Přílohy

A. Uživatelská dokumentace

A.1 Úvod

COPF (Crude oil price forecast) je systém pro odhad vývoje ceny ropy na základě textových zpravodajských informací. Poskytuje také nástroje pro vylepšení tohoto odhadu.

COPF je určen pro platformu Linux (testováno na distribucích Debian a Ubuntu).

A.2 Instalace

Nejdříve je nutné zkopírovat program z CD (cesta k adresáři v rámci CD je uvedena v tabulce C.1) do počítače, protože potřebuje zapisovat do svých složek.

Pro fungování systému je nutné nainstalovat několik externích komponent. Kromě JAVA SDK je potřeba v terminálu provést tyto příkazy:

Nejdříve je nutné nastavit do proměnné COMP_PATH cestu k adresáři components (cesta k adresáři components v rámci CD je uvedena v tabulce C.1). Tedy například:

```
COMP_PATH="./components/"
ABS_COMP_PATH=`cd $COMP_PATH; pwd`
```

Dále potřebujeme nainstalovat nástroje pro kompilaci:

```
sudo apt-get install g++
```

Poté už nainstalujeme první komponentu, Charniak Parser (zde jsme při stažení z Internetu měli problémy, museli jsme přepisovat některé hlavičky a trochu poopravit Makefile):

```
cd ${ABS_COMP_PATH}TIMEX2_system/Charniak/reranking-parser/
export GCCFLAGS="-march=native"
make clean
make
```

Dále potřebujeme knihovnu libsvm (je potřeba verze 2.81, novější verze mají jiné rozhraní), jež je napsána v Pythonu, který proto musíme také nainstalovat (testováno na verzích 2.7 a 2.6).

```
sudo apt-get install python2.7-dev
```

Pokud jsme nainstalovali jinou verzi Pythonu než 2.7, je potřeba ještě změnit cestu v Makefile pro Python. Např. pro verzi 2.6:

```
cd ${ABS_COMP_PATH}TIMEX2_system/libsvm-2.81/python/
sed -i 's/python[0-9]\.[0-9]*/python2.6/' Makefile
```

V tuto chvíli nám už nic neschází k nainstalování libsvm:

```
cd ${ABS_COMP_PATH}TIMEX2_system/libsvm-2.81/
make clean
make all
cd python/
make clean
make all
```

Nakonec ještě musíme přidat potřebná práva pro spuštění:

```
cd ${ABS_COMP_PATH}TIMEX2_system/
chmod +x find_all_timexes.sh
chmod +x find_timexes.sh
chmod +x ./timextag-0.3/deptools/charniak-dep-parser
chmod +x ./timextag-0.3/deptools/convert/shuffle-deps.pl
chmod +x ./timextag-0.3/deptools/convert/wsj2dep.pl
```

A.3 Příprava

Abychom mohli hodnotit cenu ropy, musíme získat vstupní data pro náš program. Nejdříve si nastavíme potřebné cesty k budoucím datům, což se dělá v souboru path.txt. Zde jsou tři cesty, první ke kořeni celého projektu, druhá k adresáři s daty a třetí do složky components k adresáři časových určení získaných pomocí TimexTagu. Například:

```
projectRootPath=./
downloadedTextsPath=./data/
timextagSgmlTextsPath=./components/TIMEX2_system/Texts/
```

Dále v konsoli nastavíme cesty ke spouštěcím souborům. Tedy pokud jsme v kořenovém adresáři projektu:

```
JARDIR_PATH="."
ABS_JARDIR_PATH=`cd $JARDIR_PATH; pwd`

FINDTIM_PATH="./components/TIMEX2_system/"
ABS_FINDTIM_PATH=`cd $FINDTIM_PATH; pwd`
```

Poté stáhneme články a historii cen ropy z Internetu. Zde si můžeme vybrat od jakého data chceme stahovat, a to přidáním atributu podle vzoru yyyy-MM-dd. Články program stáhne pouze chybějící. Stahování článků je náročné na čas a připojení k Internetu.

```
cd $ABS_JARDIR_PATH
java -ea -classpath COPF.jar copf.downloaders.pagedownloaders.
   DownloadAllPages [yyyy-MM-dd]
java -ea -classpath COPF.jar copf.downloaders.pricedownloaders.
   DownloadAllPrices [yyyy-MM-dd]
```

Články z Internetu jsou uloženy v XML souboru. Pro další potřeby je třeba provést konverzi (co článek, to soubor):

```
java -ea -classpath COPF.jar copf.utilities.textsconverters.
   ConverterXml2Files
```

Na článcích je nutné provést automatickou anotaci pomocí komponenty Treex. Tuto anotaci jsme prováděli na katedře ÚFAL a komponentu Treex k projektu nepřikládáme. Z ÚFALu jsme dostali články zabalené v gz formátu, proto je musíme rozbalit:

```
java -ea -classpath COPF.jar copf.utilities.textsconverters.
   ConverterGz2Txt
```

Texty je potřeba ještě časově ohodnotit. Pro toto ohodnocení musíme vytvořit soubory v SGML formátu. Po ohodnocení je nutné toto časové ohodnocení zapsat zpět do XML. Časové ohodnocování je časově nejnáročnější část predikce.

```
java -ea -classpath COPF.jar copf.utilities.textsconverters.
   ConverterXml2Sgml
cd $FINDTIM_PATH
./find_all_timexes.sh
cd $ABS_JARDIR_PATH
java -ea -classpath COPF.jar copf.utilities.textsconverters.
   ConverterTimexXml2DateRange
```

V tuto chvíli musíme provést filtraci textů. Abychom ji mohli provést, musíme mít množinu manuálně ohodnocených textů nebo potřebný model pro klasifikátor sloužící k filtraci. Manuální evaluaci pro filtraci textů provádíme pomocí speciálního nástroje (viz A.5.1).

Takto spustíme samotnou filtraci:

```
java -ea -classpath COPF.jar copf.FilterTexts [-model {cesta k
   modelu pro filtraci}]
```

Pro druhou fázi evaluace je potřeba vytvořit model (pokud máme model již vytvořený, můžeme tento příkaz vynechat):

```
java -ea -classpath COPF.jar copf.CreateModelFromTexts
```

A.3.1 Přiložená data na CD

Na přiloženém CD jsou data, která jsme používali pro experiment (složka DataForExperiment, cesta v rámci CD viz tabulka C.1) a pro konečnou evaluaci (složka LastEvalData, cesta v rámci CD viz tabulka C.1). Z důvodu velikosti je na CD v rámci těchto dat jen to, co je potřebné pro predikci a evaluaci. Texty anotované pomocí komponenty Treex jsme ponechali pouze zabalené. Pro použití

těchto dat k predikci a evaluaci je tedy nutné rozbalit anotované texty (viz krok A.3), ostatní kroky přípravy dat můžeme vynechat.

Dále jsme na CD přiložili model pro filtraci a model pro predikci (evaluaci). Cesty v rámci CD viz tabulka C.1.

A.4 Použití

Samotnou predikci ceny ropy pro určité datum provedeme následujícím příkazem s argumentem predikovaného data. Pokud nezadáme datum, algoritmus predikuje pro aktuální datum. Pokud bychom měli jiný model, podle kterého bychom chtěli predikovat, zadáme cestu k souboru, ve kterém je uložen (první argument), a model bude překopírován na správné místo a použije se při predikci. Momentálně je algoritmus nastaven tak, aby predikoval týden dopředu z článků minulých třiceti dnů (protože byl na této konfiguraci evaluován). Toto musíme vzít v potaz při posuzování důvěryhodnosti predikce.

```
java -ea -classpath COPF.jar copf.Predict [-model {cesta k modelu}]
   [-date yyyy-MM-dd]
```

Výstupem predikce je jedna z odpovědí „up“, „stag“, „down“, parametry predikce (predikované datum, délka predikce) a počet článků, z kterých se predikovalo. Zde uvádíme příklad:

```
predicted date: 2011-08-01
forecast length: 7 days
text count: 127
prediction result: down
```

Kromě predikce můžeme také evaluovat zadané atributy (pokud nezadáme žádný argument). Pokud bychom chtěli evaluovat již vytvořený model, zadáme cestu k souboru, ve kterém je uložen (první argument), a model bude překopírován na správné místo a ohodnocen.

```
java -ea -classpath COPF.jar copf.featuremanagement.FeatureEvaluator
    [-model {cesta k modelu}]
```

Výstupem evaluace je tabulka s reálnými (v řádcích) a predikovanými (ve sloupcích) hodnotami, úspěšností a baseline (definováno v 9.2). Zde je ukázka tabulky:

```
          down      stag      up        no_value
down      27        0         90        252
stag      10        0         29        517
up        38        0         121       275
Succes: 0,4698
Baseline: 0,5048
```

A.5 Nástroje

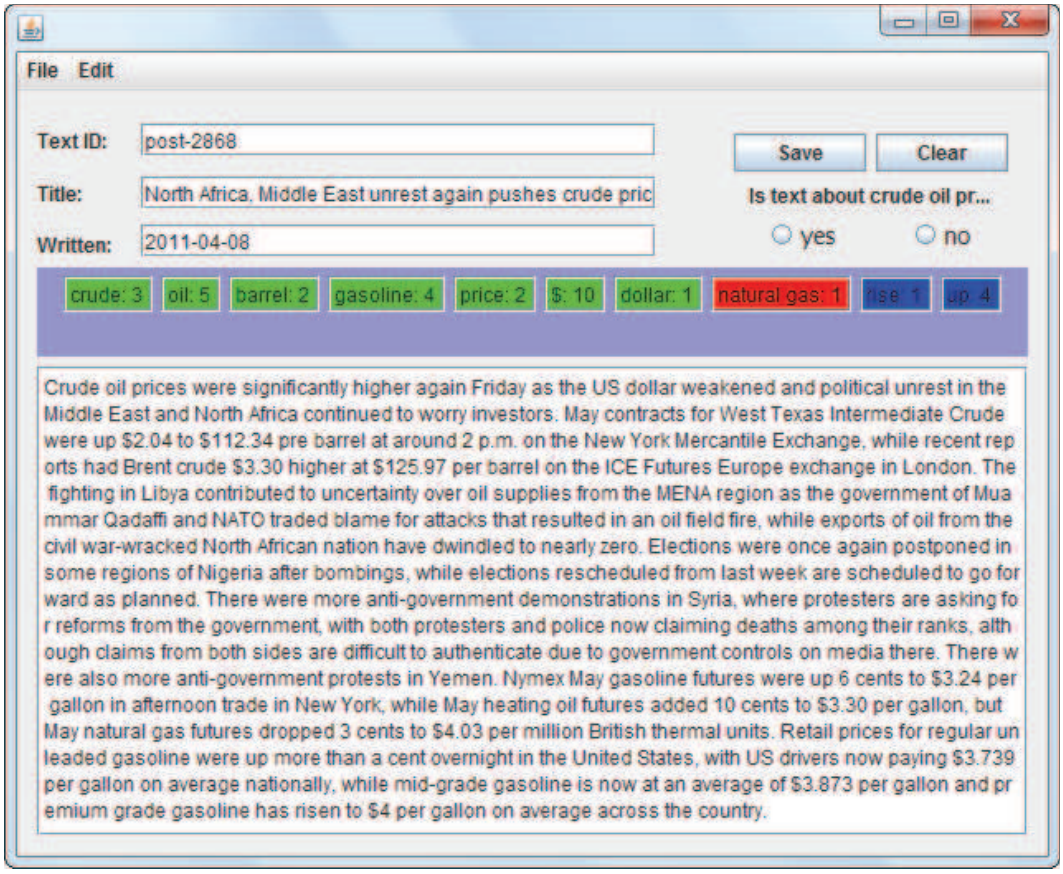
V rámci projektu COPF existuje několik užitečných nástrojů pro usnadnění práce.

A.5.1 Nástroje pro filtraci

Nástroj pro manuální evaluaci textů spustíme tímto příkazem:

```
java -ea -classpath COPF.jar copf.utilities.manualevaluation.
    TextFilterManEvalWindow
```

Pro manuální evaluaci je nutné nahrát jeden z XML souborů, ve kterých jsou uloženy texty. Tento soubor nahrajeme na horní liště pomocí File > Open. Poté uvidíme např.:



Obrázek A.1: Nástroj pro manuální ohodnocení textů pro filtraci

Nahoře jsou tři pole s identifikátorem, titulkem a datem vytvoření textu. V dolní části je samotný text. Toto všechno můžeme editovat.

Uprostřed je modrý box, v kterém se objevují některá důležitá slova pro usnadnění evaluace. Zelená slova jsou ta, která ve většině případů indikují, že se text zabývá cenou ropy. Červená slova většinou indikují, že se text nezabývá cenou ropy, a modrá barva indikuje další pro nás zajímavá slova.

V pravé horní části vidíme tlačítka „yes“ a „no“. Těmito tlačítky určujeme, zda se článek zabývá cenou ropy. Tlačítkem „Clear“ vymažeme změny od posledního uložení a tlačítkem „Save“ uložíme všechny změny a náhodně se nám načte další neohodnocený článek (stejně jako po nahrání souboru). Pokud už žádný takový není, program to ohlásí.

A.5.2 Nástroje pro vizualizaci

V rámci projektu umíme prezentovat některá data lepší cestou, než jen zkoumáním XML souborů.

Náš první nástroj ukáže graf vývoje ceny ropy:

```
java -ea -classpath COPF.jar copf.visualization.
   featureevalvisualization.PriceHistVis
```

Další nástroj ukáže výsledky poslední evaluace. Výstupem je tabulka s reálnými a predikovanými hodnotami (viz výstup evaluace A.4). Spustíme ho tímto příkazem:

```
java -ea -classpath COPF.jar copf.visualization.
   featureevalvisualization.ContingencyTableVis
```

Protože se naše výsledky mohou pro každé spuštění mírně lišit (kvůli náhodnému výběru článků ve fázi undersamplingu), vytvořili jsme další nástroj, který ukáže průměrný výsledek pro všechny provedené evaluace, které jsou uloženy v souboru „{cesta k datům}/Features/feature_evaluation.xml“. Výstupem je stejná tabulka jako v předchozím případě. Nástroj spustíme tímto příkazem:

```
java -ea -classpath COPF.jar copf.visualization.
   featureevalvisualization.AverageContTableVis
```

B. Programátorská dokumentace

COPF (Crude oil price forecast) je systém pro odhad vývoje ceny ropy na základě textových zpravodajských informací. Poskytuje také nástroje pro vylepšení tohoto odhadu.

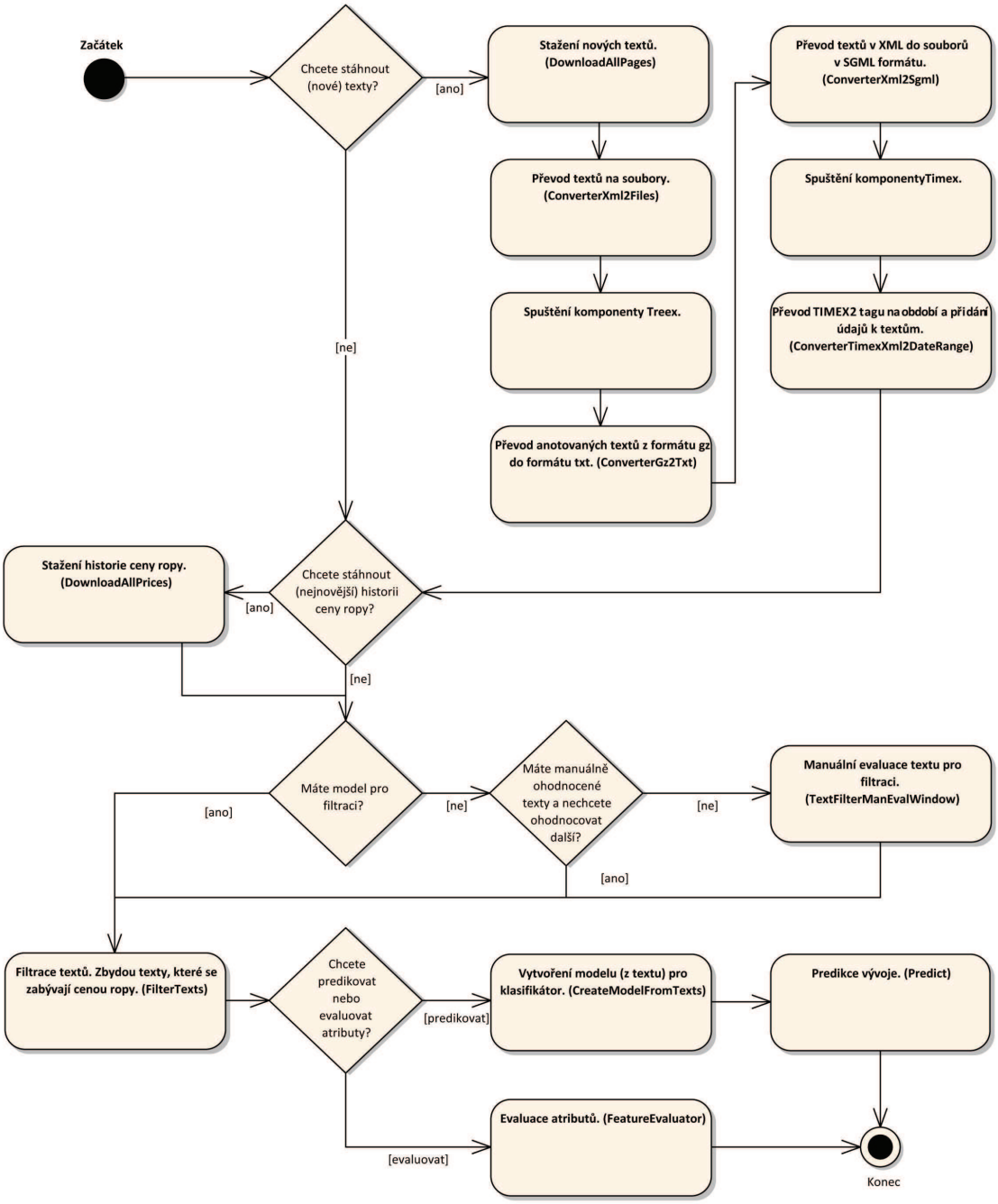
Systém COPF je napsaný v programovacím jazyce Java (JDK 1.7) ve vývojovém prostředí Netbeans 7.1. Externí komponenty jsou napsány též v jazycích Perl a Python. COPF je programován pro operační systém Linux.

B.1 Architektura systému

Celý systém je v momentálním stavu uspořádán spíše jako soubor skriptů, které je nutné volat ve správném pořadí. Důvodů pro toto uspořádání je více: v první řadě experimentální povaha projektu, kdy se struktura použití ještě často obměňuje, dále pak časté použití (nahraditelných) externích komponent v jiných programovacích jazycích, včetně komponenty Treex spouštěné na zcela jiných počítačích katedry, a též vysoká časová náročnost některých kroků v rámci celé predikce či evaluace.

B.1.1 Diagram aktivit

Pro přehled toho, co se provádí v rámci evaluace a predikce, jsme vytvořili diagram aktivit (viz B.1).



Obrázek B.1: Diagram aktivit

B.2 Rozšíření systému

Systém je připraven na dva druhy rozšíření. Prvním typem je rozšíření funkcí systému a tím druhým je přidání zdrojů dat.

B.2.1 Rozšíření funkčnosti

Do tohoto typu rozšíření spadá přidání nových typů atributů pro predikci a přidání vizualizací dat.

Nové typy atributů pro predikci

Přidání nového typu atributů pro predikci docílíme poděděním od abstraktní třídy FeatureType v balíčku copf.featuremanagement a přidáním nové položky do proměnné featTemplTypes ve třídě FeatureManager v témže balíčku. V rámci tohoto přidání definujeme řetězec, který bude reprezentovat nový typ v souborech s atributy.

Přidání nové vizualizace

Novou vizualizaci přidáme poděděním od abstraktní třídy Visualization v balíčku copf.visualization.featureevalvisualization. Vizualizace spouštíme jako samostatné skripty.

B.2.2 Přidání zdrojů dat

V rámci projektu můžeme přidávat nové zdroje dat pro texty a historii ceny ropy.

Přidání nového zdroje textů

Pro přidání nového zdroje textů musíme podědit od abstraktní třídy DownloadPage v balíčku copf.downloaders.pagedownloaders a implementovat potřebné parsování zdroje za účelem získání pro nás důležitých informací. Mnoho potřebných funkcí můžeme nalézt ve třídě Downloader v balíčku copf.downloaders. Nakonec je nutné přidat volání metody download na této třídě do funkce main třídy DownloadAllPages v balíčku copf.downloaders.pagedownloaders.

Přidání nového zdroje historie ceny ropy

Pro přidání nového zdroje historie ceny ropy musíme podědit od abstraktní třídy DownloadPrices v balíčku copf.downloaders.pricedownloaders a implementovat potřebné parsování zdroje za účelem získání pro nás důležitých informací.

Mnoho potřebných funkcí můžeme nalézt ve třídě Downloader v balíčku copf.downloaders. Nakonec je nutné přidat volání metody download na této třídě do funkce main třídy DownloadAllPrices v balíčku copf.downloaders.pricedownloaders.

B.3 Externí komponenty

Externí komponenty mají své vlastní dokumentace, proto u nich popíšeme pouze důležité části, které jsme používali.

B.3.1 Timextag

V rámci komponenty TimexTag jsme používali tyto tři skripty, které najdeme v podadresáři ./TIMEX2_system/timextag-0.3/timextag/ adresáře components (cesta k adresáři components v rámci CD je uvedena v tabulce C.1).

Vstupem komponenty TimexTag jsou SGML soubory, každý tento soubor je třeba rozdělit na dvě části, soubor s XML tagy (prázdné jsou vynechány) a soubor s textovým obsahem původního SGML souboru zbavený XML tagů a jakýchkoli komentářů, procesních instrukcí atd. Toto rozdělení provádíme tímto příkazem:

```
python splitSgml.py \
      {cesta k souboru} \
      {cesta k výstupnímu XML souboru} \
      {cesta k výstupnímu textovému souboru}
```

Dále je potřeba udělat syntaktický rozbor textů. Pro toto se používá Charniakův parser, jehož výstupem je XML s rozborem. Toto se dělá příkazem:

```
python parseCharniak.py \
      {cesta ke složce, ve které jsou textové soubory vytvořené
         pomocí splitSgml.py} \
      {cesta ke složce, ve které jsou XML soubory vytvořené pomocí
         splitSgml.py} \
      {cesta ke složce, do které budou uloženy výstupní XML soubory}
```

Nakonec, pokud máme všechno připraveno, používáme pro určení časových údajů skript timextool.py. Tento skript provede časovou analýzu textu a vrátí XML, které je ve standardu TIMEX2.

```
python timextool.py \
      --inputtxt {cesta ke složce, ve které jsou textové soubory
         vytvořené pomocí splitSgml.py} \
      --inputxml {cesta ke složce, ve které jsou XML soubory
         vytvořené pomocí splitSgml.py} \
      --inputparses {cesta ke složce, ve které jsou soubory
         vytvořenému pomocí parseCharniak.py} \
      --all \
```

```
--outputxml {cesta ke složce, do které budou uloženy výstupní
    XML soubory}/
```

Více informací nalezneme v samotných souborech a v souborech README a Examples.txt, které najdeme v podadresáři ./TIMEX2_system/timextag-0.3/ adresáře components (cesta k adresáři components v rámci CD je uvedena v tabulce C.1).

B.3.2 Treex

V rámci komponenty Treex jsme se zabývali pouze formátem vstupních a výstupních souborů, protože samotný rozbor pomocí této komponenty byl prováděn externě. Vstupem komponenty Treex byly soubory s texty, které jsme potřebovali ohodnotit (v každém souboru jeden text). Výstupem byly soubory zabalené ve formátu gz, opět co soubor, to jeden gz archiv. Více informací o komponentě je k nalezení v literatuře [8], [9], [10] a [11]).

B.4 Externí knihovny

Kromě externích komponent jsme použili i další externí knihovny napsané v jazyce Java.

B.4.1 OpenNLP MaxEnt

OpenNLP MaxEnt je knihovna implementující MaxEnt klasifikátor s příznivým uživatelským rozhraním. Knihovna obsahuje třídy pro snadné načítání dat, trénování modelu a samotnou klasifikaci. Zde je několik nejdůležitějších tříd:

- MaxentModel je rozhraní pro MaxEnt modely.
- AbstractModelWriter je abstraktní třída, která se stará o serializaci modelu.
- AbstractModelReader je abstraktní třída, která se stará o načtení modelu ze souboru.
- GISModel je implementace rozhraní MaxentModel, kterou používáme.

B.4.2 JFreeChart

Tato knihovna slouží k vizualizaci dat pomocí grafů. Zatím ji používáme pouze na zobrazení vývoje ceny ropy. Problematické je, že dokumentace k této knihovně není volně ke stažení (samotná knihovna je zdarma, ale dokumentace je zpoplatněna). Protože jsme se k dokumentaci nedostali, zmíníme pouze jednu třídu:

- XYPlot je třída sloužící k vykreslení grafu funkce.

B.4.3 HtmlCleaner

Knihovna HtmlCleaner je analyzér jazyka HTML. Protože je HTML na Internetu často špatně zformované a nepoužitelné pro další zpracování, je ho potřeba nejdříve správně zformovat, správně seřadit tagy atd. Právě k tomuto slouží HtmlCleaner. HtmlCleaner při čištění HTML kódu zároveň vytváří DOM strukturu, což je výhodné pro následné dotazování.

V této knihovně pro nás byly důležité tyto dvě třídy:

- TagNode je třída reprezentující jeden uzel DOM struktury.
- PrettyXmlSerializer je třída převádějící DOM strukturu opět na řetězec.

B.5 Důležité datové struktury

V systému COPF existují kromě externích knihoven ještě dvě komponenty. Méně důležitou je knihovna, která se nazývá SkalCore a obsahuje pomocné třídy a jednoduché funkce, které nám chyběly ve standardních knihovnách jazyka Java. Dále jsme zde obalili některé třídy, často z důvodu ošetření výjimek, abychom zpřehlednili kód v důležitých třídách hlavního projektu. Tento projekt se nazývá COPF a obsahuje tyto důležité balíčky a třídy:

B.5.1 copf

Balíček copf umožňuje samotnou predikci a důležité akce potřebné k jejímu provedení.

- CreateModel je třída pro tvorbu MaxEnt modelu z datového souboru.
- CreateModelFromTexts je třída pro tvorbu MaxEnt modelu z textů.
- FilterTexts je třída pro filtraci textů.
- Paths je třída, ve které jsou uloženy cesty v rámci projektu. Kromě souboru paths.txt, který může být v kořenovém adresáři projektu, zde můžeme měnit cesty k datům a komponentám.
- Predict je třída pro predikci z textů.
- Standards je třída, která obsahuje standardy v rámci projektu (např. formát data a času).

B.5.2 copf.downloadedtexts

Balíček copf.downloadedtexts slouží k snadnějšímu přístupu ke staženým článkům uloženým v XML. Balíček obsahuje třídy vygenerované ze souboru s XML Schematu definujícím uložení textů v XML souborech. K těmto třídám je přidáno několik potřebných atributů pro zpracování.

B.5.3 copf.downloaders

Balíček copf.downloaders obsahuje třídy pro stahování dat z Internetu.

- Downloader je třída obsahující potřebné funkce pro stahování textů.

B.5.4 copf.downloaders.pagedownloaders

Balíček copf.downloaders.pagedownloaders obsahuje třídy pro stahování textů z Internetu.

- DownloadAllPages je třída, která slouží ke stahování textů ze všech zdrojů, pro které je vytvořen skript ke stáhnutí.
- DownloadPage je abstraktní třída, od které jsou poděděny všechny skripty pro stahování textů.

B.5.5 copf.downloaders.pricedownloaders

Balíček copf.downloaders.pricedownloaders obsahuje třídy pro stahování historie ceny ropy z Internetu.

- DownloadAllPrices je třída, která slouží ke stahování historie ceny ropy ze všech zdrojů, pro které je vytvořen skript ke stáhnutí. Momentálně to je z jednoho zdroje.
- DownloadPrices je abstraktní třída, od které jsou poděděny všechny skripty pro stahování historie ceny ropy.

B.5.6 copf.exportformatparser

Balíček copf.exportformatparser poskytuje snadnou práci s texty v Exportním formátu komponenty Treex.

- ExportFormatSentence je třída reprezentující větu z textu, který je výstupem komponenty Treex.
- ExportFormatText je třída reprezentující text, který je výstupem komponenty Treex.

B.5.7 copf.featuremanagement

Balíček copf.featuremanagement slouží k práci se sadami atributů (nahrávání, hledání, ohodnocování atd.).

- FeatureEvaluator je třída pro ohodnocení konfigurací atributů. Je v ní implementována křížová evaluace.
- FeatureInvestigator je třída pro hledání atributů v textech.
- FeatureManager je třída sloužící k načítání atributů a hledání jejich výskytů v textech.
- FeatureType je abstraktní třída, od které jsou poděděny všechny typy atributů, které jsou vyhledávány v textech.
- PredictionResult je třída určená k uchovávání výsledků a informací o predikci.

B.5.8 copf.pricehistorymanagement

Balíček copf.pricehistorymanagement slouží k práci s historií ceny ropy.

- OutcomeArbiter je třída sloužící k přiřazování hodnot z množiny výsledků změně ceny za nějaké období.
- PriceHistoryManager je třída pro přístup k historii cen ropy.

B.5.9 copf.utilities.manualevaluation

Balíček copf.utilities.manualevaluation slouží k manuální evaluaci textů.

- TextFilterManEvalWindow je třída sloužící k manuální evaluaci textů pro filtraci.

B.5.10 copf.utilities.textsconverters

Balíček copf.utilities.textsconverters slouží k převádění formátů dat v rámci projektu.

- ConverterGz2Txt je třída pro rozbalování textů zabalených do formátu gz.
- ConverterTimexXml2DateRange je třída převádějící XML výstup z komponenty TimexTag na rozmezí dvou dat.

- ConverterXml2Files je třída pro převod textů z XML do textových souborů (každý text v samostatném souboru).
- ConverterXml2Sgml je třída pro převod textů z XML do SGML souborů (každý text v samostatném souboru).

B.5.11 copf.visualization.featureevalvisualization

Balíček copf.visualization.featureevalvisualization slouží k vizualizaci dat v rámci projektu.

- Visualization je abstraktní třída, od které dědí všechny vizualizace v rámci projektu.
- ContingencyTableVis je třída pro vizualizaci kontingenční tabulky pro evaluaci atributů.
- AverageContTableVis je třída pro vizualizaci kontingenční tabulky pro průměrné hodnoty ze všech evaluací atributů.
- PriceHistVis je třída pro vizualizaci vývoje ceny ropy v minulosti.

B.6 Data

Pro COPF je velmi důležitá práce s daty. Formáty dat a jejich popis jsme rozčlenili podle toho, k čemu slouží. V rámci příkladů formátů pro lepší přehlednost používáme stále stejný text.

B.6.1 TimexTag

TimexTag je externí komponenta popsaná výše (viz B.3.1). Vstupnem pro ni jsou SGML soubory. Tyto soubory jsou uloženy v podsložkách adresáře Texts (cesta v rámci CD je uvedena v tabulce C.1). Celý obsah soubory obaluje tag „DOC“ v němž jsou zanořeny tyto tagy:

- Tag „DATETIME“, který určuje referenční datum pro všechny časové údaje v textu.
- Tag „TITLE“, v kterém je uložen titulek textu.
- Tag „TEXT“, v kterém je uložen samotný text (bez titulku).

Zde je příklad jednoho takového souboru:

```
<DOC>
<DATETIME>31.01.2008</DATETIME>
<TITLE>Factoid on Oil Rigs</TITLE>
<TEXT>
In December of 2007, 2290 oil rigs were in operation in North
   America, 37 less than a year before.
</TEXT>
</DOC>
```

Výstupem komponenty je XML soubor ve standardu TIMEX2 (více viz [13]), který je uložený v podadresáři xmloutput adresáře, v kterém je vstupní SGML soubor. Zde je příklad jednoho souboru:

```
<?xml version="1.0" encoding="UTF-8"?>
<DOC generator="timexdoc.py">
<reftime rstart="1" rend="10">
</reftime>
<TEXT rstart="32" rend="131">
<TIMEX2 set="" rend="43" tmxclass="point" rstart="36" dirclass="
   before" parsenode=".1 p4" prenorm="|dex|Y|XXXX-12">December</
   TIMEX2>
<TIMEX2 set="" rend="51" val="2007" tmxclass="point" rstart="48"
   dirclass="before" parsenode=".1 p8" prenorm="|fq|_2007">2007</
   TIMEX2>
<TIMEX2 set="" rend="57" val="2290" tmxclass="point" rstart="54"
   dirclass="before" parsenode=".1 w5" prenorm="|fq|_2290">2290</
   TIMEX2>
</TEXT>
</DOC>
```

B.6.2 Treex

Komponenta Treex potřebovala na vstupu textové soubory, které jsou uloženy v podsložkách složky DownloadedTextsInFiles (cesta v rámci CD je uvedena v tabulce C.1):

```
Factoid on Oil Rigs

In December of 2007, 2290 oil rigs were in operation in North
   America, 37 less than a year before.
```

Pro každý vstupní soubor byl výstupem soubor komprimovaný pomocí programu Gzip. Po rozbalení do odpovídající podsložky složky AnotatedTextsInFiles/txt (cesta v rámci CD je uvedena v tabulce C.1) soubor vypadal například takto (více viz [11]):

```
DownloadedTextsInFiles/oil_prize_net/FactoidonOilRigs20080131.parsed
   .treex.gz:s1  Factoid|factoid|NN|1|0|ExD on|on|IN|2|1|AuxP Oil|
```

```
   Oil|NNP|3|4|Atr Rigs|Rigs|NNP|4|2|Atr  factoid|PRED|1|0|complex|n
   :|n.denot|sg|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- Oil|RSTR|2|3|
   complex|n:attr|n.denot|sg|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|-
   Rigs|LOC|3|1|complex|n:on+X|n.denot|sg
   |-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|-  0-0 2-1 3-2 1-2
DownloadedTextsInFiles/oil_prize_net/FactoidonOilRigs20080131.parsed
   .treex.gz:s2  In|in|IN|1|9|AuxP December|December|NNP|2|1|Adv of|
   of|IN|3|2|AuxP 2007|2007|CD|4|3|Atr ,|,|,|5|9|AuxX 2290|2290|CD
   |6|8|Atr oil|oil|NN|7|8|Atr rigs|rig|NNS|8|9|Sb were|be|VBD|9|0|
   Pred in|in|IN|10|9|AuxP operation|operation|NN|11|10|Adv in|in|IN
   |12|9|AuxP North|North|NNP|13|14|Atr America|America|NNP|14|12|
   Adv ,|,|,|15|14|AuxX 37|37|CD|16|14|Atr less|less|JJR|17|12|AuxP
   than|than|IN|18|17|AuxP a|a|DT|19|20|AuxA year|year|NN|20|17|NR
   before|before|IN|21|20|AuxC .|.|.|22|0|AuxK  December|TWHEN|1|6|
   complex|n:in+X|n.denot|sg|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|-
   2007|RSTR|2|1|complex|n:of+X|n.quant.def
   |-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- 2290|RSTR|3|5|complex|n:
   attr|n.quant.def|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- oil|RSTR
   |4|5|complex|n:attr|n.denot|sg
   |-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- rig|ACT|5|6|complex|n:subj
   |n.denot|pl|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- be|PRED|6|0|
   complex|v:fin|v|-|neg0|ant|ind|decl|-|-|-|-|disp0|-|it0|-|-|res0
   |-|-|1|-|- operation|LOC|7|6|complex|n:in+X|n.denot|sg
   |-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- North|RSTR|8|9|complex|n:
   attr|n.denot|sg|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- America|LOC
   |9|6|complex|n:in+X|n.denot|sg
   |-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- 37|RSTR|10|9|complex|n:
   attr|n.quant.def|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- year|CPR
   |11|9|complex|n:less_than_before+X|n.denot|sg
   |-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|0|-|- 1-0 3-1 5-2 6-3 7-4 8-5
   10-6 12-7 13-8 15-9 19-10 0-0 2-1 4-5 9-6 11-8 14-8 16-10 17-10
   18-10 20-10
```

B.6.3 Ceny ropy

Historie ceny ropy je ukládána v prostém textovém souboru s názvem price_history_all.txt, který je ve složce DownloadedPriceHistory (cesta v rámci CD je uvedena v tabulce C.1). Jedna cena za určité období je uložena na samostatný řádek ve formátu: {datum}|{cena ropy za barel}. Datum je ve formátu podle normy ISO 8601 a cena je udávána v dolarech. Zde je krátký příklad:

```
1986-01-03|25.78
1986-01-10|25.99
1986-01-17|24.57
1986-01-24|20.31
1986-01-31|19.69
```

K ceně ropy se vztahuje i definice množiny našich výsledků. Tato množina se definuje v souboru available_outcomes.txt ve složce Features (cesta v rámci CD je uvedena v tabulce C.1). V souboru se definují názvy výsledků a jejich rozmezí.

Každý řádek má tvar „{název výsledku} < {horní hranice rozmezí}“. Jen na poslední řádce je pouze název výsledku, protože horní hranice rozmezí je nekonečno. Spodní hranice rozmezí daného výsledku je definována horní hranicí předchozího výsledku. První výsledek má spodní hranici nekonečno.

Výsledek se potom určuje podle toho, do kterého rozmezí padne tato hodnota: {cena v období, které předpovídáme} - {cena v období, z kterého předpovídáme}. Takto vypadá náš aktuální soubor:

```
down < -0.5
stag < 0.5
up
```

B.6.4 Stahování textů

Texty jsou stahovány do XML souborů. Pro každý zdroj je vytvořena jedna podsložka s jedním XML souborem ve složce DownloadedTextsInXml (cesta v rámci CD je uvedena v tabulce C.1). Kořenovým tagem XML je tag „texts“, jenž obsahuje datum vytvoření, tag „created“ a texty, kde každý text je reprezentován tagem „text“. Ten obsahuje po stažení tyto tagy:

- Tag „id“, který představuje unikátní id textu.
- Tag „url“ obsahuje URL, ze kterého byl článek stažen.
- Tag „title“, v kterém je uložen titulek textu.
- Tag „date“, který určuje datum a čas vytvoření textu (formátováno podle standardu ISO 8601).
- Tag „content“, v kterém je uložen samotný text (bez titulku).

Po zjištění časových údajů pomocí komponenty TimexTag a filtraci přibudou pro článek tyto tagy:

- Tag „determined_date_from“, který určuje začátek období, kterým se zabývá (formátováno podle standardu ISO 8601).
- Tag „determined_date_to“, který určuje konec období, kterým se zabývá (formátováno podle standardu ISO 8601).

- Tag „filter_text“, který určuje, zda text prošel filtrem či nikoliv („yes“ znamená, že se zabývá vývojem ceny ropy, „no“ znamená, že se nezabývá vývojem ceny ropy).

Dále, pokud byl článek manuálně ohodnocen pro filtraci, přibude ještě tag „manualevaluation“, v kterém je vnořen ještě jeden tag „filter“ (hodnota „yes“ znamená, že se zabývá vývojem ceny ropy, „no“ znamená, že se nezabývá vývojem ceny ropy).

Zde je příklad XML souboru s jedním staženým článkem:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<texts>
  <text>
    <id>FactoidonOilRigs20080131</id>
    <url>http://www.oil-price.net/en/articles/oil-rigs.php</url>
    <title>Factoid on Oil Rigs</title>
    <date>2008-01-31</date>
    <content>In December of 2007, 2290 oil rigs were in operation in
        North America, 37 less than a year before.</content>
    <determined_date_from>2007-01-01</determined_date_from>
    <determined_date_to>2290-12-31</determined_date_to>
    <filter_text>no</filter_text>
  </text>
  <created>2011-07-28T21:27:31</created>
</texts>
```

```
Factoid on Oil Rigs

In December of 2007, 2290 oil rigs were in operation in North
   America, 37 less than a year before.
```

B.6.5 Atributy

Konfigurace atributů pro filtraci i pro predikci jsou uloženy ve stejném formátu. Konfiguraci atributů pro filtraci najdeme ve složce Features/Filter v souboru features.txt. Konfiguraci atributů pro predikci najdeme ve složce Features v souboru features.txt (cesty v rámci CD jsou uvedeny v tabulce C.1).

Každá řádka souboru s atributy je prázdná nebo obsahuje komentář nebo obsahuje atribut. Komentář se pozná podle symbolu # na začátku řádky. Atribut se může skládat ze dvou částí, první je typ atributu a druhou tvoří parametry atributu. První tabulátor na řádce odděluje typ od parametrů. Typem atributu je například „set_text“ (množina slov ve větě). Parametry jsou od sebe odděleny mezerou. Jsou to tokeny, které hledáme v textu. Tokeny mohou být tří druhů: slovo v základním formátu, a-lemma, t-lemma (více viz 5). A-lemma (t-lemma) se

skládá z mnoha částí oddělených svislítkem. Pokud nechceme specifikovat všechny části, můžeme vynechat dané místo mezi dvěma svislítky.

Soubor s atributy může vypadat například takto:

```
# up
set_text  _T_gain||||||||post||||||||||||||||
set_text  grow
set_sentence  barrel cent up

# down
set_text  decrease
set_text  fall

# stagnation
set_text  average
```

Připomínáme, že při tvorbě konfigurace atributů člověk neurčuje, jestli atribut indikuje vzrůst či pokles ceny. To se naučí program sám až podle skutečného vývoje ceny.

Výsledky evaluace atributů pro predikci se zapisují do XML souboru feature_evaluation.xml ve složce Features (cesty v rámci CD jsou uvedeny v tabulce C.1).

Kořenovým tagem souboru je tag „evaluation_history“, který obsahuje tagy „evaluation“. Tag „evaluation“ představuje jednu křížovou evaluaci atributů a obsahuje tyto tagy:

- Tag „evaluation_date“, který určuje datum a čas evaluace (formátováno podle standardu ISO 8601).
- Tag „forecast_length“, který představuje délku predikce ve dnech.
- Tag „tested_range“, který představuje šířku období (ve dnech) pro jednu predikci.
- Tag „tested_dates_per_eval“, který určuje počet týdnů predikovaných na jedno naučení klasifikátoru.
- Tag „results“, který obsahuje výsledky evaluace. Každý výsledek je reprezentován tagem „result“.

Tag „result“ obsahuje tyto tagy:

- Tag „article_count“, který představuje počet článků, z kterých se předpovídalo.

- Tag „prediction_date“, který určuje předpovídané datum (formátováno podle standardu ISO 8601).
- Tag „real_value“, který představuje výsledek podle reálného vývoje.
- Tag „outcomes“, který obsahuje všechny hodnoty z množiny řešení a jejich procentuální ohodnocení. Každá hodnota je reprezentována tagem „outcome“.
  - Tag „outcome“ má dva atributy. Atribut „name“ udává jméno hodnoty z množiny řešení a atribut „value“ udává předpovězenou pravděpodobnost pro danou hodnotu.

Zde uvádíme jako příklad výstup evaluace pro dva týdny (zbytek týdnů jsme smazali pro úsporu místa):

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<evaluation_history>
    <evaluation>
        <evaluation_date>2012-02-24T00:00:04</evaluation_date>
        <forecast_length>7</forecast_length>
        <tested_range>30</tested_range>
        <tested_dates_per_eval>10</tested_dates_per_eval>
        <results>
            <result>
                <article_count>12</article_count>
                <prediction_date>2011-08-26</prediction_date>
                <real_value>up</real_value>
                <outcomes>
                    <outcome name="down" value
                        ="0.3730728348183343"/>
                    <outcome name="up" value="0.508670087302741"/>
                    <outcome name="stag" value
                        ="0.1182570778789246"/>
                </outcomes>
            </result>
            <result>
                <article_count>2</article_count>
                <prediction_date>2011-09-02</prediction_date>
                <real_value>up</real_value>
                <outcomes>
                    <outcome name="down" value
                        ="0.35655478493486514"/>
                    <outcome name="up" value="0.5388129496896914"/>
                    <outcome name="stag" value
                        ="0.1046322653754435"/>
                </outcomes>
```

```
            </result>
        </results>
    </evaluation>
</evaluation_history>
```

B.6.6 MaxEnt

V rámci predikce je nutné připravit pro klasifikátor vstupní data. Tato data se automaticky připravují z textů pomocí konfigurace atributů. Pro klasifikátor je potřeba soubor s daty pro natrénování a soubor s daty, podle kterých se bude predikovat.

Soubor pro natrénování se skládá z atributů všech textů, které jsme vybrali k natrénování klasifikátoru. Každý článek je v souboru reprezentován jednou řádkou, na které jsou atributy daného textu oddělené mezerou (každý atribut je nahrazen unikátním číselným identifikátorem). Za poslední mezerou je správná hodnota z množiny výsledků (počítáno z historie ceny ropy), tj. aktuálně jedna z hodnot „up“, „stag“, „down“. Viz příklad:

```
16=1.0 11=1.0  79=1.0  down
14=1.0  79=1.0  down
66=1.0 11=1.0  79=1.0  62=1.0  down
11=1.0  79=1.0  down
68=1.0 11=1.0  79=1.0  down
79=1.0  stag
37=1.0 47=1.0  79=1.0  down
11=1.0  79=1.0  down
66=1.0 37=1.0 8=1.0 11=1.0  79=1.0  62=1.0  up
79=1.0  up
```

Soubor, podle kterého klasifikátor predikuje, se liší od souboru pro natrénování pouze v řetězci za poslední mezerou na řádce. Za poslední mezerou v tomto souboru je znak „?“.

```
16=1.0 18=1.0  81=1.0  ?
50=1.0 11=1.0 47=1.0  81=1.0  ?
16=1.0 11=1.0  81=1.0  ?
17=1.0 50=1.0 19=1.0 47=1.0  81=1.0  ?
11=1.0 14=1.0  81=1.0  ?
```

C. Obsah CD

Na CD přiloženém k bakalářské práci je samotná práce ve formátu PDF, COPF systém se všemi externími komponentami kromě komponenty Treex, zdrojové kódy aplikace, programátorská dokumentace, data a další. Zde jsou důležité cesty v rámci CD:

cesta	obsah
`./BakalarskaPrace.pdf`	bakalářská práce (tento dokument)
`./Program/`	kořenová složka programu
`./Program/components/`	cesta k externím komponentám systému
`./Program/components/TIMEX2_system/`	cesta k TIMEX2 systému
`./Program/components/TIMEX2_system/find_all_timexes.sh`	cesta k skriptu spouštějícímu časové určení pro texty
`./Program/components/TIMEX2_system/Texts/`	cesta k SGML souborům, které jsou vstupem komponenty TimexTag
`./Program/java_src/`	zdrojové kódy napsané v jazyce Java
`./Program/javadoc/`	vygenerovaná dokumentace hlavního projektu v systému COPF
`./Program/lib/`	složka s potřebnými knihovnami
`./Program/COPF.jar`	spouštěcí soubor pro všechny skripty v projektu napsané v jazyku Java
`./Program/paths.txt`	soubor upravující cesty k složkám v rámci projektu
`./Program/data/DataForExperiment/`	data použitá při experimentu
`./Program/data/LastEvalData/`	data použitá při závěrečné evaluaci
`./Program/data/Models/filtering_model.txt`	model pro filtraci
`./Program/data/Models/prediction_model.txt`	model pro predikci
`./Program/data/NewData/`	složka pro nová data
Další cesty jsou v rámci jedné z datových složek DataForExperiment, LastEvalData, NewData:	
`./DownloadedTextsInFiles/`	složka s texty v souborech, které používá komponenta Treex
`./DownloadedTextsInXml/`	složka s texty v XML souborech, jejichž součástí jsou další doplňující údaje
`./AnotatedTextsInFiles/txt/`	složka s oanotovanými texty v souborech
`./DownloadedPriceHistory/`	složka s historií ceny ropy
`./Features/`	složka se soubory potřebnými pro atributy
`./Features/Filter/`	složka se soubory potřebnými pro atributy filtrace

Tabulka C.1: Cesty k důležitým souborům a složkám v rámci CD

D. Formáty souborů

D.1 Příklad SGML formátu

```
<DOC>
<DATETIME>2012-02-24</DATETIME>
<TITLE>Gold futures lower in Asian trade</TITLE>
<TEXT>
Forexpros - Gold futures were lower in Asian trade on Friday. On the
    Comex division of the New York Mercantile Exchange, Gold futures
    for April delivery traded at USD1779.45 a troy ounce at time of
   writing falling 0.38%. It earlier traded at a session low USD1778
   .85 a troy ounce. Gold was likely to find support at USD1730.25
   and resistance at USD1788.85. US Dollar Index, which tracks the
   performance of the greenback versus a basket of six other major
   currencies, rose 0.01% to trade at USD78.69. Elsewhere on the
   Comex, Silver for March delivery fell 0.74% to trade at USD35.293
    a troy ounce while Copper for March delivery rose 0.12% to trade
    at USD3.822 a pound.
</TEXT>
</DOC>
```

D.2 Příklad výstupu komponenty TimexTag

```
<?xml version="1.0" encoding="UTF-8"?>
<DOC generator="timexdoc.py">
<reftime rstart="1" rend="10" val="2011-12-12">
<TIMEX2 rstart="1" rend="10" val="2011-12-12">2011-12-12</TIMEX2>
</reftime>
<TEXT rstart="43" rend="1503">
<TIMEX2 set="" rend="84" val="2011-12-12" tmxclass="point" rstart
   ="79" dirclass="same" parsenode=".1 p16" prenorm="|dex|W|XXXX-WXX
   -1">Monday</TIMEX2>
<TIMEX2 set="" rend="334" val="P3M" anchor_dir="BEFORE" tmxclass="
   duration" rstart="323" anchor_val="2011-12" parsenode=".2 p48"
   prenorm="P3M">three months</TIMEX2>
<TIMEX2 set="" rend="804" val="2010" tmxclass="point" rstart="788"
   dirclass="before" parsenode=".6 p3" prenorm="|dex|Y|_" mod="
   APPROX">much of last year</TIMEX2>
<TIMEX2 set="" rend="939" val="PAST_REF" anchor_dir="BEFORE"
   tmxclass="genpoint" rstart="932" anchor_val="2011-12-12" dirclass
   ="before" parsenode=".7 p11" prenorm="PAST_REF">
recently</TIMEX2>
```

<TIMEX2 set="" rend="1028" val="P1M" anchor_dir="ENDING" tmxclass="
duration" rstart="1015" anchor_val="2011-12" parsenode=".8 p23"
prenorm="P1M">the past month</TIMEX2>
</TEXT>
</DOC>

E. Sady atributů

E.1 Sada atributů pro 1. fázi experimentu

```
# up
set_text  gain
set_sentence  price rise
set_sentence  price climb
set_sentence  price increase
set_sentence  price skyrocket
set_sentence  price boost
set_sentence  price push higher
set_sentence  price growth
set_sentence  price grow
set_sentence  price go up
set_sentence  price high
set_sentence  price exceed
set_sentence  oil expensive
set_sentence  trade high
set_sentence  barrel high $
set_sentence  barrel cent up
set_sentence  barrel cent high

# down
set_text  loss
set_sentence  price decrease
set_sentence  price fall
set_sentence  price descent
set_sentence  price drop
set_sentence  price low
set_sentence  price go down
set_sentence  price cheap
set_sentence  price plunge
set_sentence  price decline
set_sentence  trade low
set_sentence  barrel low $
set_sentence  barrel cent drop
set_sentence  barrel cent down
set_sentence  barrel cent low

# stagnation
set_sentence  price stagnate
set_sentence  price stagnation
set_sentence  price settle
```

```
set_sentence  price average
set_sentence  price comparable
set_sentence  price stable
set_sentence  price constant
set_sentence  price fixed
set_sentence  price change _T_#Neg|||||||||||||||||||||||||
set_sentence  price alter _T_#Neg|||||||||||||||||||||||||
set_sentence  cost equal
set_sentence  cost same
```

E.2 Sada atributů pro 2. fázi experimentu

```
# up
set_text  gain
set_sentence  price rise
set_sentence  price raise
set_sentence  price climb
set_sentence  price increase
set_sentence  price skyrocket
set_sentence  price boost
set_sentence  price push higher
set_sentence  price growth
set_sentence  price grow
set_sentence  price go up
set_sentence  price high
set_sentence  price exceed
set_sentence  oil expensive
set_sentence  trade high
set_sentence  barrel up $
set_sentence  barrel high $
set_sentence  barrel cent up
set_sentence  barrel cent high

# down
set_text  loss
set_sentence  price decrease
set_sentence  price fall
set_sentence  price descent
set_sentence  price drop
set_sentence  price low
set_sentence  price go down
set_sentence  price cheap
set_sentence  price plunge
set_sentence  price decline
set_sentence  trade low
set_sentence  barrel drop $
```

```
set_sentence  barrel down $
set_sentence  barrel low $
set_sentence  barrel cent drop
set_sentence  barrel cent down
set_sentence  barrel cent low

# stagnation
set_sentence  price stagnate
set_sentence  price stagnation
set_sentence  price settle
set_sentence  price average
set_sentence  price comparable
set_sentence  price stable
set_sentence  price constant
set_sentence  price fixed
set_sentence  price change _T_#Neg|||||||||||||||||||||||||
set_sentence  price alter _T_#Neg|||||||||||||||||||||||||
set_sentence  cost equal
set_sentence  cost same
```

E.3 Poslední sada atributů

```
# clever configuration with past and future tense
# and with feature describing source

source

# up
set_text  _T_gain|||||||||post|||||||||||||||||
set_sentence  price _T_rise|||||||||post|||||||||||||||||
set_sentence  price _T_raise|||||||||post|||||||||||||||||
set_sentence  price _T_climb|||||||||post|||||||||||||||||
set_sentence  price _T_increase|||||||||post|||||||||||||||||
set_sentence  price _T_skyrocket|||||||||post|||||||||||||||||
set_sentence  price _T_boost|||||||||post|||||||||||||||||
set_sentence  price _T_push|||||||||post||||||||||||||||| higher
set_sentence  price growth
set_sentence  price _T_grow|||||||||post|||||||||||||||||
set_sentence  price _T_go|||||||||post|||||||||||||||||| up
set_sentence  price high
set_sentence  price _T_exceed|||||||||post|||||||||||||||||
set_sentence  price above average
set_sentence  oil expensive
set_sentence  _T_trade|||||||||post|||||||||||||||||| high
set_sentence  barrel up $
set_sentence  barrel high $
```

```
set_sentence  barrel cent up
set_sentence  barrel cent high
## nr
set_text  _T_gain|||||||||nr||||||||||||||||
set_sentence  price _T_rise|||||||||nr||||||||||||||||
set_sentence  price _T_raise|||||||||nr||||||||||||||||
set_sentence  price _T_climb|||||||||nr||||||||||||||||
set_sentence  price _T_increase|||||||||nr||||||||||||||||
set_sentence  price _T_skyrocket|||||||||nr||||||||||||||||
set_sentence  price _T_boost|||||||||nr||||||||||||||||
set_sentence  price _T_push|||||||||nr||||||||||||||||| higher
set_sentence  price _T_grow|||||||||nr||||||||||||||||
set_sentence  price _T_go|||||||||nr||||||||||||||||| up
set_sentence  price _T_exceed|||||||||nr||||||||||||||||
set_sentence  _T_trade|||||||||nr||||||||||||||||| high

# down
set_text  _T_loss|||||||||post||||||||||||||||
set_sentence  price _T_decrease|||||||||post||||||||||||||||
set_sentence  price _T_fall|||||||||post||||||||||||||||
set_sentence  price _T_descent|||||||||post||||||||||||||||
set_sentence  price _T_drop|||||||||post||||||||||||||||
set_sentence  price low
set_sentence  price _T_go|||||||||post||||||||||||||||| down
set_sentence  price cheap
set_sentence  price _T_plunge|||||||||post||||||||||||||||
set_sentence  price _T_decline|||||||||post||||||||||||||||
set_sentence  price below estimate
set_sentence  price below normal
set_sentence  price discount
set_sentence  trade low
set_sentence  barrel _T_drop|||||||||post||||||||||||||||| $
set_sentence  barrel down $
set_sentence  barrel low $
set_sentence  barrel cent _T_drop|||||||||post||||||||||||||||
set_sentence  barrel cent down
set_sentence  barrel cent low
## nr
set_text  _T_loss|||||||||nr||||||||||||||||
set_sentence  price _T_decrease|||||||||nr||||||||||||||||
set_sentence  price _T_fall|||||||||nr||||||||||||||||
set_sentence  price _T_descent|||||||||nr||||||||||||||||
set_sentence  price _T_drop|||||||||nr||||||||||||||||
set_sentence  price _T_go|||||||||nr||||||||||||||||| down
set_sentence  price _T_plunge|||||||||nr||||||||||||||||
set_sentence  price _T_decline|||||||||nr||||||||||||||||
set_sentence  barrel _T_drop|||||||||nr||||||||||||||||| $
```

```
set_sentence  barrel cent _T_drop|||||||||nr||||||||||||||||

# stagnation
set_text  balance
set_sentence  price _T_stagnate|||||||||post||||||||||||||||
set_sentence  price stagnation
set_sentence  price _T_settle|||||||||post||||||||||||||||
set_sentence  price average
set_sentence  price comparable
set_sentence  price stable
set_sentence  price constant
set_sentence  price fixed
set_sentence  price _T_change|||||||||post|||||||||||||||| _T_#Neg
    |||||||||||||||||||||||||
set_sentence  price _T_alter|||||||||post|||||||||||||||| _T_#Neg
    |||||||||||||||||||||||||
set_sentence  cost equal
set_sentence  cost same
## nr
set_sentence  price _T_stagnate|||||||||nr||||||||||||||||
set_sentence  price _T_settle|||||||||nr||||||||||||||||
set_sentence  price _T_change|||||||||nr|||||||||||||||| _T_#Neg
    |||||||||||||||||||||||||
set_sentence  price _T_alter|||||||||nr|||||||||||||||| _T_#Neg
    |||||||||||||||||||||||||
```