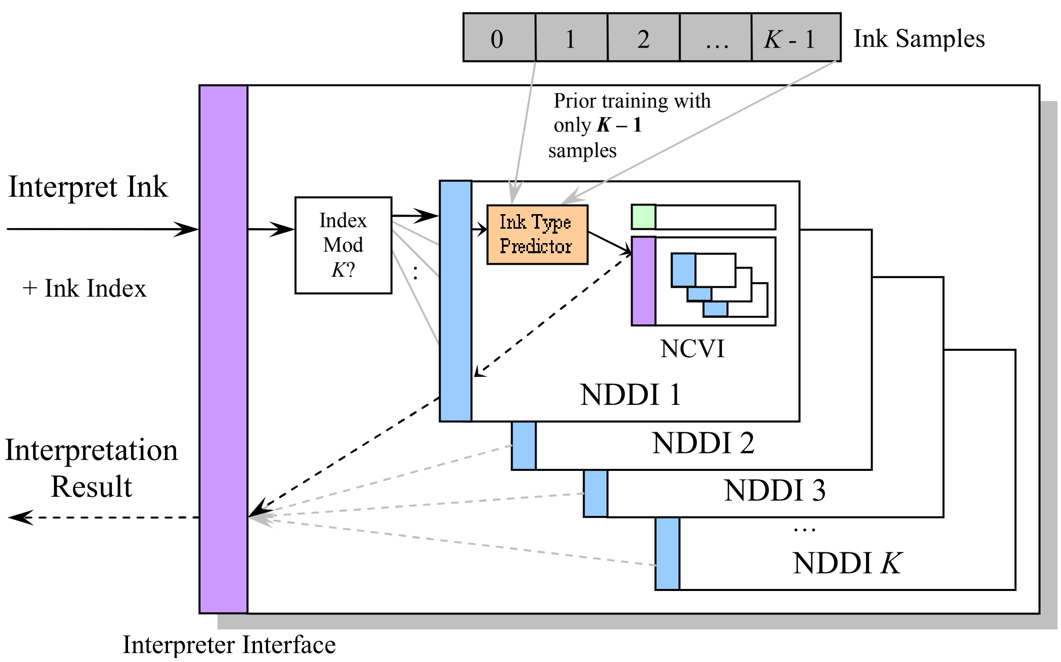


Figure 5-4. A schematic of a simple Nested Cross Validation Interpreter with *K*-folds is shown.

In a similar fashion, we also created the *K*-fold Nested Cross Validation Interpreter (NCVI-K) to ensure we do not train the NDDIs with our intended test ink samples while evaluating the prediction and interpretation accuracy of our NDDIs. Figure 5-4 shows a simple schematic of ink dispatch through an NCVI, which has yet another NCVI nested within the NDDIs.

The CVIs and NCVIs are not meant for deployment and require knowledge of the experimental framework, e.g., ink sample index numbers; they are used only for evaluating interpretation accuracy of our dynamic dispatching architecture. In deployment, the DDIs and NDDIs should be used—trained with *all* prior ink sample data—instead of the CVIs and NCVIs, respectively.

5.5 Evaluation

We evaluated our dynamic dispatch interpretation system by computing final ink interpretation accuracy for the domain of introductory computer science. Accuracy is measured as the edit distance [Atallah, 1998] between the interpreter's output and the original example string used for input.

We chose this domain, consisting of five types—numbers, strings, sequences, true-false, Scheme expressions—because most of the student answers in the domain are in the form of text, not drawings. We could thus make comparisons easily with other text interpreters such as Microsoft's default interpreter, as well as our already deployed interpreter (INKv3).

5.5.1 Base Type Results

After running our interpretation experiments, we found that interpreters with type information provided *a priori* for each ink sample performed the best, but that our dynamic dispatch interpreter was a close second. The interpretation results for the five different base types in the introductory computer science domain are listed in Table 5.1. Our latest version of the deployed CLP interpreter (INKv3) obtained 89% accuracy while an earlier version (INKv1) obtained 87%. Both of these interpreters made use of expected type information that we provided to bias ink pre-processing and interpretation for better accuracy. Microsoft's default interpreter obtained 62% accuracy, mainly due to the fact that it was not trained for the domain of introductory computer science and did not bias for expected types.

Table 5.1: Base type results in percent for our different interpreters on the same data set grouped by the 5 base types for the introductory computer science domain.

Base Type	INKv3	INKv1	NDDI	NCVI-10	NCVI-4	Microsoft
Number	98.27	98.27	95.24	93.51	94.37	30.74
Scheme Expression	84.72	84.72	84.72	84.72	84.79	80.91
Sequence	87.03	76.22	87.03	83.35	81.61	71.17
String	78.06	78.06	77.57	74.08	73.69	54.95
True-False	97.64	97.64	97.64	97.64	97.64	74.53
Total	**81.82**	**80.18**	**80.52**	**78.53**	**78.44**	**51.00**
Total (Equal Weight)	**89.14**	**86.98**	**88.44**	**86.66**	**86.42**	**62.46**

Our approach described in this thesis obtained close to 87% accuracy, comparable with our other interpreters developed for use with CLP. The main difference was that our dynamic dispatch interpreter (NCVI-10) required no contextual information to be provided *a priori* for each ink sample, and relied instead on machine learning to predict the expected type just from information extracted from the digital ink. The good news is that, as we had hypothesized, with the same ink input, our interpreter outperformed Microsoft's default interpreter by 24%, while almost reaching the level of accuracy of our best *a priori* interpreter, INKv3 (see Figure 5-5).

The detailed table of interpretation results grouped by representative types is listed in Appendix B.

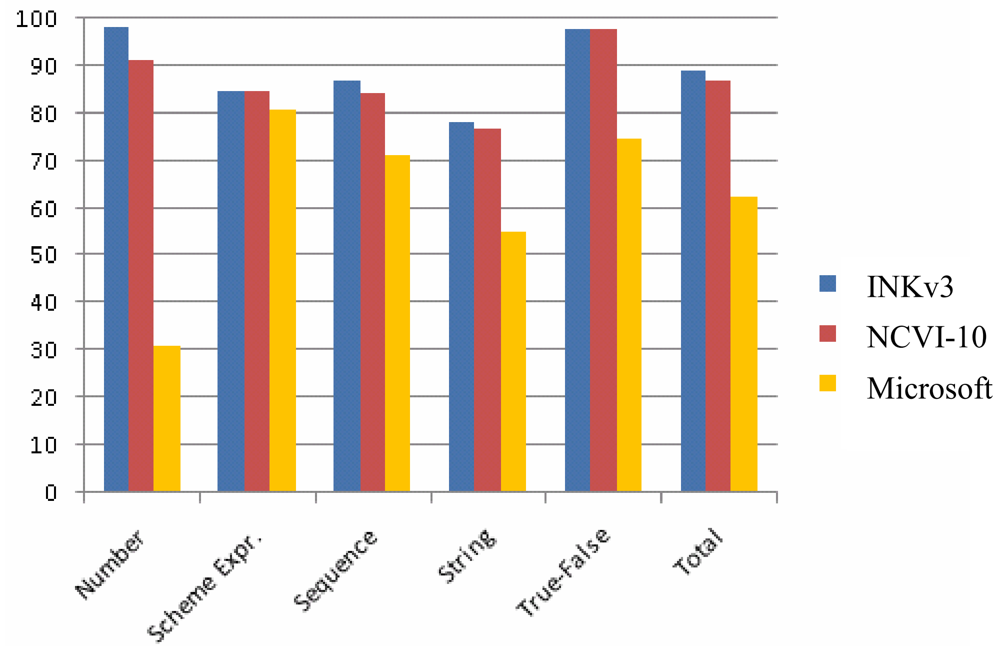


Figure 5-5. This graph shows overall interpretation accuracy: the INKv3 interpreter was provided with contextual type information and performed the best at 89% for all samples; our interpreter NCVI-10 achieved a comparable 87% without such information, better than Microsoft's interpreter at 62%.

5.5.2 Discussion

On the whole, we are pleased with the performance of our dynamic dispatch interpretation method: Its accuracy in predicting and interpreting five different ink sample types was very close to the accuracy of our best interpreter that required *a priori* ink type information, and much better than an interpreter with no ink type information. Its architecture allows for easy integration of additional specialized interpreters unlike the other interpreters we tested, and requires far less input from an instructor using it in an application such as CLP.

There are limitations to this approach, however. A deployed ink type predictor in a DDI will only have knowledge of a small subset of the universe of representative examples. Leave-one-out cross validation results showed it might be possible to extrapolate additional new unknown representative examples, but the system would undoubtedly deteriorate in prediction performance the more the examples come from outside our training subset. The time saved for the instructor, thus, becomes time gained for the ink interpreter "trainer" in creating relevant training sets. In addition, we would need to perform retraining occasionally after deployment, but this activity could be as simple as labeling real data collected post-deployment.

Chapter 6
Related Work

Our work draws on research from various subfields of ink interpretation. We mentioned in Section 2.1 sketch recognition work on sequences, chemical diagrams [Ouyang & Davis, 2007], box-and-pointer diagrams [Chevalier, 2007], and marking [Wu, 2008]. Here we discuss two other related areas—handwriting recognition research and confidence measure-based approaches.

6.1 General Approaches

Handwriting recognition research is a very active field. Variations in writing styles cause difficulty in developing highly accurate handwriting recognizers [Liu & Cai, 2003] [Plamondon & Srihari, 2000]. There are many general approaches that aim to improve ink interpretation across the board, without any domain-specific restrictions. Most of these successful approaches to date use artificial intelligence algorithms. Specific techniques used include support vector machines (SVM), hidden Markov models (HMMs) [Hu et al, 1996] [Yasuda et al, 2000], neural networks, genetic algorithms, and convolutional time delay neural networks (TDNN). Some of these statistical and machine-learning approaches support online (e.g., [Bellegarda et al, 1994], [Anquetil & Lorette, 1995]) and offline (e.g., [Seni & Cohen, 1994], [Srihari & Keubert, 1997]) recognition of handwriting; other approaches may also be writer-independent (e.g., [Hu et al, 2000]). All these approaches use different representations and metrics for segmenting handwriting [Breuel, 2002], and report varying measures of success for their respective domains of recognition use.

Apart from artificial intelligence algorithms, different domain-specific heuristics have also been used to further improve handwriting recognition. Handwritten *sequence* interpretation, for example, is useful in recognizing postal addresses [Srihari & Keubert, 1997] and general document optical character recognition (OCR) work [Manmatha & Srimal, 1999]. There are many punctuation detection heuristics (e.g., [Seni & Cohen, 1994]), as well as spatial detection measures (e.g., also [Mahadevan & Nagabushnam, 1995], [Wang et al, 2005]) which are applicable for the domain of English sequence interpretation, but may not be useful with other written forms like classical Chinese, or chemical structures. As such, there is currently no ideal "universal handwriting recognizer" that has been developed by researchers. The best recognizers to date work well only in selected narrow domains, and they often make use of specialized heuristics or have been subjected to training with many ink samples.

6.2 Confidence Measure-based Approaches

Studies have been done to compare different confidence measures for deciding when to accept or reject interpreted results. Examples of such confidence measures are: recognition score, likelihood ratio [Brakensiek et al, 2002], and estimated posterior probability [Pitrelli & Perrone, 2003]. These studies illustrate the usefulness of confidence measures in the unsupervised retraining of handwriting data, and in improving interpretation accuracy by being able to reject a fraction of the handwritten input. We chose not to use confidence measures despite their useful potential, because not all specialized interpreters that we would like to use have confidence measures, or can accurately measure a confidence value of their interpretation result. Using a confidence-based ranking scheme also requires that we interpret the ink with potentially all interpreters (to obtain their confidence measures), a computationally costly process. Our approach in using ink type prediction, as described in the previous chapters, suggests a viable, but not necessarily exclusive alternative to the use of confidence measures.

Chapter 7
Conclusion

We conclude with a list of possible future work and a summary of the main contributions of this thesis.

7.1 Future Work

The field of ink interpretation is exciting and filled with many challenges in every niche. While this thesis has tried to tackle a very narrow scope of improving interpretation accuracy within the domain of the classroom, invariably there are always improvements that can be made, and new hypotheses that need to be proven. We describe such future work in the following sections.

7.1.1 Creating a Public Interpreter API

We are designing a new architecture that will allow independently developed interpreters to be easily integrated into our dynamic dispatch interpreter. Figure 7-1 shows the current design of this new architecture.

Two interesting challenges are: (1) defining an application programming interface (API) for communicating ink samples and interpreted results between interpreters developed independently, and (2) integrating a top-level user-interface (UI) with any UIs that may accompany the new interpreters. We will want the API to work with new interpreters, but also with applications other than CLP. Moreover, integrating Ouyang and Davis' chemical diagram interpreter, for example, will require us to develop a UI that supports the real-time feedback and rendering that the program provides.

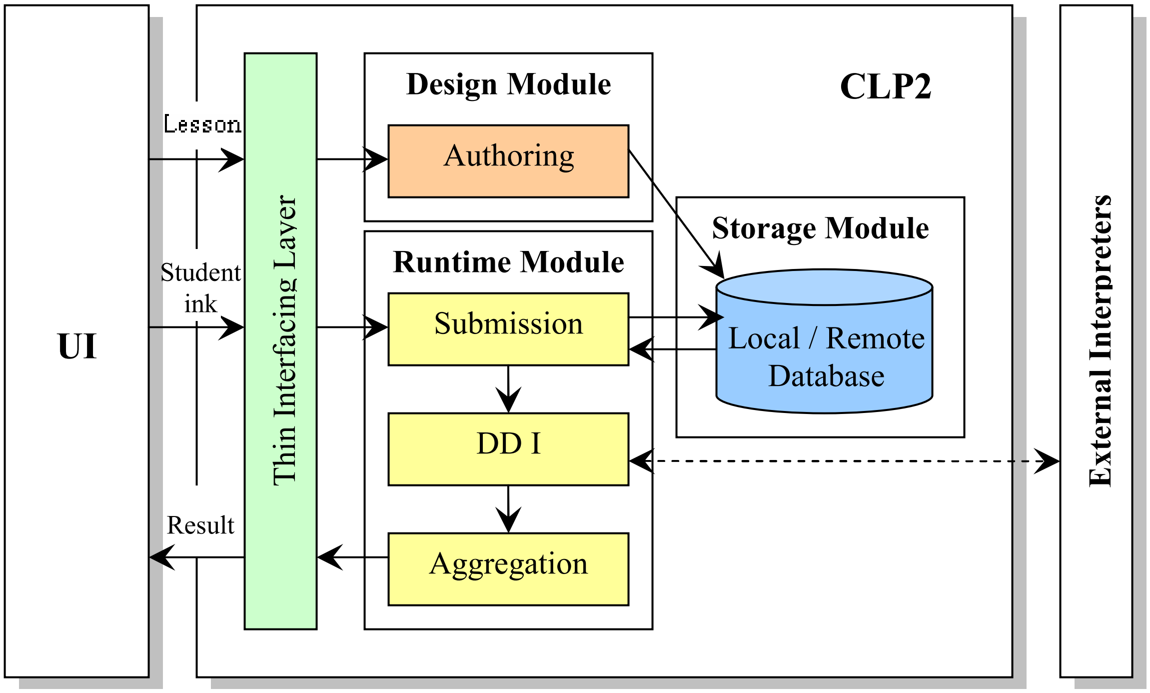


Figure 7-1. A schematic showing a new architecture to support integration into our dynamic dispatch interpreter (DDI) of independently developed ink interpreters.

7.1.2 Better Semantic Representation for Aggregation

Our current concept of semantic representation, i.e., ink interpreter output, follows from Rbeiz's work and presents a processed and summarized notion of the digital ink that is understood by our system [Rbeiz, 2006]. This semantic representation contains just enough information to allow rendering in printed form (if desired) and aggregation of similar ink samples that have the same representation; all dynamic information present in the digital ink such as the timing of strokes, positions, curvature, etc., that would exhibit high variance over many samples have not been included in this summary. This semantic representation has sufficed for our purposes in prototyping with CLP because the aggregator did not require more detailed information. As we support more complex aggregators, however, in various other applications and newer versions of CLP, we undoubtedly will want to include dynamic features for data-mining and clustering algorithms. Hence, we propose that the semantic representation output of the future system not only store the simplified summary of interpreted ink, but also any processed and unprocessed stroke data as auxiliary metadata to be used for aggregation algorithms and other applications.

7.1.3 Improving Interpretation Accuracy

Although we have shown that reasonably good interpretation can be achieved without the provision of *a priori* contextual information, we are still far from the 97% accuracy desired for users to feel comfortable [Giudice & Mottershead, 1999], [LaLomia, 1994]. Improving interpretation accuracy of digital ink has been the primary focus of this thesis and continues to be one of our goals. The more information we can provide with each ink sample, e.g., its question type, its writer, our expected answers to the question, etc., the better the resulting interpretation. We, thus, also are focusing on additional ways to supply our domain-specialized interpreters with better contextual information. With improved ink interpretation accuracy, we anticipate greater adoption in classrooms of systems such as CLP, which hold great promise for improving student learning.

7.2 Contributions

In this thesis, we presented a novel method for improving ink interpretation accuracy: using machine learning to predict expected ink types and using that type information to dynamically select appropriate specialized interpreters. We have shown that this approach does not rely on confidence measures of domain-specialized handwriting interpreters, and is in fact a more efficient alternative in terms of interpretation work that needs to be performed. In our approach of using machine learning, we extract many features from the dynamic ink strokes and use feature selection to generically improve prediction accuracy over the baseline for many experiment classes. The use of an SVM classifier consistently achieves high accuracies of greater than 80% for both *K*-fold and leave-one-out cross validation, even when there are up to five different classes to predict from. We also have deployed ink type prediction to be used as a module in an experimental CLP framework. Finally, we have demonstrated that our dynamic dispatch interpreters can achieve far more accurate interpretations (87% accuracy) than the default Microsoft interpreter (62%). Moreover, this accuracy level is close to that of our original INKv3 interpreter (89%), which required *a priori* type information to be provided.

References

[Alvarado & Davis, 2004] Alvarado, C., Oltmans, M. and Davis, R. A framework for multi-domain sketch recognition. *AAAI Spring Symposium on Sketch Understanding*, 2002, pp. 1-8.

[Anderson et al, 2004] Anderson, R., Anderson, R., Simon, B., Wolfman, S., VanDeGrift, T., and Yasuhara, K. Experiences with a tablet-pc-based lecture presentation system in computer science courses. In *Proceedings of SIGCSE '04*.

[Anderson et al, 2005] Anderson, R., Anderson, R., McDowell, L., and Simon, B. Use of Classroom Presenter in engineering courses. In *Proc of ASEE/IEEE Frontiers in Education Conference, 2005*.

[Anquetil & Lorette, 1995] Anquetil, E., and Lorette, G., On-Line Cursive Handwrittten Character Recognition Using Hidden Markov Models*, Traitement du Signal,* 12(6), pp. 575-583, 1995.

[Atallah, 1998] Atallah, M. J. (Editor), *Algorithms and Theory of Computation Handbook*, “Levenshtein Distance (13-5)”, CRC Press, 1998.

[Bellegarda et al, 1994] Bellegarda, E.J., Bellegarda, J.R., Nahamoo, D., and Nathan, K.S. A fast statistical mixture algorithm for on-line handwriting recognition in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 12, December 1994.

[Brakensiek et al, 2002] Brakensiek, A., Kosmala, A., and Rigoll, G. Evaluation of Confidence Measures for On-Line Handwriting Recognition. In *DAGM 2002*, Springer-Verlag Berlin Heidelberg, 2002, pp. 507-514.

[Breuel, 2002] Breuel, T.M. Representations and Metrics for Off-line Handwriting Segmentation. In *IEEE Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, 2002.

[Calhoun et al, 2002] Calhoun, C., Stahovich, T.F., Kurtoglu, T. and Kara, L.B. Recognizing multi-stroke symbols. 2002.

[Chen, 2006] Chen, J.I. Instructor Authoring Tool: A Step Towards Promoting Dynamic Lecture-style Classrooms. MIT EECS Master of Engineering thesis. May, 2006.

[Chevalier, 2007] Chevalier, K. Interpretation of Box and Pointer Diagrams in Classroom Learning Partner. MIT EECS Master of Engineering thesis. May, 2007.

[Dempster et al, 1977] Dempster, A.P., Laird, N.M. and Rubin, D.B. Maximum Likelihood from Incomplete Data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B, vol. 39, 1:1-38, 1977.

[Dunn, 1973] Dunn, J.C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics* 3: 32-57, 1973.

[Gamma et al, 1994] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*, Chapter 4: Structural Patterns, Addison-Wesley Professional, 1994.

[Gennari et al, 2005] Gennari, L., Kara, L.B., and Stahovich,T.F. Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Computers and Graphics* 29 (4), pp.547-562.

[Giudice & Mottershead, 1999] Giudice, J., Mottershead, B. Advanced Interfaces: Handwriting Recognition and the Human-Computer Interface. In *Speech Technology Magazine, Feb 1999.* [http://www.speechtechmag.com/Articles/ReadArticle.aspx?ArticleID=29380]

[Guyon et al, 2002] Guyon, I., Weston, J., Barnhill, S., Vapnik, V. Gene selection for cancer classification using support vector machines. In *Machine Learning*. 46:389-422.

[Hu et al, 1996] Hu, J., Brown, M., and Turin W. HMM Based On-Line Handwriting Recognition. In *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 18(10), October 1996.

[Hu et al, 2000] Hu, J., Lim, S.G., Brown. M. Writer independent on-line handwriting recognition using an HMM approach. In *Pattern Recognition Volume 33* (2000) pp. 133-147, January 1999.

[Kara & Stahovich, 2004] Kara, L.B. and Stahovich, T.F. Hierarchical parsing and recognition of hand-sketched diagrams. In *Proceedings of UIST 2004*, pp. 13-22.

[Keerthi et al, 2001] Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K., Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, 13(3), pp 637-649, 2001.

[Koile et al, 2007a] Koile, K., Chevalier, K., Rbeiz., M., Rogal, A., Singer, D., Sorensen, J., Smith, A., Tay, K.S., and Wu, K. Supporting Feedback and Assessment of Digital Ink Answers to In-Class Exercises. To appear in *Proceedings of IAAI 2007*, July, 2007.

[Koile et al, 2007b] Koile, K., Chevalier, K., Low, C., Pal, S., Rogal, A., Singer, D., Sorensen, J., Tay, K.S., and Wu, K. Supporting Pen-Based Classroom Interaction: New Findings and Functionality for Classroom Learning Partner. To appear in *Proceedings of First International Workshop on Pen-Based Learning Technologies 2007*, May, 2007a.

[Koile & Singer, 2005] Koile, K. and Singer, D. A. Development of a tablet-PC-based system to increase instructor-student classroom interactions and student learning. In *Impact of Pen-based Technology on Education: Vignettes, Evaluation, and Future Directions*. D. Berque, J. Prey, and R. Reed (eds). Purdue University Press. 2005.

[Koile & Singer, 2006] Koile, K. and Singer, D.A. Improving Learning in CS1 via Tablet-PC-Based In-Class Assessment. In *Proceedings of ICER 2006,* September 9-10, 2006, University of Kent, Canterbury, UK.

[Labahn et al, 2006] Labahn, G., MacLean, S., Marzouk, M., Rutherford, I. and Tausky, D. A Preliminary Report on the MathBrush Pen-Math System, In *Proceedings of Maple 2006 Conference*, pp. 162-178.

[LaLomia, 1994] LaLomia, M. J., User acceptance of handwritten recognition accuracy, In *Companion Proceedings of the CHI'94 Conference on Human Factors in Computing Systems,* p. 107. New York, ACM, 1994.

[LaViola & Zeleznik, 2005] LaViola, J. and Zeleznik, R., *MathPad: A System for the Creation and Exploration of Mathematical Sketches,* Brown University, 2005.

[Liu & Cai, 2003] Liu, Z. and Cai, J., *Handwriting Recognition, Soft Computing and Probabilistic Approaches*, Springer, 2003.

[Mahadevan & Nagabushnam, 1995] Mahadevan, U. and Nagabushnam, R.C. Gap Metrics for Word Separation in Handwritten Lines. In *Third International Conference on Document Analysis and Recognition (ICDAR'95) - Volume 1, p. 124*, 1995.

[Manmatha & Srimal, 1999] Manmatha, R. and Nitin, S. Scale Space Technique for Word Segmentation in Handwritten Documents. *Scale-Space Theories in Computer Vision: Second International Conference, Scale-Space'99, Corfu, Greece, September 1999. Proceedings*, Volume 1682/1999, Springer, 1999.

[O' Boyle et al, 2000] O' Boyle, C., Smyth, B., Geiselbrechtinger, F. An Automatic Configuration System for Handwriting Recognition Problems. *13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2000, June 19-22, 2000.*

[Oh et al, 2004] Oh, Y., Do, E.Y-L., and Gross, M.D. Intelligent Critiquing of Design Sketches. *AAAI Fall Symposium: Making Pen-Based Interaction Intelligent and Natural,* 2004.

[Ouyang & Davis, 2007] Ouyang, T. and Davis, R. Recognition of Hand-Drawn Chemical Diagrams. In *Proceedings of AAAI 2007*, July, 2007.

[Pitrelli & Perrone, 2003] Pitrelli, J. F., Perrone, M.P., Confidence-Scoring Post-Processing for Off-Line Handwritten-Character Recognition Verification. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR) 2003.* IEEE, 2003.

[Plamondon & Srihari, 2000] Plamondon, R, Srihari, S., On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. In *IEEE Trans. Pattern Anal. Machine Intelligence*., 22, Jan. 2000, pp. 63–85.

[Platt, 1998] Platt, J. Fast Training of Support Vector Machines using Sequential Minimal Optimization. *Advances in Kernel Methods - Support Vector Learning,* Schoelkopf, B., Burges, C., and Smola, A., eds., MIT Press, 1998.

[Quinlan, 1986] Quinlan, J.R. *Machine Learning*. 1986, 1, pp. 81-106.

[Quinlan, 1993] Quinlan, J.R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[Rbeiz, 2006] Rbeiz, M. A. Semantic Representation of Digital Ink in the Classroom Learning Partner. MIT EECS Master of Engineering thesis. May, 2006.

[Robnik-Sikonja & Kononenko, 1997] Robnik-Sikonja, M. and Kononenko, I. An adaptation of Relief for attribute estimation in regression. In *Fourteenth International Conference on Machine Learning*, pp. 296-304, 1997.

[Rubine, 1991] Rubine, D. Specifying gestures by example. *Computer Graphics,* 25(4), pp. 329-337, 1991.

[Seni & Cohen, 1994] Seni, G., and Cohen, E. External word segmentation of off-line handwritten text lines. In *Pattern Recognition Volume 27 Issue 1,* pp. 41-52, January 1994.

[Sezgin & Davis, 2005] Sezgin, T.M. and Davis, R. HMM-based efficient sketch recognition. In *Proceedings of IUI '05*, pp. 281-283.

[Shilman et al, 2002] Shilman, M., Pasula, H., Russell, S. and Newman, R. Statistical visual language models for ink parsing. In *AAAI Spring Symposium on Sketch Understanding*, 2002.

[Shilman et al, 2004] Shilman, M., Viola, P., and Chellapilla, K. Recognition and Grouping of Handwritten Text In Diagrams and Equations, *IEEE 9th Intl Workshop on Frontiers in Handwriting Recog,* 2004.

[Smith, 2006] Smith, A. C. Aggregation of Student Answers in a Classroom Setting. MIT EECS Master of Engineering thesis. August, 2006.

[Srihari & Keubert, 1997] Srihari, S. and Keubert, E.J. Integration of Handwritten Address Interpretation Technology into the United States Postal Service Remote Computer Reader System. In *Proceedings of the Fourth International Conference in Document Analysis and Recognition,* Vol. 2*,* pp. 892-896, August, 1997.

[Wang et al, 2005] Wang, J., Neskovic, P., and Cooper, L.N. A probabilistic model for cursive handwriting recognition using spatial context. In *IEEE, Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05).* IEEE International Conference*,* 2005.

[Witten & Frank, 2005] Witten, I. H., Frank, E. *Data Mining: Practical machine learning tools and techniques, 2nd edition,* Morgan Kaufmann, San Francisco, 2005.

[Wu, 2008] Wu, K. D. Interpretation and Aggregation of Marks in Classroom Learning Partner. MIT EECS Master of Engineering thesis. February, 2008.

[Yasuda et al, 2000] Yasuda, H., Takahashi, K., and Matsumoto, T. A Discrete HMM for Online Handwriting. In *International Journal of Pattern Recognition and Artificial Intelligence,* Vol. 14, No. 5, pp. 675-688, 2000.

Appendix A

Representative Examples

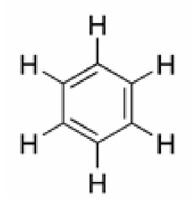
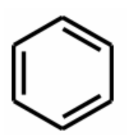
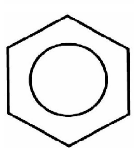
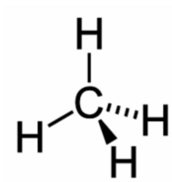
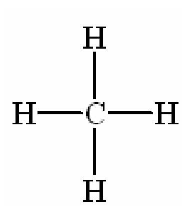
Table A.1: List of 181 representative examples sorted by their Representative ID (Rep ID) number, showing the example string/diagram shown to students, and the expected semantic representation (simplified from XML form)

RepID	Example String/Diagram	Simplified Semantic Representation
1	#f	#f
2	#t	#t
3	false	False
4	true	True
5	π	PI
6	Π	PI
7	Ω	OMEGA
8	$\frac{\mathbf{22}}{\mathbf{7}}$	22/7
10	0	0
11	1	1
12	2	2
13	5	5
14	6	6
15	7	7
16	9	9
17	10	10
18	11	11
19	50	50
20	55	55
21	100	100
22	101	101
30	0.1	0.1
31	2.71828	2.71828
32	123.45	123.45

Rep ID	Example String/Diagram	Simplified Semantic Representation
33	3.14	3.14
34	3.14159	3.14159
35	19.95	19.95
36	.007	.007
50	O	O
51	I	I
52	l	l
53	/	/
54	Z	Z
55	S	S
56	G	G
57	>	>
58	q	q
59	g	g
60	lo	lo
61	II	II
62	ll	ll
63	//	//
64	/l	/l
65	so	so
66	ss	ss
67	loo	loo
68	IOI	IOI
69	lol	lol
100	'done	'done
110	double-tree	double-tree
120	cons	cons
121	error	error
122	list	list
123	nil	nil
124	quote	quote
150	O(n)	O(n)
151	pi	pi
170	benzene	benzene
171	methane	methane
172	phenol	phenol
173	carbolic acid	carbolic acid
174	alanine	alanine
175	acetic acid	acetic acid
176	ethanoic acid	ethanoic acid
177	proton	proton
178	electron	electron
179	neutron	neutron

Rep ID	Example String/Diagram	Simplified Semantic Representation
180	serine	serine
181	phenylalanine	phenylalanine
190	Ala	Ala
191	Ser	Ser
192	Phe	Phe
200	[1 2 3]	[1,2,3]
201	1, 3, 6, 10, 15	[1,3,6,10,15]
202	2 30 400 5000	[2,30,400,5000]
203	80, 90, 100, 110	[80,90,100,110]
220	defg abc	[d,e,f,g,a,b,c]
221	A B E F G K L H C I J D	[A,B,E,F,G,K,L,H,C,I,J,D]
222	a, b, c, d, e, f, g, h, i, j, k, l	[a,b,c,d,e,f,g,h,i,j,k,l]
223	#, #, # -> #	[#,#,#,->,#]
224	g, ng, ing, ring	[g,ng,ing,ring]
240	number number	[number,number]
241	boolean -> string	[boolean,->,string]
243	lecture & recitation	[lecture,recitation]
244	nbr, nbr, nbr -> nbr	[nbr,nbr,nbr,->,nbr]
245	reading, talking, listening	[reading,talking,listening]
300	152 kJ	[152,kJ]
301	47 ohms	[47,ohms]
302	1 kg	[1,kg]
303	1.79 g/L	[1.79,g/L]
304	2.9 lbs	[2.9,lbs]
305	3 bonds	[3,bonds]
306	32 F	[32,F]
307	273.15 K	[273.15,K]
320	- 11 N	[-,11,N]
321	- 23 mm	[-,23,mm]
330	$ 100.00	[$,100.00]
340	47 Ω	[47,OMEGA]
350	37 ^{o}C	[37,DEG,C]
351	78.1 gmol^{-1}	[78.1,gmol,^-1]
352	3.53 Wm^{-1}K^{-1}	[3.53,Wm,^-1,K,^-1]
353	0.89 cm^{2}	[0.89,cm,^2]
380	x + y = z	[x,+,y,=,z]
381	a = b + c	[a,=,b,+,c]
382	10 + 14 = 24	[10,+,14,=,24]
383	x = 23 y - 77	[x,=,23,y,-,77]
384	x y z = 503	[x,y,z,=,503]
385	$y = x^2$	[y,=,x,^2]
386	$x^3 + 10\,x^2 - x + 15 = 0$	[x,^3,10,x,^2,-,x,+,15,=,0]
400	n^2	[n,^2]

Rep ID	Example String/Diagram	Simplified Semantic Representation
401	n^3	[n,^3]
402	x^2	[x,^2]
403	e^x	[e,^x]
404	O_2	[O,_2]
405	SO_4^{2-}	[S,O,_4,^2-]
406	10^{100}	[10,^100]
407	a_1	[a,_1]
408	b_2	[b,_2]
409	x_1y_1	[x,_1,y,_1]
410	x_2y_2	[x,_2,y,_2]
411	$6 \text{ x } 10^{23}$	[6,x,10,^23]
430	C_6H_6	[C,_6,H,_6]
431	CH_4	[C,H,_4]
432	C_6H_5OH	[C,_6,H,_5,O,H]
433	$HO_2CCH(NH_2)CH_3$	[H,O,_2,C,C,H,(,N,H,_2,),C,H,_3]
434	CH_3COOH	[C,H,_3,C,O,O,H]
450	$C + O_2 = CO_2$	[C,+,O,_2,=,C,O,_2]
451	$2\ H_2 + O_2 = 2\ H_2O$	[2,H,_2,+,O,_2,=,2,H,_2,O]
470	$1\ s^1$	[1,s,^1]
471	$1\ s^2\ 2\ s^1$	[1,s,^2,2,s,^1]
472	$1\ s^2\ 2\ s^2\ 2\ p^3$	[1,s,^2,2,s,^2,2,p,^3]
473	$1\ s^2\ 2\ s^2\ 2\ p^6\ 3\ s^1$	[1,s,^2,2,s,^2,2,p,^6,3,s,^1]
474	$[\ Kr\ ]\ 4\ d^{10}$	[[,Kr,],4,d,^10]
475	$[\ Ar\ ]\ 4\ s^2\ 3\ d^5$	[[,Ar,],4,s,^2,3,d,^5]
476	$[\ Xe\ ]\ 6\ s^1\ 4\ f^{14}\ 5\ d^{10}$	[[,Xe,],6,s,^1,4,f,^14,5,d,^10]
477	$He : 1\ s^2$	[He,:,1,s,^2]
478	$F : 1\ s^2\ 2\ s^2\ 2\ p^5$	[F,:,1,s,^2,2,s,^2,2,p,^5]
479	$F^- : 1\ s^2\ 2\ s^2\ 2\ p^6$	[F,^-,:,1,s,^2,2,s,^2,2,p,^6]
480	$Ca : [\ Ar\ ]\ 4\ s^2$	[Ca,:,[,Ar,],4,s,^2]
481	$Ca^{2+} : [\ Ar\ ]$	[Ca,^2+,:,[,Ar,]]
482	$Pb : [\ Xe\ ]\ 4\ f^{14}\ 5\ d^{10}\ 6\ s^2\ 6\ p^2$	[Pb,:,[,Xe,],4,f,^14,5,d,^10,6,s,^2,6,p,^2]
483	$Pb^{2+} : [\ Xe\ ]\ 4\ f^{14}\ 5\ d^{10}\ 6\ s^2$	[Pb,^2+,:,[,Xe,],4,f,^14,5,d,^10,6,s,^2]
500	(a b)	(a b)
501	(caar seq)	(caar seq)
502	(cdddr exp)	(cdddr exp)
503	(eq? id1 id2)	(eq? id1 id2)
504	(map double-tree tree)	(map double-tree tree)
505	(/ 2 tree)	(/ 2 tree)
506	(a 7)	(a 7)
507	(define x 3)	(define x 3)
508	(1 2)	(1 2)
509	(* 1 2)	(* 1 2)
700	(cons (cdar seq) (cddr seq))	(cons (cdar seq) (cddr seq))

Rep ID	Example String/Diagram	Simplified Semantic Representation
701	(first (second exp))	(first (second exp))
702	(car (quote (quote a)))	(car (quote (quote a)))
703	(set-cdr! (last-pair x) x)	(set-cdr! (last-pair x) x)
704	(lambda (new) (set! x new))	(lambda (new) (set! x new))
705	(element-of-tree? x (left-branch tree))	(element-of-tree? x (left-branch tree))
706	(define (list->stream l) (cons-stream (car l) (list->stream (cdr l)))	(define (list->stream l) (cons-stream (car l) (list->stream (cdr l)))
707	(lambda (a b) (+a b))	(lambda (a b) (+a b))
708	(list (m-eval init env))	(list (m-eval init env))
709	(define ints (cons-stream 1 (add-streams ints ones)))	(define ints (cons-stream 1 (add-streams ints ones)))
710	(cons (cons x (+ 1 (+ 1 (seq-length seq)))	(cons (cons x (+ 1 (+ 1 (seq-length seq)))
720	(foo bar)	(foo bar)
721	((((foo baz))) bar)	((((foo baz))) bar)
1000		BENZENE
1001		BENZENE
1002		BENZENE
1003		METHANE
1004		METHANE

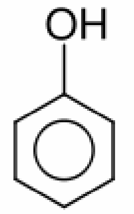
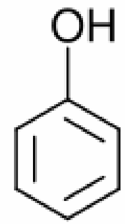
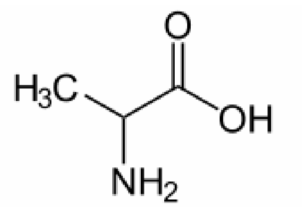
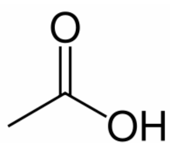
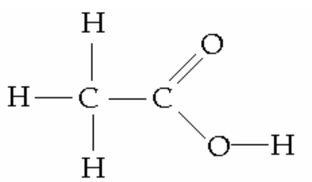
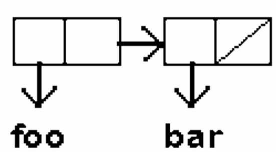
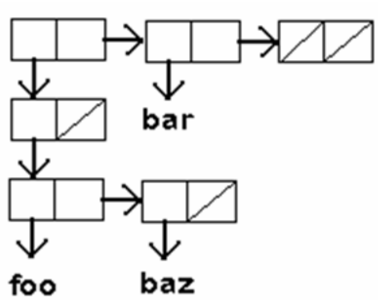
Rep ID	Example String/Diagram	Simplified Semantic Representation
1005		PHENOL
1006		PHENOL
1007		ALANINE
1008		ETHANOIC_ACID
1009		ETHANOIC_ACID
1100		(foo bar)
1101		((((foo baz))) bar)

Table A.2: List of 181 representative examples sorted by their Representative ID (Rep ID) number, showing the expected type and sample student (the author's) ink.

RepID	Expected Type	Ink Sample
1	True-False	#f
2	True-False	#t
3	True-False	false
4	True-False	true
5	Symbol	π
6	Symbol	π
7	Symbol	Ω
8	Number Fraction	$\frac{22}{7}$
10	Number	0
11	Number	1
12	Number	2
13	Number	5
14	Number	6
15	Number	7
16	Number	9
17	Number	10
18	Number	11
19	Number	50
20	Number	55
21	Number	100
22	Number	101
30	Decimal Number	0.1
31	Decimal Number	2.71828
32	Decimal Number	123.45
33	Decimal Number	3.14
34	Decimal Number	3.14159
35	Decimal Number	19.95
36	Decimal Number	.007
50	String	O
51	String	1
52	String	1
53	String	/
54	String	Z
55	String	S
56	String	G
57	String	7
58	String	9
59	String	9
60	String	1o
61	String	11
62	String	11
63	String	//
64	String	/1

RepID	Expected Type	Ink Sample
65	String	So
66	String	SS
67	String	loo
68	String	101
69	String	lol
100	Quoted String	'done
110	Variable String	double-tree
120	Scheme String	cons
121	Scheme String	error
122	Scheme String	list
123	Scheme String	nil
124	Scheme String	quote
150	Math String	O(n)
151	Math String	pi
170	Chemistry String	benzene
171	Chemistry String	methane
172	Chemistry String	phenol
173	Chemistry String	carbolic acid
174	Chemistry String	alanine
175	Chemistry String	acetic acid
176	Chemistry String	ethanoic acid
177	Chemistry String	proton
178	Chemistry String	electron

RepID	Expected Type	Ink Sample
179	Chemistry String	neutron
180	Chemistry String	serine
181	Chemistry String	phenylalanine
190	Chemistry String	Ala
191	Chemistry String	Ser
192	Chemistry String	Phe
200	Number Sequence	[1 2 3]
201	Number Sequence	1, 3, 6, 10, 15
202	Number Sequence	2 30 400 5000
203	Number Sequence	80, 90, 100, 110
220	Single Char Sequence	defg abc
221	Single Char Sequence	A B E F G K L H C I J
222	Single Char Sequence	a, b, c, d, e, f, g, h, i, j, k, l
223	Single Char Sequence	#, #, # → #
224	String Sequence	g, ng, ing, ring
240	String Sequence	number number
241	String Sequence	boolean → string
243	String Sequence	lecture & recitation
244	String Sequence	nbr, nbr, nbr → nbr
245	String Sequence	reading, talking, listening

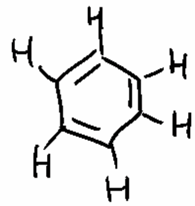
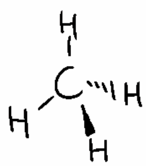
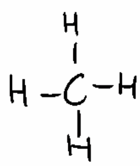
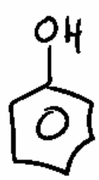
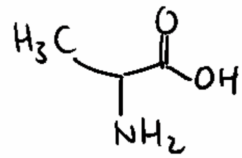
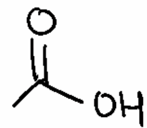
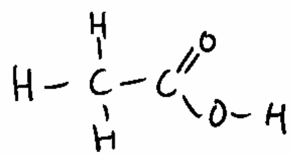
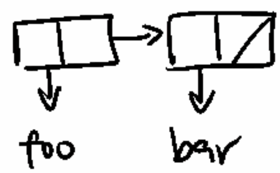
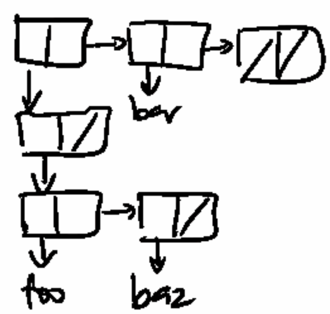
RepID	Expected Type	Ink Sample
300	Chemistry Sequence	152 kJ
301	Chemistry Sequence	47 ohms
302	Chemistry Sequence	1 kg
303	Chemistry Sequence	1.79 g/L
304	Chemistry Sequence	2.9 lbs
305	Chemistry Sequence	3 bonds
306	Chemistry Sequence	32 F
307	Chemistry Sequence	273.15 K
320	Chemistry Sequence	- 11 N
321	Chemistry Sequence	- 23 mm
330	Chemistry Sequence	$ 10000
340	Chemistry Sequence	47 Ω
350	Chemistry Sequence	37 °C
351	Chemistry Sequence	78.1 $gmol^{-1}$
352	Chemistry Sequence	3.53 $Wm^{-1}K^{-1}$
353	Chemistry Sequence	0.89 cm^2
380	Chemistry Sequence	$x + y = z$
381	Chemistry Sequence	$a = b + c$
382	Chemistry Sequence	$10 + 14 = 24$
383	Chemistry Sequence	$x = 23y - 77$
384	Chemistry Sequence	$xyz = 503$
385	Chemistry Sequence	$y = x^2$
386	Chemistry Sequence	$x^3 + 10x^2 - x + 15 = 0$

RepID	Expected Type	Ink Sample
400	Chemistry Sequence	n^2
401	Chemistry Sequence	n^3
402	Chemistry Sequence	x^2
403	Chemistry Sequence	e^x
404	Chemistry Sequence	O_2
405	Chemistry Sequence	SO_4^{2-}
406	Chemistry Sequence	10^{100}
407	Chemistry Sequence	a_1
408	Chemistry Sequence	b_2
409	Chemistry Sequence	$x_1 y_1$
410	Chemistry Sequence	$x_2 y_2$
411	Chemistry Sequence	6×10^{23}
430	Chemistry Sequence	C_6H_6
431	Chemistry Sequence	CH_4
432	Chemistry Sequence	C_6H_5OH
433	Chemistry Sequence	$HO_2CCH(NH_2)CH_3$
434	Chemistry Sequence	CH_3COOH
450	Chemistry Sequence	$C + O_2 = CO_2$
451	Chemistry Sequence	$2H_2 + O_2 = 2H_2O$
470	Chemistry Sequence	$1s^1$
471	Chemistry Sequence	$1s^2 2s^1$
472	Chemistry Sequence	$1s^2 2s^2 2p^3$
473	Chemistry Sequence	$1s^2 2s^2 2p^6 3s^1$

RepID	Expected Type	Ink Sample
474	Chemistry Sequence	$[Kr]\ 4d^{10}$
475	Chemistry Sequence	$[Ar]\ 4s^2\ 3d^5$
476	Chemistry Sequence	$[Xe]\ 6s^1\ 4f^{14}\ 5d^{10}$
477	Chemistry Sequence	$He: 1s^2$
478	Chemistry Sequence	$F: 1s^2\ 2s^2\ 2p^5$
479	Chemistry Sequence	$F^-: 1s^2\ 2s^2\ 2p^6$
480	Chemistry Sequence	$Ca: [Ar]\ 4s^2$
481	Chemistry Sequence	$Ca^{2+}: [Ar]$
482	Chemistry Sequence	$Pb: [Xe]\ 4f^{14}\ 5d^{10}\ 6s^2$
483	Chemistry Sequence	$Pb^{2+}: [Xe]\ 4f^{14}\ 5d^{10}\ 6s^2$

RepID	Expected Type	Ink Sample
500	Flat Scheme Expression	(a b)
501	Flat Scheme Expression	(caar seq)
502	Flat Scheme Expression	(cdddr exp)
503	Flat Scheme Expression	(eq? id1 id2)
504	Flat Scheme Expression	(map double-tree tree)
505	Flat Scheme Expression	(/ 2 tree)
506	Flat Scheme Expression	(a 7)
507	Flat Scheme Expression	(define x 3)
508	Flat Scheme Expression	(1 2)
509	Flat Scheme Expression	(* 1 2)

RepID	Expected Type	Ink Sample
700	Nested Scheme Expression	(cons (cdar seq) (cddr seq))
701	Nested Scheme Expression	(first (second exp))
702	Nested Scheme Expression	(car (quote (quote a)))
703	Nested Scheme Expression	(set-cdr! (last-pair x) x)
704	Nested Scheme Expression	(lambda (new) (set! x new))
705	Nested Scheme Expression	(element-of-tree? x (left-branch tree))
706	Nested Scheme Expression	(define (list→stream l) (cons-stream (car l) (list→stream (cdr l))))
707	Nested Scheme Expression	(lambda (a b) (+ a b))
708	Nested Scheme Expression	(list (m-eval init env))
709	Nested Scheme Expression	(define ints (cons-stream 1 (add-streams ints ones)))
710	Nested Scheme Expression	(cons (cons x (+ 1 (+ 1 (seq-length seq))))

RepID	Expected Type	Ink Sample
720	Flat Scheme Expression	(foo bar)
721	Nested Scheme Expression	((((foo baz))) bar)
1000	Chemistry Diagram	
1001	Chemistry Diagram	
1002	Chemistry Diagram	
1003	Chemistry Diagram	
1004	Chemistry Diagram	
1005	Chemistry Diagram	
1006	Chemistry Diagram	
1007	Chemistry Diagram	
1008	Chemistry Diagram	
1009	Chemistry Diagram	
1100	Box-and-Pointer Diagram	
1101	Box-and-Pointer Diagram	

Appendix B

Representation Results

Table B.1: Representation results for our different interpreters on the same data set grouped by the different representative examples in the field of introductory computer science

RepID	Semantic Representation	INKv3	INKv1	NDDI	NCVI-10	NCVI-4	Microsoft
1	#f	86.11	86.11	86.11	86.11	86.11	22.22
2	#t	100.00	100.00	100.00	100.00	100.00	62.50
3	false	100.00	100.00	100.00	100.00	100.00	87.50
4	true	100.00	100.00	100.00	100.00	100.00	93.75
10	0	100.00	100.00	45.45	45.45	45.45	9.09
11	1	100.00	100.00	36.36	36.36	36.36	0.00
12	2	100.00	100.00	100.00	100.00	100.00	9.09
13	5	100.00	100.00	100.00	100.00	100.00	9.09
14	6	100.00	100.00	100.00	100.00	100.00	0.00
15	7	100.00	100.00	100.00	100.00	100.00	0.00
16	9	100.00	100.00	90.91	90.91	90.91	0.00
17	10	100.00	100.00	100.00	100.00	100.00	27.27
18	11	95.45	95.45	95.45	95.45	95.45	18.18
19	50	90.91	90.91	90.91	81.82	90.91	54.55
20	55	100.00	100.00	100.00	100.00	100.00	63.64
21	100	100.00	100.00	100.00	100.00	100.00	45.45
22	101	96.97	96.97	96.97	96.97	96.97	51.52
50	O	72.73	72.73	72.73	72.73	72.73	18.18
51	I	63.64	63.64	63.64	63.64	63.64	0.00
52	l	8.33	8.33	8.33	8.33	8.33	0.00
53	/	81.82	81.82	81.82	81.82	81.82	0.00
54	Z	81.82	81.82	81.82	81.82	81.82	0.00
55	S	100.00	100.00	100.00	100.00	100.00	0.00
56	G	90.91	90.91	90.91	90.91	90.91	9.09
57	>	90.91	90.91	90.91	90.91	90.91	0.00
58	q	63.64	63.64	63.64	63.64	63.64	0.00
59	g	100.00	100.00	100.00	100.00	100.00	0.00

RepID	Semantic Representation	INKv3	INKv1	NDDI	NCVI-10	NCVI-4	Microsoft
60	lo	63.64	63.64	63.64	63.64	63.64	22.73
61	II	27.27	27.27	27.27	27.27	27.27	4.55
62	ll	0.00	0.00	0.00	0.00	0.00	0.00
63	//	0.00	0.00	0.00	0.00	0.00	0.00
64	/l	0.00	0.00	0.00	0.00	0.00	0.00
65	so	100.00	100.00	100.00	100.00	100.00	18.18
66	ss	100.00	100.00	100.00	100.00	100.00	9.09
67	loo	33.33	33.33	33.33	33.33	33.33	24.24
68	IOI	30.30	30.30	30.30	30.30	30.30	15.15
69	lol	30.30	30.30	30.30	30.30	30.30	24.24
70	IO	45.00	45.00	45.00	45.00	45.00	10.00
100	'done	97.14	97.14	97.14	90.00	90.00	82.86
110	double-tree	98.30	98.30	98.30	98.30	98.30	93.18
120	cons	100.00	100.00	100.00	100.00	100.00	87.50
121	error	100.00	100.00	100.00	100.00	100.00	91.25
122	list	100.00	100.00	100.00	100.00	100.00	57.81
123	nil	100.00	100.00	100.00	100.00	100.00	83.33
124	quote	100.00	100.00	95.00	90.00	95.00	92.50
150	O(n)	60.94	60.94	60.94	60.94	60.94	40.63
200	[1,2,3]	68.75	61.61	68.75	66.96	65.18	60.71
201	[1,3,6,10,15]	97.60	83.65	92.31	90.87	91.35	86.54
202	[2,30,400,5000]	98.89	95.56	98.89	98.89	93.33	86.67
203	[80,90,100,110]	97.78	91.11	95.56	95.56	85.56	75.56
220	[d,e,f,g,a,b,c]	87.92	64.17	78.33	76.25	65.83	60.83
221	[A,B,E,F,G,K,L,H,C,I,J,D]	92.00	53.87	92.00	88.53	87.47	48.80
222	[a,b,c,d,e,f,g,h,i,j,k,l]	84.27	60.00	84.27	84.27	80.80	55.20
223	[#,#,#,->,#]	67.78	41.11	67.78	59.44	63.33	37.22
224	[g,ng,ing,ring]	85.56	78.89	85.56	81.11	76.67	58.89
240	[number,number]	99.11	99.11	99.11	97.78	93.33	94.67
241	[boolean,->,string]	85.26	90.88	82.46	82.46	78.60	80.00
243	[lecture,recitation]	89.67	96.00	88.67	87.67	87.67	92.00
244	[nbr,nbr,nbr,->,nbr]	71.15	66.92	71.15	71.15	69.23	66.54
245	[reading,talking,listening]	91.01	96.30	91.01	91.53	84.39	90.74
500	(a b)	77.27	77.27	77.27	77.27	77.27	72.73
501	(caar seq)	73.33	73.33	73.33	73.33	76.67	68.89
502	(cdddr exp)	81.00	81.00	82.00	84.00	82.00	93.00
503	(eq? id1 id2)	71.82	71.82	71.82	73.64	75.45	72.73
504	(map double-tree tree)	96.50	96.50	96.50	96.50	97.00	93.50
505	(/ 2 tree)	82.50	82.50	82.50	83.75	83.75	78.75
506	(a 7)	95.00	95.00	97.50	97.50	97.50	87.50
507	(define x 3)	92.00	92.00	92.00	92.00	92.00	86.00

RepID	Semantic Representation	INKv3	INKv1	NDDI	NCVI-10	NCVI-4	Microsoft
508	(1 2)	100.00	100.00	100.00	100.00	100.00	97.50
509	(* 1 2)	68.00	68.00	68.00	66.00	68.00	58.00
510	(if test #f #t)	83.33	83.33	83.33	83.33	83.33	79.17
700	(cons (cdar seq) (cddr seq))	83.75	83.75	83.33	83.33	83.33	71.25
701	(first (second exp))	97.22	97.22	97.22	97.22	97.22	94.44
702	(car (quote (quote a)))	92.50	92.50	92.50	91.50	91.50	80.50
703	(set-cdr! (last-pair x) x)	91.30	91.30	91.30	91.30	91.30	83.09
704	(lambda (new) (set! x new))	96.14	96.14	96.14	96.14	96.14	91.79
705	(element-of-tree? x (left-branch tree))	94.14	94.14	94.14	94.14	94.14	90.43
706	(define (list->stream l) (cons-stream (car l) (list->stream (cdr l)))	81.31	81.31	81.31	81.31	81.31	77.95
707	(lambda (a b) (+a b))	88.89	88.89	88.89	88.89	88.89	90.20
708	(list (m-eval init env))	80.95	80.95	80.95	80.95	80.95	76.72
709	(define ints (cons-stream 1 (add-streams ints ones)))	81.63	81.63	81.63	81.41	81.41	80.73
710	(cons (cons x (+ 1 (+ 1 (seq-length seq)))	77.45	77.45	77.45	77.45	77.12	69.93
711	((p 'SET-CAR!) new-car)	77.55	77.55	78.23	78.23	78.23	79.59
712	(define x (let ((two '(2))) (list (cons 1 two) (list 1) two)))	78.11	78.11	78.11	78.11	78.11	76.23
	Total (Equal Weight)	**81.82**	**80.18**	**80.52**	**78.53**	**78.44**	**51.00**

Appendix C
Features Considered

This section describes the features we considered in greater detail than what we have already listed in Table 4.2.

Table C.1: Features we considered, their descriptions and our hypotheses

No.	Name	Description and Hypothesis
F1	Total number of strokes	This feature counts the total number of strokes (from pen-down to pen-up) an ink sample has, a useful metric for generally distinguishing simple and complex ink samples.
F2	Total number of positive inter-stroke adjacent spacing	Inter-stroke adjacent spacing is the distance between two adjacent strokes in an ink sample. This feature counts the number of such positive spacing and hence allows differentiation of short or diagrammatic ink samples from long sequence-like ones.
F3	Sample height span	The total height of an ink sample measured in ink space units. Diagrams are generally taller than regular text.
F4	Sample width span	The total width of an ink sample measured in ink space units. Sequences and Scheme expressions are generally longer than numbers.
F5	Sample width-height ratio	The ratio of an ink sample's total width to total height. This feature is useful for telling ink samples that are taller than wide or vice versa, and has greater importance since we do not do scale normalization. Diagrams in our domain are generally square-shaped while text is flat.
F6	Stroke area density of points	This feature computes the density of pen-tip points over an ink stroke's bounding box, effectively measuring the amount of ink for each stroke. This density is helpful in differentiating different types of strokes for diagrams or characters.
F7	Stroke horizontal density of points	This feature computes the density of pen-tip points over the horizontal width of each ink stroke, effectively measuring the amount of ink for each unit of width of the stroke. This density is helpful in differentiating vertical and horizontal strokes in text or diagrams.
F8	Stroke heights	The height of each ink stroke measured in ink space units. Useful for telling tall characters like 'l', 'f', 'g', etc. from short ones like '-', ',' or 'a'.

No.	Name	Description and Hypothesis
F9	Stroke widths	The width of each ink stroke measured in ink space units. Useful for telling wide characters like ‘w’, ‘z’, ‘—‘, etc. from narrow ones like ‘/’, ‘I’, or ‘!’.
F10	Stroke lengths	The length of each ink stroke measured in ink space units. Useful for telling long characters like ‘|’, ‘—‘, ‘}’, etc. from short ones like ‘,’, ‘c’, or ‘^’.
F11	Stroke points count	The amount of ink of each ink stroke. Useful for telling diagrams or dense complex characters like ‘*’, ‘&’, ‘B’, etc. from sparse or simple ones like ‘s’, ‘(‘ or ‘o’.
F12	Stroke adjacent spacing	The inter-stroke spacing distance between each pair of adjacent strokes measured in ink space units. Useful for differentiating sequences and strings from single characters and numbers.
F13	Stroke adjacent spacing differentials	Once all inter-stroke adjacent spacing distance is calculated for an ink sample, the distances are sorted in ascending order. A first order differential on this discrete number sequence is then computed by taking the differences between each adjacent element of the spacing sequence. This differential ‘profile’ computed is a useful feature that tells sequences apart from strings because of the wider inter-word gaps that inter-character gaps in sequences.
F14	Number of stroke intersections	The total number of intersections a stroke has with itself and also with other strokes. Useful for differentiating characters that have strokes that intersect like ‘+’, ‘x’, ‘#’, etc. from others like ‘v’, ‘s’, or ‘=’. Also useful for differentiating diagrams and text.
F15	Stroke angles	The angle of orientation for each part of an ink stroke measured in radians. Useful for telling certain characters that slant and curve apart from others.
F16	Stroke speeds	The ratio of stroke length to the number of pen-tip points (amount of ink) for each stroke. Useful for telling strokes that were written/drawn faster than others, e.g., diagrams are generally drawn faster than printed text.
F17	Similarity of a stroke to a number	There are many ambiguous strokes that can look like numbers or Roman alphabets and thus it was important to differentiate these two if we could. Template matching [Ouyang & Davis, 2007] is a popular feature generator for such single character comparisons to a pre-computed template dictionary. We opted for a simple approximation here, however: we chose to use an unbiased and untrained Microsoft recognizer to interpret each ink stroke. We count the proportion, within the interval of [0, 1]. of the ink sample’s strokes that had numbers returned by the recognizer and use the proportion as a feature.

Appendix D
Feature Importance

This section includes three figures of the individual monochrome grids highlighting feature importance making up the visualization shown in Figure 4-3 for non-color printing. In order, the figures show summaries of feature importance for three different feature selection algorithms: SVM-Weight, GainRatio and InfoGain. The darker a cell in the diagrams, the more important a feature is. (Note that this is different from Figure 4-3 which presents all three grids as color channels, with brighter colors denoting greater importance.)

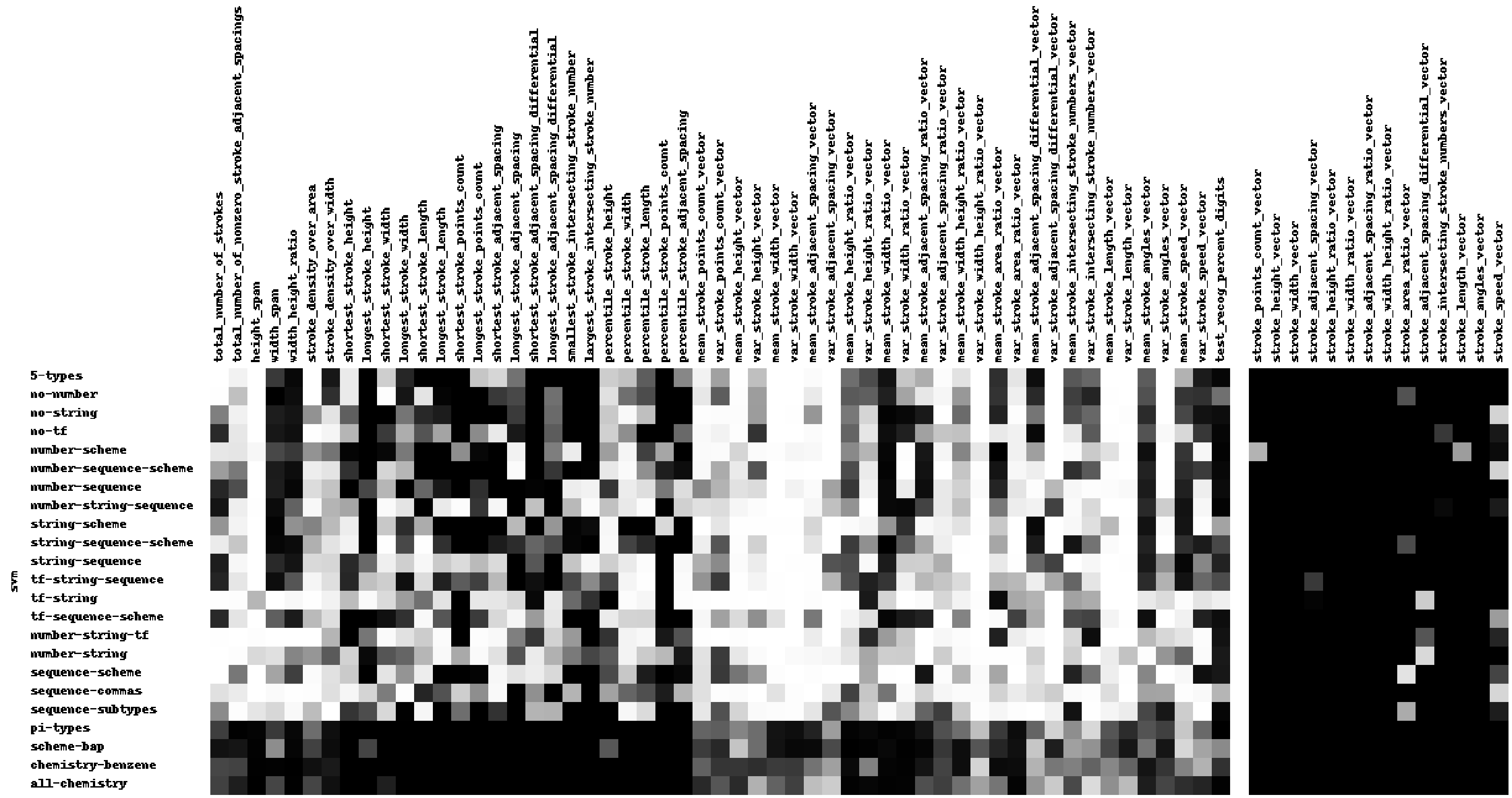


Figure D-1. This visualization summarizes the work of the SVM Weight feature selection algorithm, highlighting the important features among all features that we extracted with darker cells.

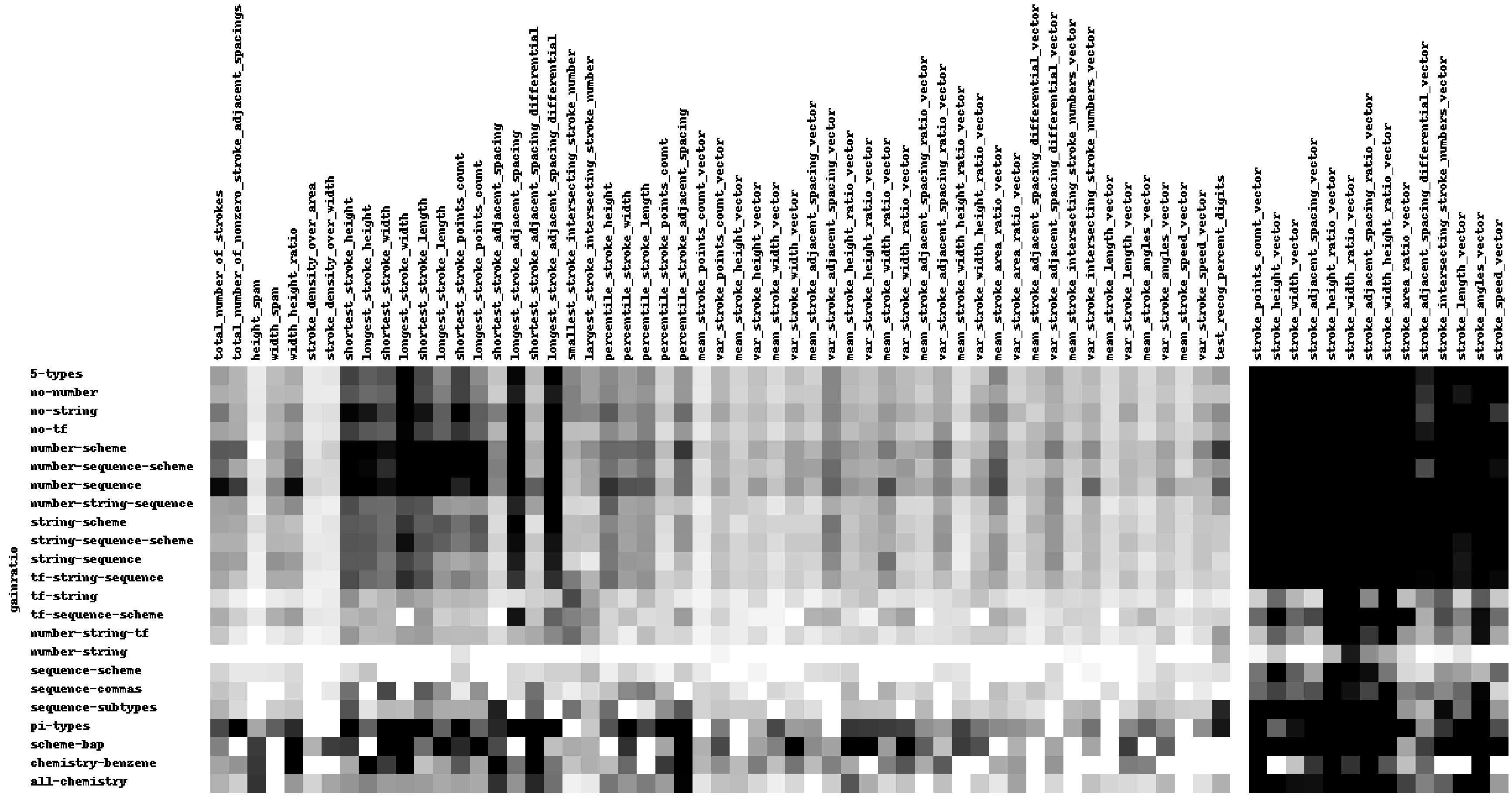


Figure D-2. This visualization summarizes the work of the GainRatio feature selection algorithm, highlighting the important features among all features that we extracted with darker cells.

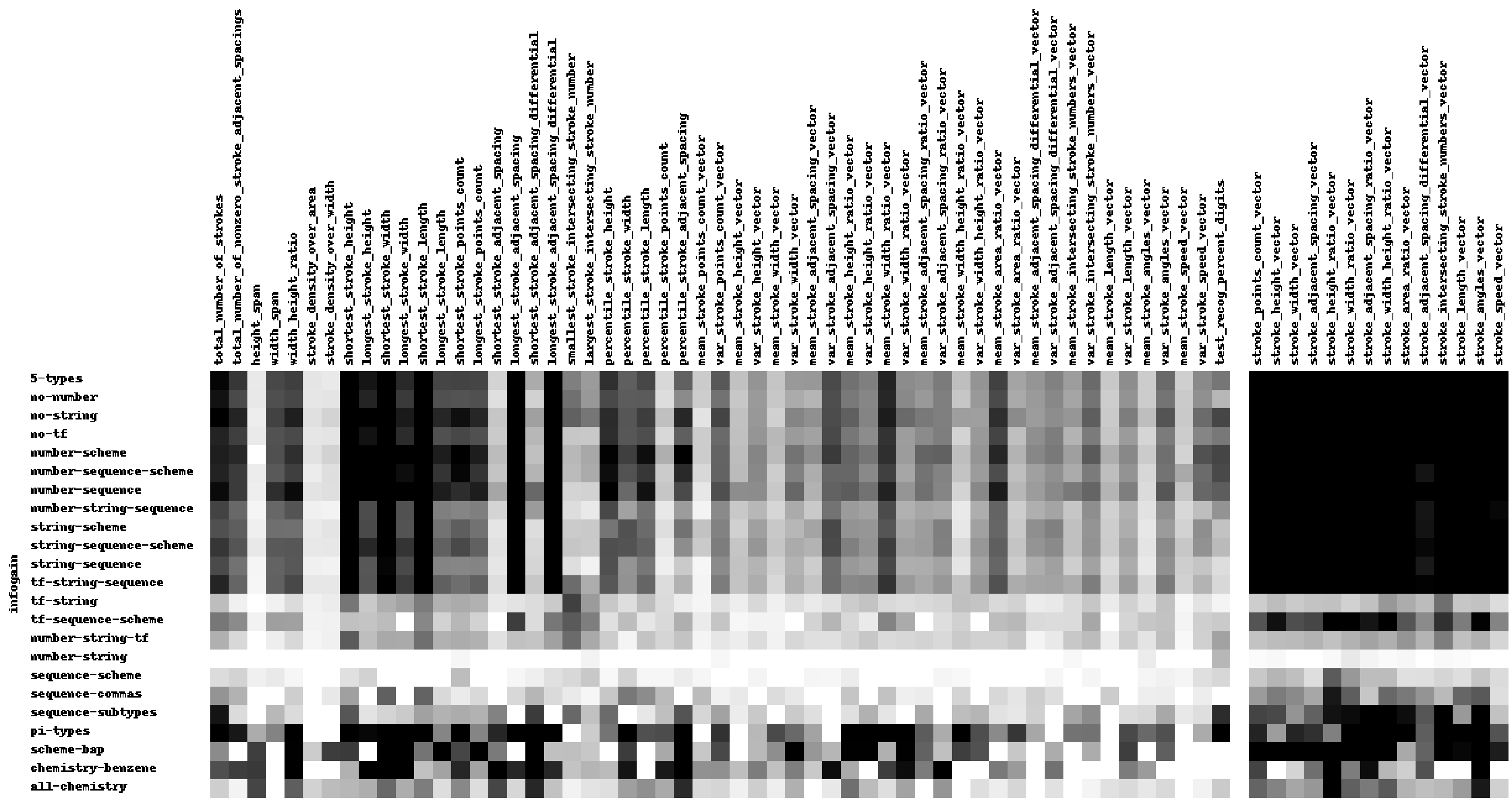


Figure D-3. This visualization summarizes the work of the InfoGain feature selection algorithm, highlighting the important features among all features that we extracted with darker cells.

Appendix E

Ink Type Prediction Confusion Matrix

Table E.1: Confusion matrix of our classification over 8 expected type classes for all 1958 samples using the SMO classifier and InfoGain feature selection algorithm. Precision (*P*), recall (*R*) and F-measure (*F*) values are also shown for each class.

x classified as X	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	***P***	***R***	***F***
True-False (a)	**36**	0	0	0	27	0	0	1	0.923	0.563	0.699
Scheme Exp (b)	0	**203**	0	0	4	1	0	41	0.886	0.815	0.849
Symbol (c)	0	0	**27**	0	3	0	0	2	0.931	0.844	0.885
Fraction (d)	0	0	0	**10**	0	0	0	0	1.000	1.000	1.000
String (e)	1	4	2	0	**431**	1	37	41	0.775	0.834	0.803
Diagram (f)	0	0	0	0	4	**117**	0	0	0.983	0.967	0.975
Number (g)	0	0	0	0	29	0	**168**	16	0.771	0.789	0.780
Sequence (h)	2	22	0	0	58	0	13	**657**	0.867	0.874	0.870
Correctly	**1649**								**84.22 %** = Accuracy		
Incorrectly	309								15.78 % = Error Rate		