USER MANUAL OF HAZY-CLASSIFY

M. Levent Koc
University of Wisconsin-Madison
koc@cs.wisc.edu

Christopher Ré
University of Wisconsin-Madison
koc@cs.wisc.edu

February 5, 2011

Contents

1 Overview

Classification is a ubiquitous and well-studied statistical task. Many applications such as Facebook and Twitter use classification. There are great tools to classify the data. Some examples are: SVM Light, Svm Sgd and Libsvm. However, the downside of these tools is that they are not able to incrementally maintain the output of classification results. Also, they are not well integrated with open source databases such as PostgreSQL.

Hazy-Classify is a system that implements classification inside an RDBMS for dynamic data as part of the ongoing Hazy Project. It supports classification techniques such as SVMs. One of the main contributions of Hazy-Classify is that it can efficiently maintain the output of classification results by using novel techniques. It achieves scalability and orders of magnitude speedup compared to prior implementations on many datasets including Forest Cover, DBLife and Citeseer. Hazy-Classify is implemented in C++ on top of PostgreSQL.

This manual contains an installation and usage guide for Hazy-Classify. For more information, please visit our project website.

2 Installation

Hazy-Classify is implemented in C++ on top of PostgreSQL. Therefore, it requires gcc/g++ compiler and PostgreSQL (8.4 or higher). In addition, it requires ocaml in order to parse Hazy queries. Figure 1 lists the URLs where you can get them.

gcc/g++	http://gcc.gnu.org/releases.html
PostgreSQL	http://www.postgresql.org/download/
ocaml	http://caml.inria.fr/download.en.html
Hazy-Classify	http://www.cs.wisc.edu/hazy/hazy-classify/download.php

Figure 1: Download links of the softwares

2.1 Installing Prerequisites

Install gcc/g++ If you don't have gcc/g++ compiler, you can download and install it from this link.

Install and Configure PostgreSQL You can download PostgreSQL 8.4(or higher) from here. Some systems can have PostgreSQL by default, but since Hazy-Classify uses libpq library of PostgreSQL, you need to download the source code and install it from scratch.

1. Let `PG_DIST` be the location where you unpacked the source distribution. Let `PG_PATH` be the location where you want to install PostgreSQL. Run the following commands:

```
cd PG_DIST
./configure --prefix=PG_PATH
gmake; gmake install
cd PG_PATH/bin
mkdir PG_PATH/data
```

```
initdb -D PG_PATH/data
postmaster -D PG_PATH/data &
createdb hazy_demo
```

initdb command initializes the database and postmaster runs PostgreSQL server. At the end, a database called hazy_demo is created.

2. You also need to add PG_PATH to the PATH variable of your system in order to run PostgreSQL commands without navigating to PG_PATH/bin directory.

```
PATH=$PATH:PG_PATH/bin
```

3. You need to set LD_LIBRARY_PATH to include PG_PATH/lib directory in order to run Hazy.

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:PG_PATH/lib
```

4. (Optional step) Finally, you can tune configuration parameters to obtain better performance results with Hazy-Classify. Assuming 4GB RAM (you can change parameters proportionally to RAM), the parameters in postgresql.conf (under PG_PATH/data) can be changed as follows:

```
shared buffers=896MB
work mem=256MB
maintenance work mem=512MB
effective cache size=1792MB
checkpoint segments=65536
```

The system might not allow shared buffer parameter to be increased since it uses more System V shared memory. In order to change this, you can edit /etc/sysctl.conf file and add these two lines, then run the command "sysctl -p". Note that, it requires root access.

```
kernel.shmall=671088640
kernel.shmmax=671088640
```

Install ocaml You can download ocaml from this link.

1. Let `OCAML_DIST` be the location where you unpacked the source distribution. Let `OCAML_PATH` be the location where you want to install ocaml. Run the following commands:

```
./configure -prefix OCAML_PATH
make world
make opt
make install
```

2. You also need to add OCAML_PATH to the PATH variable of your system in order to run ocaml commands without navigating to OCAML_PATH/bin directory.

```
PATH=$PATH:OCAML_PATH/bin
```

2.2 Configuring Hazy-Classify

Download Hazy-Classify from the project website. Let the directory where you unpack is HAZY_HOME. After unpacking, you should see two directories: "doc" and "system_src". When you navigate to "system_src", you can see Hazy-Classify configuration file named "hazy.conf" (Figure 2).

- POSTGRES_PATH: PG_PATH specified during the installation of PostgreSQL.
- Database: Name of the database you want to use Hazy-Classify
- Storage_Manager: Type of the storage manager that Hazy-Classify uses.
  Options: Main_Memory_SM, Ondisk_SM, Hybrid_SM
- Use_Eps_Map: Hazy-Classify uses eps map when Storage_Manager is Hybrid_SM and this variable is set to 1.
- Hazy_Strategy: The strategy that is used by Hazy-Classify during incremental updates.
  Options: EAGER_HAZY, EAGER_NAIVE, LAZY_HAZY, LAZY_NAIVE
- Reservoir_Size: If you want Hazy-Classify to use reservoir during the training, you can set this variable to a value > 0 (e.g. 1000)
- Buffer_Size: Size of the buffer that is used by Hazy-Classify when Storage_Manager is Hybrid_SM. (Usually, 1% of the # of entities).

```
POSTGRES_PATH=/hazy/Programs/postgresql
Database=hazy_demo
Storage_Manager=Ondisk_SM
Use_Eps_Map=0
Hazy_Strategy=EAGER_HAZY
Reservoir_Size=0
Buffer_Size=0
```

Figure 2: Default Configuration File

After setting the parameters in the configuration file, you can install Hazy-Classify by running the following commands. The sample output of this installation is shown in Figure 3.

```
cd HAZY_HOME
cd system_src/Hazy
sh scripts/install.sh
```

Note that, you need to call `install.sh` script for each database that you want to use Hazy-Classify.

3 Using Hazy-Classify

In this section, we describe how to use Hazy-Classify. In order to explain how Hazy-Classify can be used, we will use the demo examples that you can find under `HAZY_HOME/system_src/Hazy/demo`. We first show how to load demo examples and then how to create, update and query classifications by using Hazy-Classify.

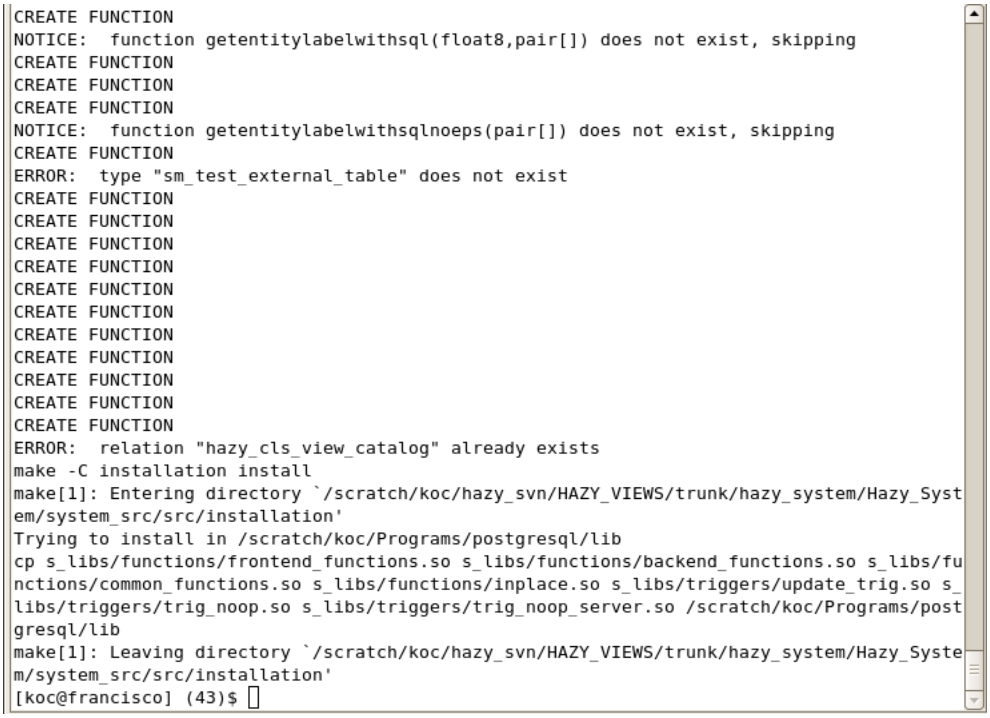

Figure 3: Terminal Output of Hazy installation

3.1 Loading demo examples

Demo examples contain 50 computer science publication names. Our purpose is to classify these papers as database and non-database papers and maintain them efficiently in the presence of updates. You can load these examples to the database by running the following commands:

```
cd HAZY_HOME/system_src/Hazy
psql hazy_demo < demo/demo_papers.sql
```

This will create two tables in `hazy_demo` database: `demo_papers` that contains 50 publication names and `demo_training_examples` that will be used to give training examples to Hazy-Classify. Sample terminal output of these commands is shown in Figure 4.

3.2 Creating Classification

In order to use Hazy-Classify, first you need to run Hazy server. To start the Hazy server, run the following commands as shown in Figure 5.

```
cd HAZY_HOME/system_src/Hazy
./exec/hazy_main
```

After starting Hazy server, now we are ready to create the view. In order to create a view, you need to write a query in Hazy-Classify syntax. `HAZY_HOME/system_src/Hazy/demo/create_demo_view_query` contains create view query for the demo example.

Figure 4: Terminal Output of Loading Demo Examples

Figure 5: Terminal Output of Starting Hazy Server

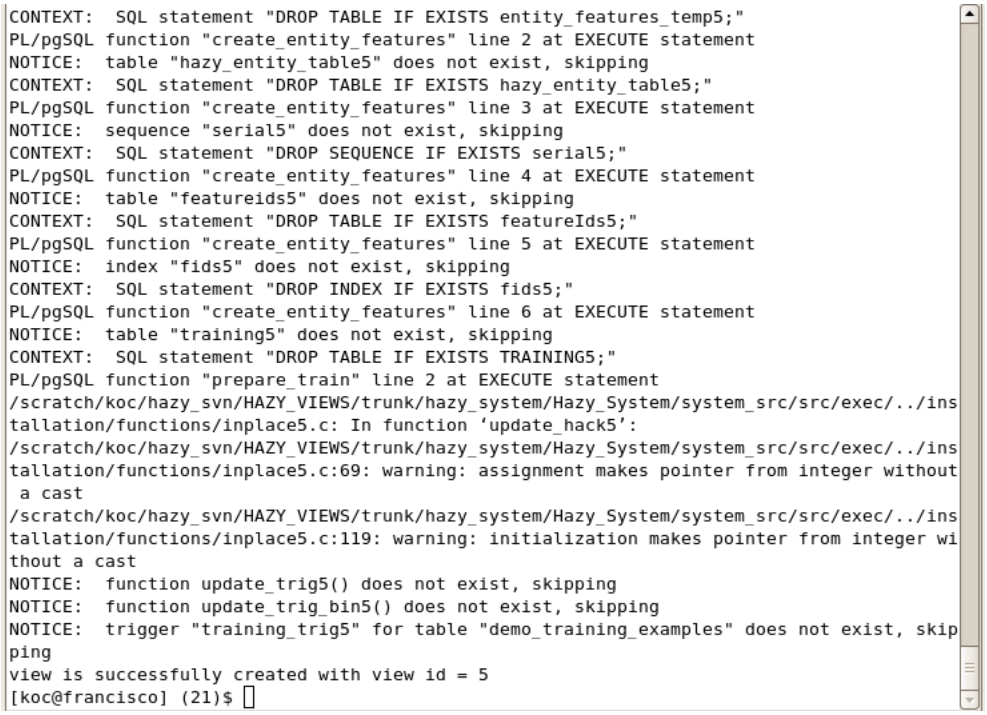

Figure 6: Terminal Output of Create View

```
CREATE CLASSIFICATION VIEW demo_labeled_papers KEY id
ENTITIES FROM demo_papers KEY id
LABELS FROM Paper_Area LABEL l
EXAMPLES FROM demo_training_examples KEY id LABEL l
FEATURE FUNCTION tf_bag_of_words;
```

This query declares a view demo_labeled_papers that contains each paper in demo_papers where papers are labeled by Hazy-Classify as database or non-database papers. Name of the entity table that contains id and title of the publication is demo_papers. demo_training_examples table contains training examples to train a model by Hazy-Classify. Also, to extract the features of the publications, Hazy-Classify uses tf_bag_of_words. Paper_Area table specifies possible labels (in our case, database paper or not).

In order to create the view, execute the following commands:

```
cd HAZY_HOME/system_src/Hazy
cat demo/create_demo_view_query | ./exec/hazy_parser
```

As you can see from Fig 6, the view is successfully created with view id 5.

3.3 Querying the View

Since our view demo_labeled_papers is created in hazy_demo database of PostGreSQL, we can query it like a regular database table (Actually, since our strategy is eager in hazy.conf file, the view is already a regular database table). First, we need to login PostGreSQL:

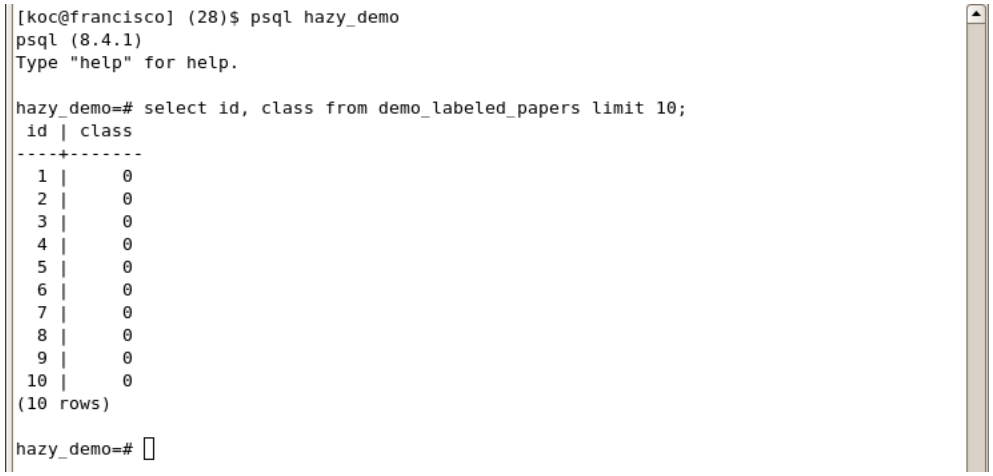

Figure 7: Terminal Output of Query View

```
psql hazy_demo
```

Then, we can run the following SQL query to retrieve publication id and their classes:

```
hazy_demo=# select id, class from demo_labeled_papers limit 10;
```

Output of this query is shown in Fig 7.

As we did not provide any training example, the SVM model initially classifies every entity as non-database paper (class is 0).

3.4 Updating the View

We can provide training examples to Hazy-Classify in order to obtain more successful classification results. For example, when we query demo_papers, we see that the title of the paper with id = 7 is "Query Processing in Spatial Network Databases" which is a database paper.

```
hazy_demo=# select * from demo_papers where id = 7;
id | name
----+-----------------------------------------------
7 | Query Processing in Spatial Network Databases.
```

However, this paper is classified as non-database paper as shown in Fig 7. We can insert a training example to Hazy-Classify that the paper with id 7 is a database paper (class = 1) and it handles re-classification of entities and updating the labels of entities automatically. However, we need to have paper's feature vector since demo_training_examples table requires feature vector field. One of Hazy-Classify's internal tables (hazy_entity_table5) contains all feature vectors. Note that, 5 at the end of hazy_entity_table is id of our view. In order to insert this training example, we run the following SQL query:

```
hazy_demo=# insert into demo_training_examples values(7, 1, (select feature_vector FROM hazy_entity_ta
```

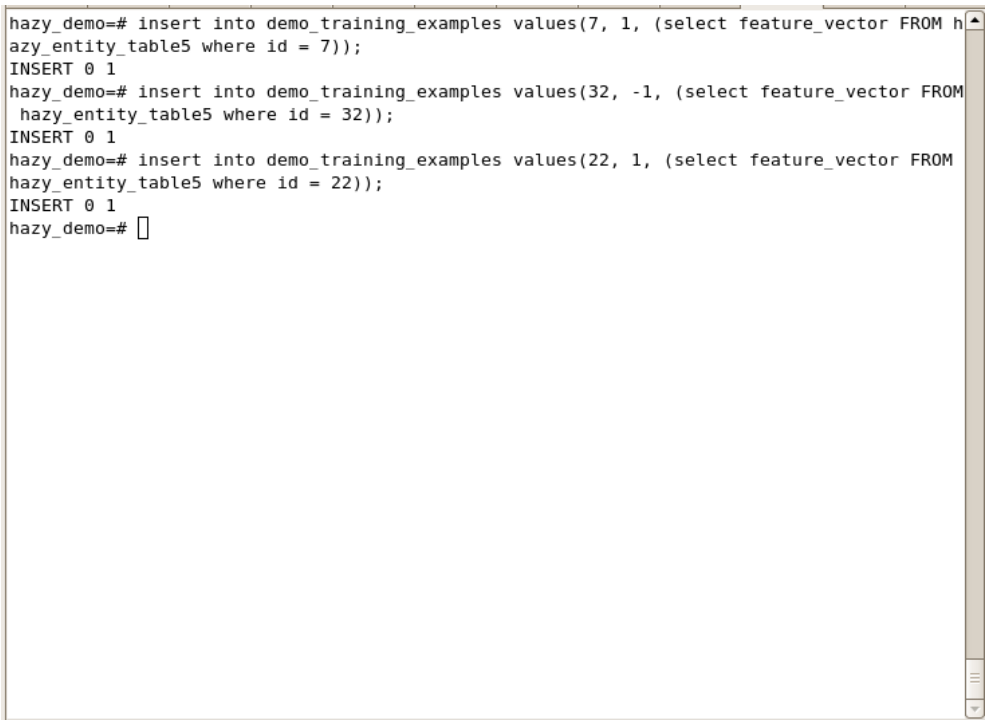

Figure 8: Terminal Output of Inserting Examples

We can feed Hazy-Classify with more training examples:

```
hazy_demo=# insert into demo_training_examples values(32, -1, (select feature_vector FROM hazy_entity_
hazy_demo=# insert into demo_training_examples values(22, 1, (select feature_vector FROM hazy_entity_t
```

Figure 8 shows the terminal output of inserting examples.

Now, when we re-query the view, we see that the classes of the examples change as shown in Figure 9.

```
hazy_demo=# select * from demo_labeled_papers;
```

```
 id | eps | class
----+-----+-------
  1 |   0 |     1
  2 |   0 |     1
  3 |   0 |     1
  4 |   0 |     1
  5 |   0 |     1
  6 |   0 |     1
  7 |   0 |     1
  8 |   0 |     1
  9 |   0 |     1
 10 |   0 |     1
 11 |   0 |     1
 12 |   0 |     1
 13 |   0 |     1
 14 |   0 |     1
 15 |   0 |     1
 16 |   0 |     1
 17 |   0 |     1
 18 |   0 |     1
 19 |   0 |     0
 20 |   0 |     1
 21 |   0 |     1
 22 |   0 |     1
 23 |   0 |     1
 24 |   0 |     1
 25 |   0 |     1
 26 |   0 |     1
 27 |   0 |     1
 28 |   0 |     1
--More--
```

Figure 9: Terminal Output of Query View