

# MUTATIONMINER: A Graph Theoretic Approach to Extract Point Mutation Data from Biomedical Literature

Lawrence C. Lee<sup>1</sup>, Florence Horn, Fred E. Cohen • Department of Molecular and Cellular Pharmacology / Biological and Medical Informatics • University of California San Francisco • <sup>1</sup>lle8@itsa.ucsf.edu

## Introduction

Point mutation information lends itself to robust methods of automated extraction because of its relative homogeneity in representation. Using simple indexing, parsing, and searching methods, **MUTATIONMINER** extracts point mutation data from biomedical literature and organizes it into a functional format.

## Why?

Mutation→Effect relationships are an integral part of protein structure/function studies. The recent trend of exponential growth in amounts of biomedical literature exceeds the ability of a researcher to manually read/extract information efficiently. **MUTATIONMINER** would fill the need for quick and efficient parsing of biomedical literature for point mutation information.

## Project Goals and Strategy

The goal of **MUTATIONMINER** is to efficiently extract point mutation information from a large number of journal articles about any protein family and to store the extracted point mutation information in a database.

1. Develop a **Information Retrieval (IR)** engine to efficiently search and download full text articles from PubMed related to a specific protein family.
2. Develop an Information Retrieval system to download complete gene and protein information from SWISS-PROT for a complete protein family.
3. Develop a **Information Extraction (IE)** algorithm for extracting point mutation information from scientific literature and a cross checking facility to verify the mutations found.
4. Create a relational database schema for storing the full text articles, protein information, and mutation data.
5. Exploit this database to gain new insights into structure-function relationships within a protein family.

## Methods

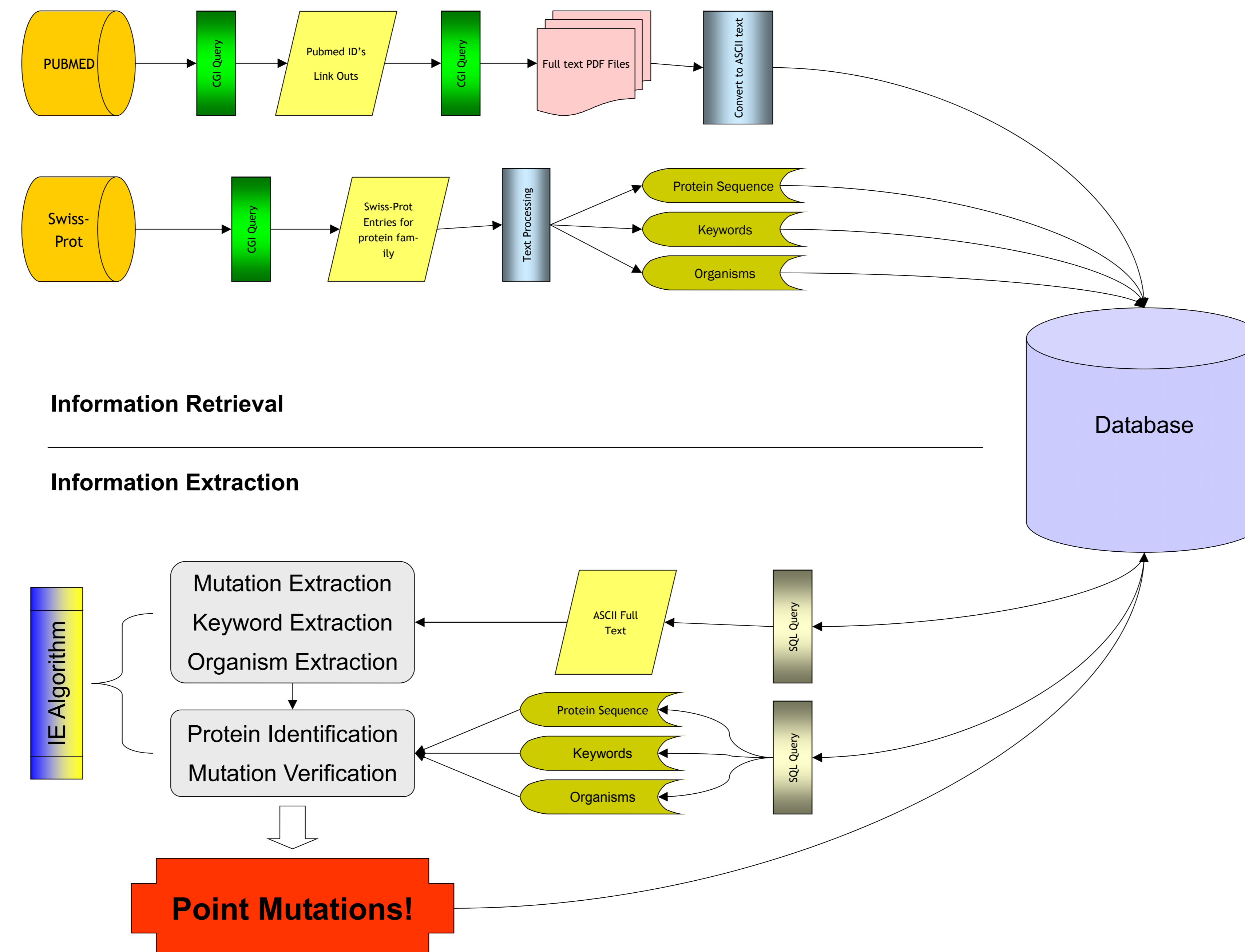
### Information Retrieval (IR)

1. A set of Python tools were written to query the NCBI Entrez EUtilities to search PubMed for relevant articles about a specific protein family and to download full-text PDF files for those articles.
2. Downloaded XML build of Swiss-Prot database and used Python tools to parse/extract relevant protein entries.
3. Constructed "dictionaries" containing keywords, descriptors, gene names, and organism names of relevant proteins from SwissProt entries.

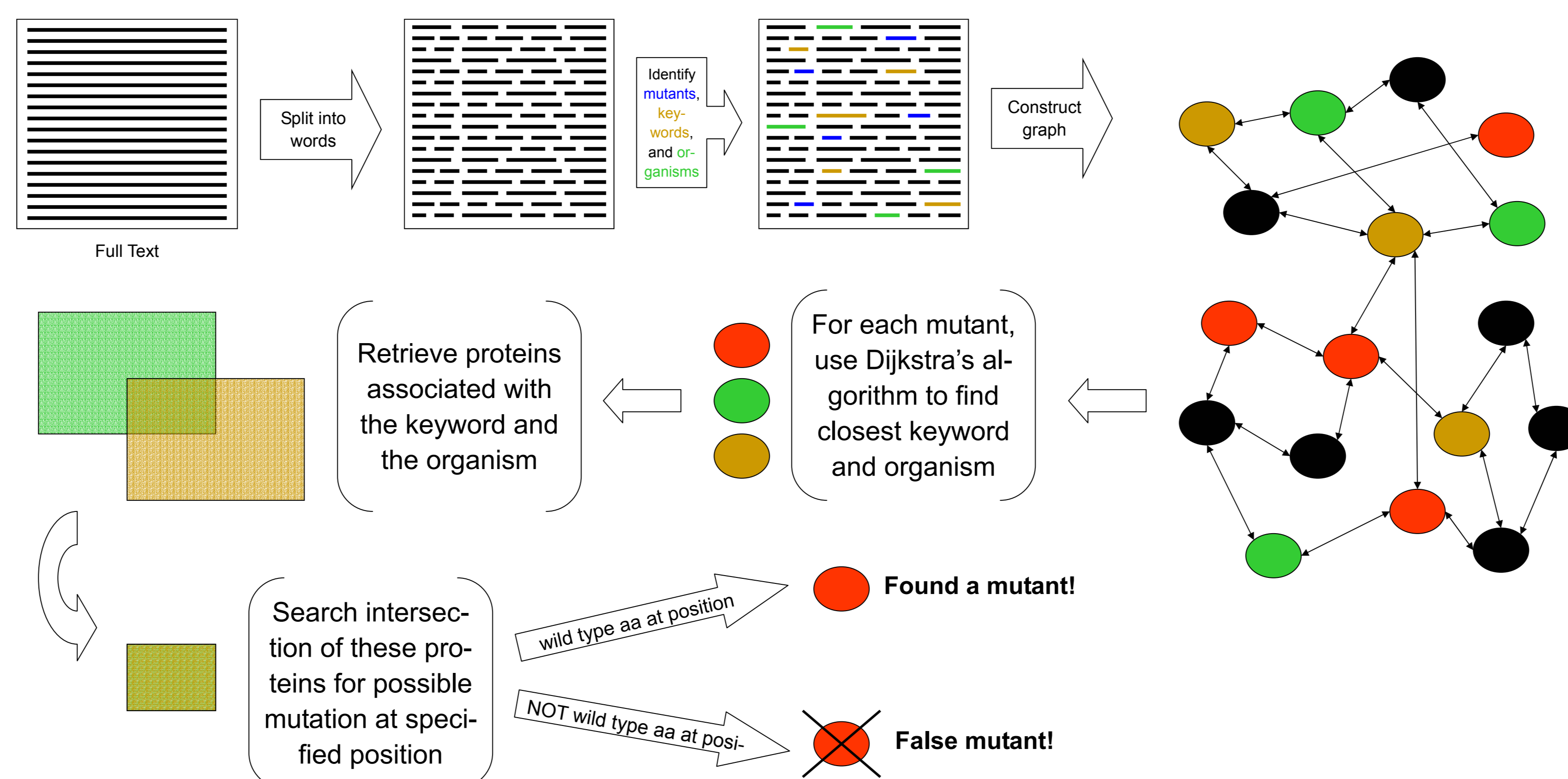
### Information Extraction (IE) Algorithm

1. For each journal article, first split the text by dictionary terms and mutation strings, then by white space and punctuation.
  - a. Mutation strings are formatted like: A123T, Ala123Thr, Ala(123)Thr, Ala123→Thr.
2. Parse the list of split terms and create a fully connected graph with each node representing a unique term found in the list and edges connecting terms which are present adjacent to each other in the list.
3. Using a heuristic function which calculates overall "distance" between two terms in the graph, find the "closest" keyword/descriptor term(s) and organism name(s) to each mutation string.
4. Search relevant Swiss-Prot entries for proteins containing the targeted keyword/descriptor term and organism name.
5. Compare resulting protein(s) sequence with mutation string, checking if the wild-type amino acid at the mutation position corresponds to the same wild-type amino acid in the mutation string.

## Process Flow



## Information Extraction Algorithm



## Test

The IR tools were tested by retrieving articles returned by querying PubMed for "tyrosine kinase mutations". The IE algorithm was experimented on a set of 100 G-Protein Coupled Receptor articles taken from the TinyGrap<sup>1</sup> GPCR Mutation database and 70 tyrosine kinase articles retrieved. All 70 of the tyrosine kinase articles were used as a training set, while 50 of the GPCR articles were used as a training set and the remaining 50 as a test set.

The IE algorithm was "trained" by running the algorithm on the training sets, then recursively altering regular expression rules, search parameters, filter rules, and dictionary construction until false positive and false negative counts were at acceptable levels respective to the data sets.

<sup>1</sup><http://tinygrap.uit.no/>

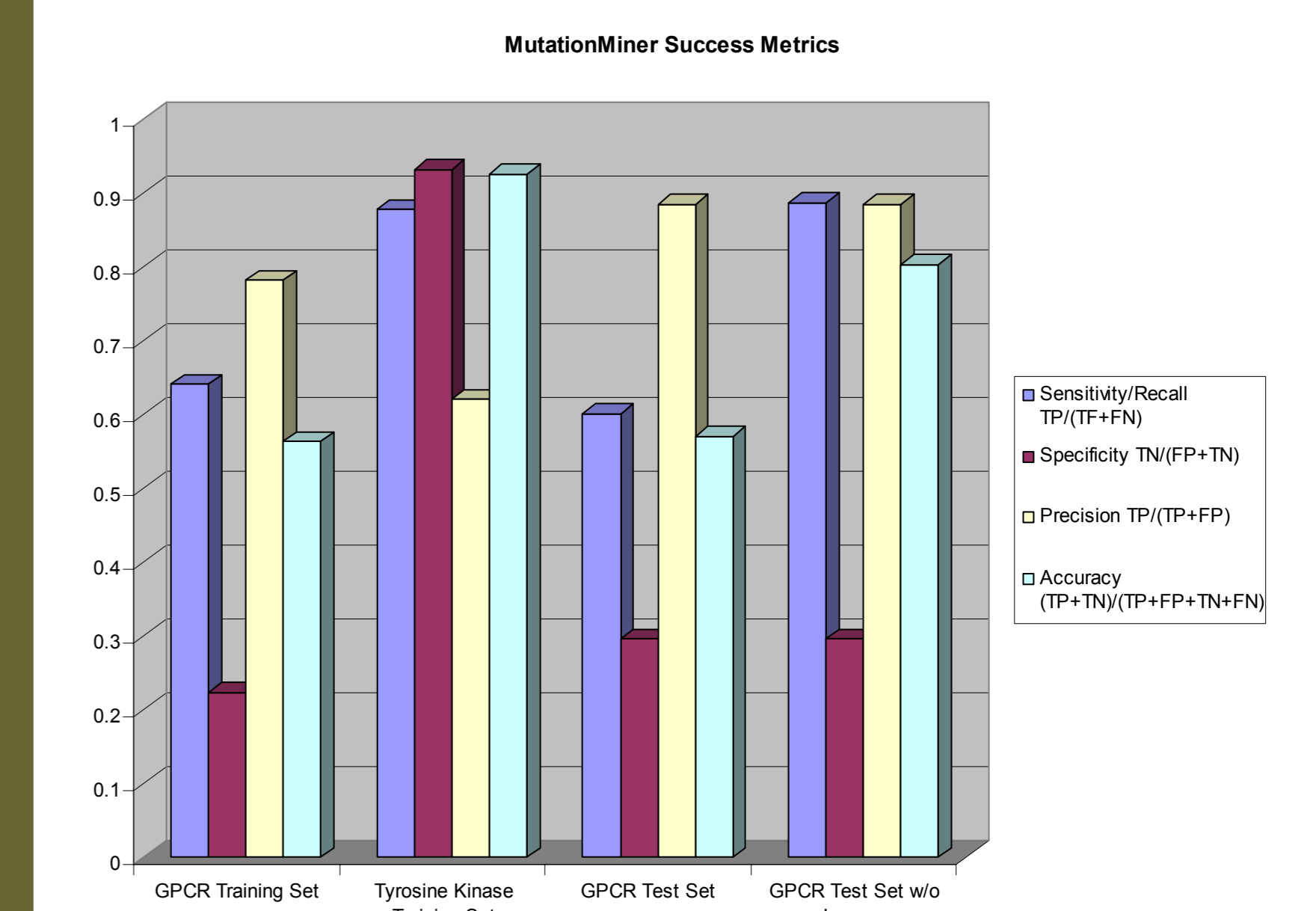
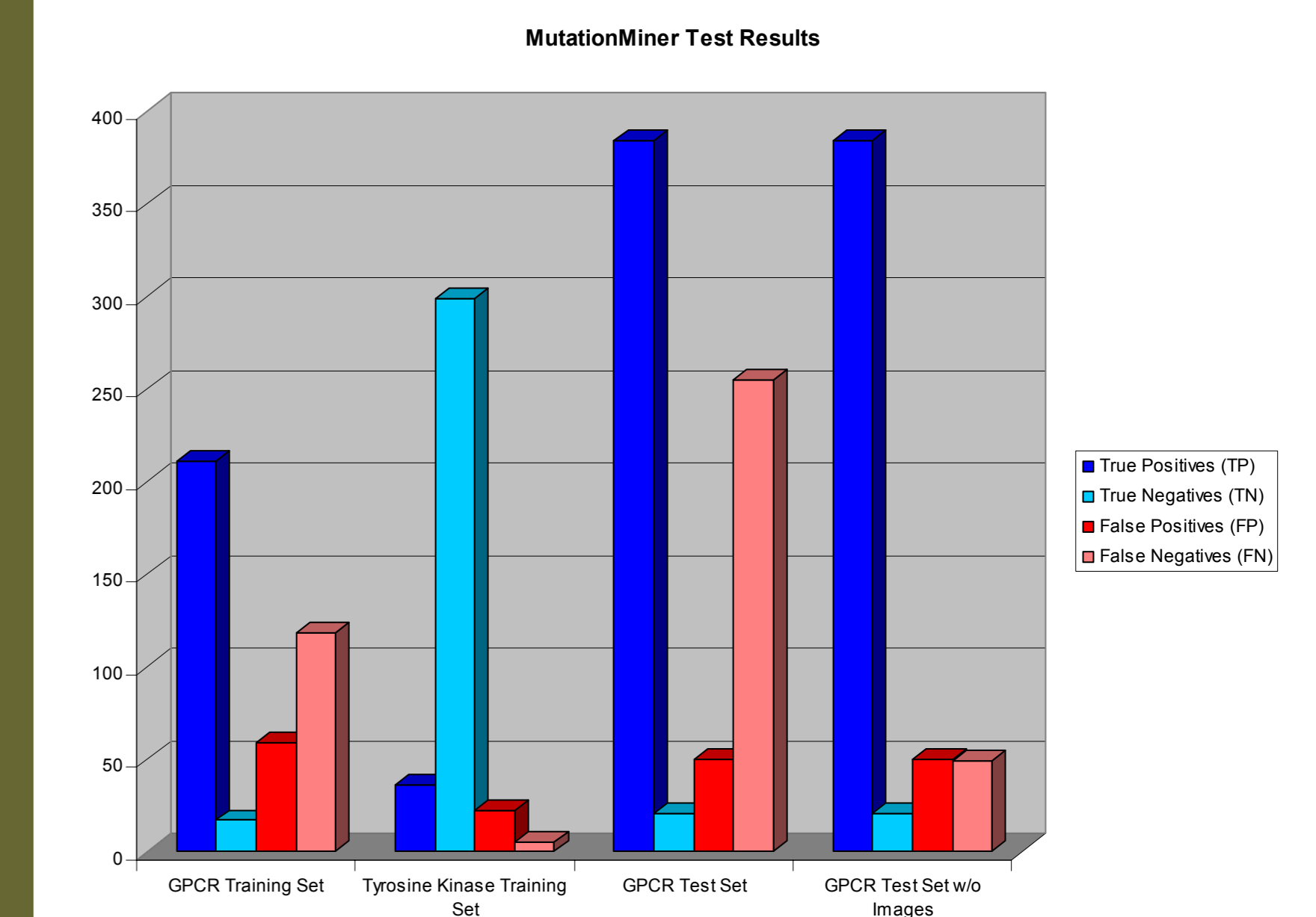
## Information Retrieval (IR) Results

As of 6/02

# MEDLINE hits for "tyrosine kinase mutation"	6412
# PDF files downloaded from the hits	1620
# PDF files converted to ASCII text	1390
# tyrosine kinase proteins in Swiss-Prot	323
# keywords extracted from Swiss-Prot entries	896
# organisms extracted from Swiss-Prot entries	245

## Information Extraction (IE) Results

	GPCR Training Set	Tyrosine Kinase Training Set	GPCR Test Set	GPCR Test Set w/o Images
True Positives (TP)	211	36	384	384
True Negatives (TN)	17	299	21	21
False Positives (FP)	59	22	50	50
False Negatives (FN)	118	5	255	49
Sensitivity/Recall TP/(TP+FN)	0.64	0.88	0.60	0.89
Specificity TN/(FP+TN)	0.22	0.93	0.30	0.30
Precision TP/(TP+FP)	0.78	0.62	0.88	0.88
Accuracy (TP+TN)	0.56	0.93	0.57	0.80



## Discussion/Conclusion

Because of the wide variance in the way different online journals allow access to their full text PDF articles, only 1/4th of the total number of articles could be downloaded. Some other factors which influenced the number of articles which could be downloaded included: login/password requirements, javascript links which could not be handled with CGI commands, and journal publications not offering PDF versions of articles.

The mutation extraction/verification algorithm works at a high precision/recall level. Problems with false positives arise when the protein belonging to the mutation is not in the database, but we find another protein which happens to have the same wild type amino acid at the mutant position. Problems with false negatives arise when the mutation is in a grammatical format instead of a regular format. A natural language processing component will need to be added to catch these mutations.