Warszawa, 17.01.2013

**Projekt WEDT**

Kursy akcji na giełdach a informacje.

Dokumentacja końcowa

Adrian Wiśniewski,

*nr indeksu 206298*

Jan Wilczak,

*nr indeksu 214698*

# Temat projektu

Próba wykorzystania informacji pojawiających się na stronach WWW do prognozowania wyników akcji/indeksów giełdowych bądź innych wskaźników finansowych.

# Propozycja implementacji

Zbadamy, czy zachowanie spółki giełdowej można opisać przy użyciu ukrytych modeli Markowa. Zakładamy, że spółka jest pewną skończoną maszyną stanów, której struktury nie znamy – możemy natomiast się jej domyślać na podstawie obserwacji pewnych sygnałów. Sygnałami tymi mogą być:

* Dzienne wahania kursu
* Dwie wartości: wzrost, spadek
* Czasowniki występujące w wiadomościach dotyczących spółki pojawiających się na portalach internetowych
* Zastosowanie NLP
* Indeksu grupy wiadomości
* Po uprzednim pogrupowaniu wiadomości z użyciem metod text-mining’owych.

Każdy ze stanów maszyny ma określone prawdopodobieństwa wzrostu, utrzymania i spadku kursu, co pozwala w praktyczny sposób wykorzystać model. Ponadto analiza ilości stanów i połączeń między nimi może dodatkowo umożliwić planowanie w długim horyzoncie czasu. Rozważamy możliwość budowania modelu dla jednej konkretnej spółki lub wielu podobnych spółek.

# Źródła danych

## Komunikaty

<http://www.money.pl/gielda/spolki_gpw/grupa;lotos;sa,lts,emitent,1.html>

# Narzędzia

* Java - jako język implementacji
* Jahmm - biblioteka do ukrytych modeli Markova
  + <https://code.google.com/p/jahmm/>
* Weka - pakiet dataminingowy użyty do grupowania
  + <http://www.cs.waikato.ac.nz/ml/weka/>
* Morfologik -pakiet NLP
  + <http://morfologik.blogspot.com/>
* JSoup - pakiet do manipulacji dokumentami html
  + <http://jsoup.org/>

# Konfiguracja

Wykorzystanym szkieletem tworzenia aplikacji jest Spring Framework. Dzięki implementacji zasady odwrócenia sterowania („Inversion-of-Control”) pozwala na szybkie i prostsze składanie aplikacji poprzez konfigurację w pliku *app.config.xml*.

<bean class="stockfortuneteller.app.Main">

<property name="tasks">

<list>

<ref bean="stockStemmerCSVBuilder" />

<ref bean="stockWekaARFFBuilder"/>

<!--<ref bean="stockWekaBuilder"/>-->

<ref bean="stockWekaHierarchicalClusterBuilder"/>

<!--<ref bean="stockWekaSimpleKBuilder"/>-->

<!--<ref bean="dataSplitter" />-->

<!--<ref bean="modelBuilder" />-->

</list>

</property>

</bean>

<bean id="stockWekaHierarchicalClusterBuilder" class="stockfortuneteller.data.StockWekaHierarchicalClusterBuilder">

<property name="arffFileName" value="data/info.arff"/>

<property name="csvResultFileName" value="${dataDir}/result"/>

<property name="numberOfClusters" value="60"/>

</bean>

Aplikacja składa się z wielu narzędzi pomocniczych, wykonujących poszczególne etapy przetwarzania. Właściwość <*tasks*> obiektu Main pozwala podać listę narzędzi użytych przy konkretnym uruchomieniu aplikacji. Dzięki temu rozwiązaniu niektóre etapy (takie jak pobranie danych z internetu, czy zbudowanie zbiorów trenującego oraz testowego) można wykonać tylko raz i na wyprodukowanych danych uruchamiać pozostałe narzędzia.

Ponadto plik konfiguracyjny zawiera definicje wszystkich obiektów używanych w programie i pozwala dowolnie modyfikować ich właściwości.

# Zebranie i przetwarzanie danych

1. Pobranie danych

2. Stemming

3. Zapis do CSV

4. Konwersja do ARFF

5. Klasteryzacja wybranym algorytmem

## 1. Pobieranie danych

Dane pochodzą z zewnętrznego serwisu [www.money.pl](http://www.money.pl/). Na stronie każdej spółki giełdowej zaprezentowanej w serwisie można znaleźć komunikaty. Dla odpowiednio skonfigurowanych spółek pobierane są wszystkie tytuły komunikatów z całej historii spółki. Do pobrania danych wykorzystane bibliotekę *JSoup*.

## 2. Stemming

Każdy wyraz w zdaniach składających się na komunikaty zostaje zamieniony na odpowiadający mu stem i pozbawiony polskich znaków. Ponadto z tytułów komunikatów zostają usunięte słowa (nazwy własne) składające się na nazwy analizowanych spółek. Do wykonania tych czynności użyto pakiet *Morfologik*.

## 3. Zapis do pliku csv

Wynikiem przetwarzania stron html i stemmingu jest plik csv, który zawiera 3 kolumny: komunikat, 0 lub 1 (0 dla oznaczenia wzrostu kursu akcji po danym komunikacie, 1 dla spadku), nr spółki.

## 4. Konwersja do pliku arff

Aby dane można było grupować z użyciem biblioteki Weka, muszą one zostać zapisane w formie pliku .arff.

## 5. Klasteryzacja wybranym algorytmem

Klasteryzacja opiera się na podziale wczytanych komunikatów na podstawie ich treści. Następnie pliki wynikowe (oddzielnie dla każdej spółki) tworzone są zgodnie ze wzorcem: unikalny identyfikator, numer klastra, treść komunikatu, 0 lub 1 (0 dla oznaczenia wzrostu kursu akcji po danym komunikacie, 1 dla spadku).

W pakiecie WEKA znajduje się wiele algorytmów klasteryzacji. Przeanalizowane przez nas zostały 3: cobweb, Simple K-Means i Hierarchical Clustering.

Cobweb rosnąco organizuje obserwacje w drzewo decyzyjne. Weka pozwala ustawić parametry algorytmu: acuity, cutoff i seed. Niestety, dla bardzo różnych zestawów parametrów sposób klasteryzacji nie był zadowalający (ok. 90% instancji w tym samym klastrze).

Algorytm K-Means (K-średnich) również nie dał zadowalających rezultatów.

Algorytmem używanym dalej jest Hierarchical Clustering (metoda Czekanowskiego). Dane służące do zbudowania klastra są w nim klasyfikowane zgodnie z przewidywaniami (niektóre komunikaty mogą pojawiać się częściej i wtedy można zaobserwować więcej instancji w danym klastrze, jednak są to różnice na poziomie kilku procent). Pakiet WEKA pozwala na wybór metryki (Euclidean distance), a także sposobu obliczania odległości między klastrami (średnia). Dla naszych danych przyjęliśmy ilość klastrów = 60.

W programie zostały zaimplementowane w/w algorytmy. Poprzez łatwą konfigurację (opisane poniżej) możliwe jest łatwe przełączanie między algorytmami i uruchamianie ich na wcześniej pobranych danych.

# BUDOWA I EWALUACJA MODELU PREDYKCYJNEGO

## Podział danych

**Narzędzie:** dataSplitter

Stworzone serie danych są następnie dzielone na mniejsze serie poprzez skanowanie oknem o długości <*sequencesSize*> (przykładowo seria zdarzeń 12345 jest dzielona na serie 123, 234 i 345). Taki podział pozwala otrzymać dużo danych i odpowiada sposobowi odpytywania systemu, który polega na podaniu sekwencji zaczynającej się w dowolnym momencie czasu. Prawdopodobieństwa początkowe stanów modelu Markowa estymują tutaj rozkład stanów wszystkich spółek w dowolnym czasie.

Kolejnym krokiem jest losowy podział serii na zbiór trenujący i testowy. Wielkość zbioru testowego określa parametr <*testRatio*>. Przed podziałem kolekcja wszystkich sekwencji jest losowo permutowana. Sekwencje są zapisane w katalogach <${*trainDir*}> i <${*testDir*}> w postaci oddzielnych plików tekstowych, co pozwala w łatwy sposób obejrzeć ich zawartość.

## Budowa modelu

**Narzędzie:** modelBuilder

Model predykcji opiera się o ukryty model Markowa, który jest budowany z utworzonego wcześniej zbioru trenującego. Biblioteka jahmm pozwala utworzyć model Markowa przy użyciu algorytmy k-średnich. Nasza aplikacja pozwala dodatkowo określić ciąg posprocessorów modelu, których zadaniem jest dalsza jego obróbka. Dostępne implementacje:

* BwlPostprocessor – używa algorytmu Bauma-Welscha, aby dopasować model do sekwencji trenujących.
* ResetInitialProbabilitiesPostprocessor – resetuje prawdopodobieństwa początkowe modelu. Został wprowadzony w obliczu obawy, że prawdopodobieństwa te będą szkodziły w klasyfikacji sekwencji testowych. Okazał się jednak niepraktyczny, ponieważ oba zbiory posiadają bardzo zbliżony rozkład.

## Predykcja i ewaluacja

**Narzędzie:** tester

Predykcja opiera się o wprowadzenie do modelu sekwencji testowej, a następnie obliczeniu prawdopodobieństwa, że w kolejnych obserwacjach pierwszy wystąpi wzrost albo spadek. Następnie obie wartości są porównywane ze sobą, a decyzja odpowiada większej. Skuteczność predykcji jest określana na cały zbiorze testowym, jako ocena klasyfikacji binarnej. Aplikacja wyświetla macierz pomyłek oraz metryki takie jak: accuracy, precision, recall i specificity. Pozwala ponadto zasymulować ile użytkownik mógłby zyskać inwestując daną kwotę z pomocą naszej aplikacji.

# Wyniki

## Grupowanie k-średnich (k=40)

TRUE POSITIVES - 567.0

TRUE NEGATIVES - 1078.0

FALSE POSITIVES - 30.0

FALSE NEGATIVES - 770.0

ACCURACY - 0.6728016

RECALL - 0.42408377

PRECISION - 0.94974875

SPECIFICITY - 0.90834695

INITIAL CASH - 1000.0

GAINED - 231.90184

LOST - 12.269938

DELTA - 219.6319

NOT LOST (SAVED) - 440.89978

NOT GAINED (MISSED) - 314.92844

TOTAL CASH - 1219.632

BEST CASH - 1546.8303

WORST CASH - 546.8303

BEST GAIN RATIO - 0.40164548

EFFICENCY - 0.6728016

## Grupowanie cobweb

TRUE POSITIVES - 15.0

TRUE NEGATIVES - 960.0

FALSE POSITIVES - 438.0

FALSE NEGATIVES - 496.0

ACCURACY - 0.5107386

RECALL - 0.029354207

PRECISION - 0.03311258

SPECIFICITY - 0.6866953

INITIAL CASH - 1000.0

GAINED - 7.8575172

LOST - 229.4395

DELTA - -221.58199

NOT LOST (SAVED) - 502.8811

NOT GAINED (MISSED) - 259.8219

TOTAL CASH - 778.418

BEST CASH - 1267.6794

WORST CASH - 267.67938

BEST GAIN RATIO - -0.82778865

EFFICENCY - 0.5107387

## Grupowanie hierarchiczne

TRUE POSITIVES - 369.0

TRUE NEGATIVES - 572.0

FALSE POSITIVES - 86.0

FALSE NEGATIVES - 882.0

ACCURACY - 0.49292824

RECALL - 0.29496402

PRECISION - 0.810989

SPECIFICITY - 0.8693009

INITIAL CASH - 1000.0

GAINED - 193.29492

LOST - 45.049763

DELTA - 148.24516

NOT LOST (SAVED) - 299.6333

NOT GAINED (MISSED) - 462.022

TOTAL CASH - 1148.2451

BEST CASH - 1655.3169

WORST CASH - 655.3169

BEST GAIN RATIO - 0.22621904

EFFICENCY - 0.49292824

# Wnioski

Model pozwala uzyskać skuteczność klasyfikacji powyżej 50%. Najlepsza wartość to 67% dla grupowania k-średnich. Należy jednak zwrócić uwagę na to, że większość błędów to błędy typu False Negative, czyli szanse na dobrą inwestycję, z której klasyfikator nie skorzystał. Błędy te są znacznie mniej dotkliwe, niż False Positive, czyli zakup akcji, które stracą. Na szczęście tych drugich błędów jest o rząd wielkości mniej. Widać to w metryce precision, która oznacza, że jeżeli klasyfikator podjął decyzję to na 95% użytkownik nie straci. Z drugiej strony metryka recall oznacza, że klasyfikator korzysta tylko z 42% sytuacji, w których można zyskać. W połączeniu z wysoką precyzją świadczy to o tym, że nie podejmuje ryzykownych decyzji.