fenominal Documentation

Release v0.7.10

Peter Robinson

Jul 27, 2023

CONTENTS:

CHAPTER

ONE

FENOMINAL

Fenominal is a Java 17 library for text-mining Human Phenotype Ontology (HPO) terms from text. Fenominal is a multimodule project with a `core` module with the text-mining logic and a `cli` module for use from the command line. A graphical user interface (GUI) is available in a separate project called Fenominal-GUI.

Fenominal implements the T-BLAT algorithm, which is inspired by the BLAST algorithm for biosequence alignment. T-BLAT screens texts for potential matches on the basis of matching k-mer counts and scores candidates based on conformance to typical patterns of spelling errors derived from 2.9 million clinical notes. Fenominal also implements exact matching but matches also on multitoken HPO labels or synonyms that are permuted.

Fenominal does not rely on external APIs and can be used in settings in which a firewall does not permit applications to access the internet. Fenominal is intended for use as a software library and the CLI module only contains simple demo applications.

1.1 Fenominal's Matching Algorithms

Fenominal performs both exact and fuzzy (T-BLAT) matching. In both cases, the order of the tokens of each HPO term can be present in any order and stop words are ignored. For instance, Mallet finger HP:0030771 will be inferred from both `mallet finger` and `finger mallet`. For the term Anomalous hepatic venous drainage into the left atrium HP:0032181, the stopwords `the` and `into` are not considered.

1.1.1 Exact matching

The exact matching algorithm searches for exact matches to term or synonym labels, ignoriing stop words, whereby the tokens can be present in any order in the text.

1.1.2 T-BLAT (fuzzy) matching

TODO – short description

1.2 CLI fenominal application

fenominal is a Java library written in Java 17. fenominal contains a command-line interface (cli) module that demonstrates some of the functionality of the library.

See rstclisetup for instructions on building the application.

See rstclidownload for instructions on how to download the `hp.json` file that is needed for the app.

Once you have built the application, you should see this

```
java -jar fenominal-cli/target/fenominal.jar
    Usage: fenominal [-hV] [COMMAND]
    phenotype/disease NER
        -h, --help      Show this help message and exit.
        -V, --version   Print version information and exit.
    Commands:
        download, D  Download files for fenominal
        parse, P     Parse text
```

in the following we will show only `fenominal.jar`. Adjust the path accordingly.

1.2.1 Fenominal's Matching

Fenominal can perform exact matching and fuzzy matching. See *Fenominal's Matching Algorithms* for an explanation.

To see the options, run `java -jar fenominal.jar parse -h`.

Short option	Long option	Explanation
-e	–exact	Use exact matching algorithm
-h	–help	Show this help message and exit.
	–hp=<path>	Path to HP json file (default data/hp.json)
-i	–input=<path>	Path to HP json file (default data/hp.json)
-0	–output=<path>	Path to output file
-V	–version	Print version information and exit.
	–verbose	Show parse results in shell

Exact matching

For this example, create a file called `text-exact.txt` with the following contents

> A 28-year-old woman who was diagnosed with Noonan syndrome at age 4 because of growth retardation, cardiomyopathy, and hypertelorism.

Run fenominal as follows.

```
java -jar fenominal.jar parse --exact -i text-exact.txt --verbose
(...)
Growth delay HP:0001510      growth retardation      observed        79      97      A␣
↪28-year-old woman who was diagnosed with Noonan syndrome at age 4 because of growth␣
↪retardation, cardiomyopathy, and hypertelorism.
Cardiomyopathy       HP:0001638      cardiomyopathy  observed        99      113     A␣
↪28-year-old woman who was diagnosed with Noonan syndrome at age 4 because of growth␣
↪retardation, cardiomyopathy, and hypertelorism.
```

(continues on next page)

(continued from previous page)

```
Hypertelorism        HP:0000316      hypertelorism   observed        119     132     A␣
↪28-year-old woman who was diagnosed with Noonan syndrome at age 4 because of growth␣
↪retardation, cardiomyopathy, and hypertelorism.
```

T-BLAT (fuzzy) Matching

For this example, create a file called `text-errors.txt` with the following contents.

> A 28-year-old woman who was diagnosed with Noonan syndrome at age 4 because of growth retadation, cardiomyopathic, and hypertelorisn.

Run fenominal as follows.

```
java -jar fenominal.jar parse -i text-errors.txt --verbose
(...)
 Agnosia     HP:0010524      agnosia observed        28      37      A 28-year-old woman␣
↪who was diagnosed with Noonan syndrome at age 4 because of growth retadation,␣
↪cardiomyopathic, and hypertelorisn.
 Growth delay        HP:0001510      growth retardation      observed        79      96 ␣
↪    A 28-year-old woman who was diagnosed with Noonan syndrome at age 4 because of␣
↪growth retadation, cardiomyopathic, and hypertelorisn.
 Cardiomyopathy      HP:0001638      cardiomyopathy  observed        98      113     A␣
↪28-year-old woman who was diagnosed with Noonan syndrome at age 4 because of growth␣
↪retadation, cardiomyopathic, and hypertelorisn.
 Hypertelorism       HP:0000316      hypertelorism   observed        119     132     A␣
↪28-year-old woman who was diagnosed with Noonan syndrome at age 4 because of growth␣
↪retadation, cardiomyopathic, and hypertelorisn.
```

Fenominal picks up four terms. Agnosia is false postion (from the word diagnosed). The remaining three terms are inferred correctly despite the presence of spelling errors or variants. Note that T-BLAT is the default approach, and exact matching is only performed if the `--exact` flag is passed.

1.3 fenominal library

To use fenominal as a library for Java 17 or higher applications, add the following to the POM file.

```
<dependency>
    <groupId>org.monarchinitiative.fenominal</groupId>
    <artifactId>fenominal-core</artifactId>
    <version>...</version>
</dependency>
<dependency>
    <groupId>org.monarchinitiative.phenol</groupId>
    <artifactId>phenol-core</artifactId>
    <version>...</version>
</dependency>
<dependency>
    <groupId>org.monarchinitiative.phenol</groupId>
    <artifactId>phenol-io</artifactId>
    <version>...</version>
</dependency>
```

Using the latest versions of fenominal and phenol.

TODO explain how to use GRAIL.

1.3.1 Imports

Use the following imports

```
import org.monarchinitiative.fenominal.core.FenominalRunTimeException;
import org.monarchinitiative.fenominal.core.TermMiner;
import org.monarchinitiative.fenominal.model.MinedSentence;
import org.monarchinitiative.fenominal.model.MinedTermWithMetadata;
import org.monarchinitiative.phenol.io.OntologyLoader;
import org.monarchinitiative.phenol.ontology.data.Ontology;
import org.monarchinitiative.phenol.ontology.data.TermId;
```

Initialize with the path to the hp.json file

```
Ontology ontology = OntologyLoader.loadOntology(new File(hpoJsonPath));
```

Decide whether to do exact or T-BLAT (fuzzy) matching

```
boolean doExactMatching = ....// your code decides
TermMiner miner;
if (exact) {
    miner = TermMiner.defaultNonFuzzyMapper(this.ontology);
} else {
    miner = TermMiner.defaultFuzzyMapper(this.ontology);
}
```

You can use fenominal to retrieve three types of objects.

1.3.2 sentences

Retrieve a collection of `MinedSentence` objects that represent each of the sentences in the input string in which at least one HPO term is indentified. Each MinedSentence object has a collection of MinedTermWithMetadata objects.

```
String inputString = ....// your code provides input String
Collection<MinedSentence> setences = miner.mineSentences(inputString);
```

1.3.3 MinedTermWithMetadata

Returns a collection of MinedTermWithMetadata objects, each of which provides the following methods

- `String getMatchingString()`: the matching string in the original text
- `double getSimilarity()` the similaroty score for the match
- `TermId getTermId()`: the HPO TermId
- `int getTokenCount()`: - the number of matching tokens

Additionally, all of the methods of MinedTerm are provided (see below)

```
String inputString = ....// your code provides input String
Collection<MinedTermWithMetadata> setences = miner.mineTermsWithMetadata(inputString);
```

1.3.4 MinedTerm

Returns a collection of MinedTerm objects, each of which provides the following methods

- `int getBegin()`: zero-based start coordinate of the match in the original text
- `int getEnd()`: zero-based end coordinate (included) of the match in the original text
- `String getTermIdAsString()`: String version of the HPO term id.
- `boolean isPresent()`: true of the HPO term was observed, false if it was excluded according to the original text

```
String inputString = ....// your code provides input String
Collection<MinedTerm> sentences = miner.mineTerms(inputString);
```