STOCK MARKET TREND PREDICT FROM HEADLINES

Nguyen Duc Duy

UNITN – June 2016

Content

1. Introduction

1.1. Stock market

1.2 Stock market indices & trend

1.3 Stock trend predict

1.4 Ideas

1.1 Stock market

- Aggregation of buyers and sellers of stocks (also called shares);
- These may include securities listed on a stock exchange as well as those only traded privately

Fig1. A European Stock Market session

Source: *stockmarket-watch.com*

1.2 Stock indices & trend

- A stock index or stock market index is a measurement of the value of a section of the stock market.
- US market:
  - **Dow Jones Industrial Average**
  - **NASDAQ Composite**

Fig2. Dow Jones & Nasdaq logos

1.3 Stock trend predict

increase ▲ ▼ decrease

WHY?

- Decision making
- **Business Intelligence**

How?

Fig3. What affect the trends?

1.4 Ideas

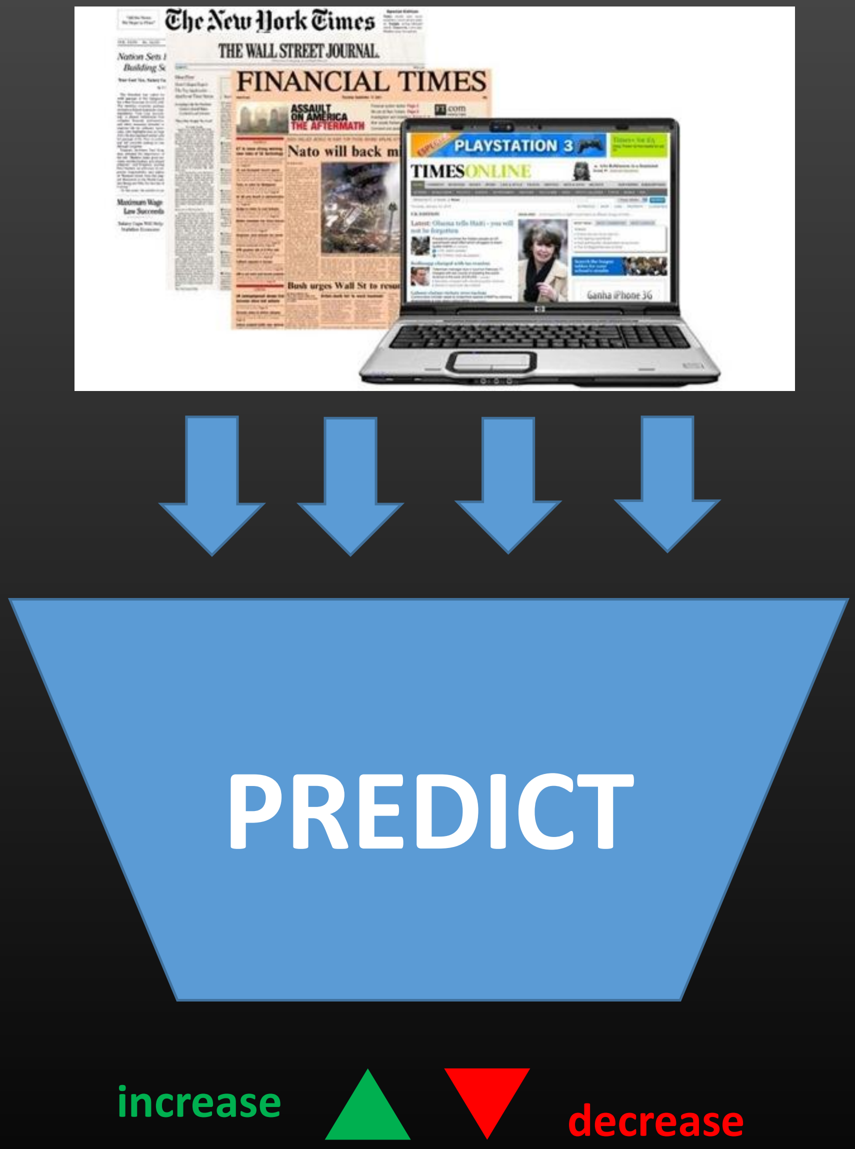


Fig4. Project idea

2. Method

2.1 Dataset

2.2 Feature extraction

2.3 Model build

2.4 Evaluation

2 Method: overview

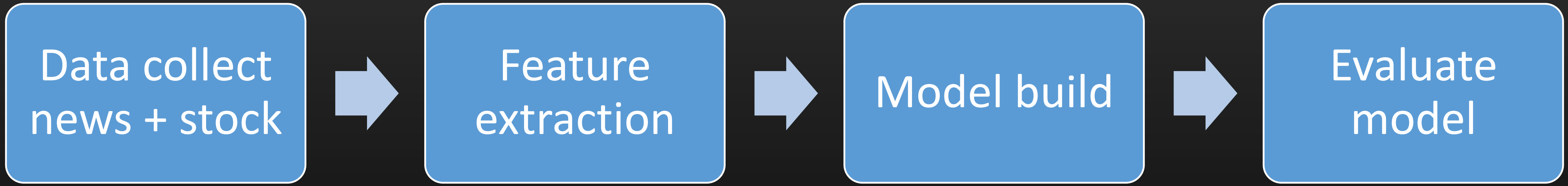


Fig5. Method overview

2.1 Dataset

Headlines

A crawler continuously collected:

- **Corpus size**: **545,323** headlines
- **Time**: *(GMT+7)*
  - From: *2016-04-01 23:02:21.285*
  - To: *2016-06-05 21:21:20.656*
- **Site**:
  - Time (http://time.com)
  - Google News (https://news.google.com/)
  - Wall Street Journal Europe (http://www.wsj.com/europe)
  - Washington Post (https://www.washingtonpost.com/)
  - New York Time (http://www.nytimes.com/)

Stock data

Download from Dukascopy Bank SA

- **Data size: 95,040** records
- **Index:** Dow Jones Industrial Average (USA 30)
- **Data freq**: 1 record/min
- **Time**: *(GMT)*
  - From: 2016-04-01 00:00:00.000
  - To: 2016-06-05 23:59:00.000
- **Site:** https://www.dukascopy.com

2.2 Feature extraction

HEADLINE CORPUS

SEMANTIC SPACE

DS

Time window 1

H1: (X1_0, X1_1,X1_2,X1_3,X1_4...)

H2: (X2_0, X2_1,X2_2,X2_3,X2_4...)

H3: (X3_0, X3_1,X3_2,X3_3,X3_4...)

...

Hn: (Xn_0,Xn_1,Xn_2,Xn_3,Xn_4...)

Time window 2

...

⋮

Time window m

...

HEADLINE VECTORS

TM1: (w1_0, w1_1,w1_2,w1_3,w1_4...)

TM2: (w2_0, w2_1,w2_2,w2_3,w2_4...)

⋮

TMm: (wm_0,wm_1,wm_2,wm_3,wm_4...)

TIME WINDOW VECTORS

STOCK DATA

L1

L2

Lm

(LABELS)

Fig7. Feature Extraction

2.2 Feature extraction: *semantic space*

- From **184,954** news headlines
- Collected continuously from September 16th to November 6 2015.


```
{
    "_id" :
    ObjectId("561e31e27cae6a1af4eed296"),
    "content" : "The Life of a Sheepherder",
    "crawlerid:" : "WTJSpider",
    "time" : ISODate("2015-10-
    14T17:43:46.807Z")
}
```


Fig8. Sample crawled record

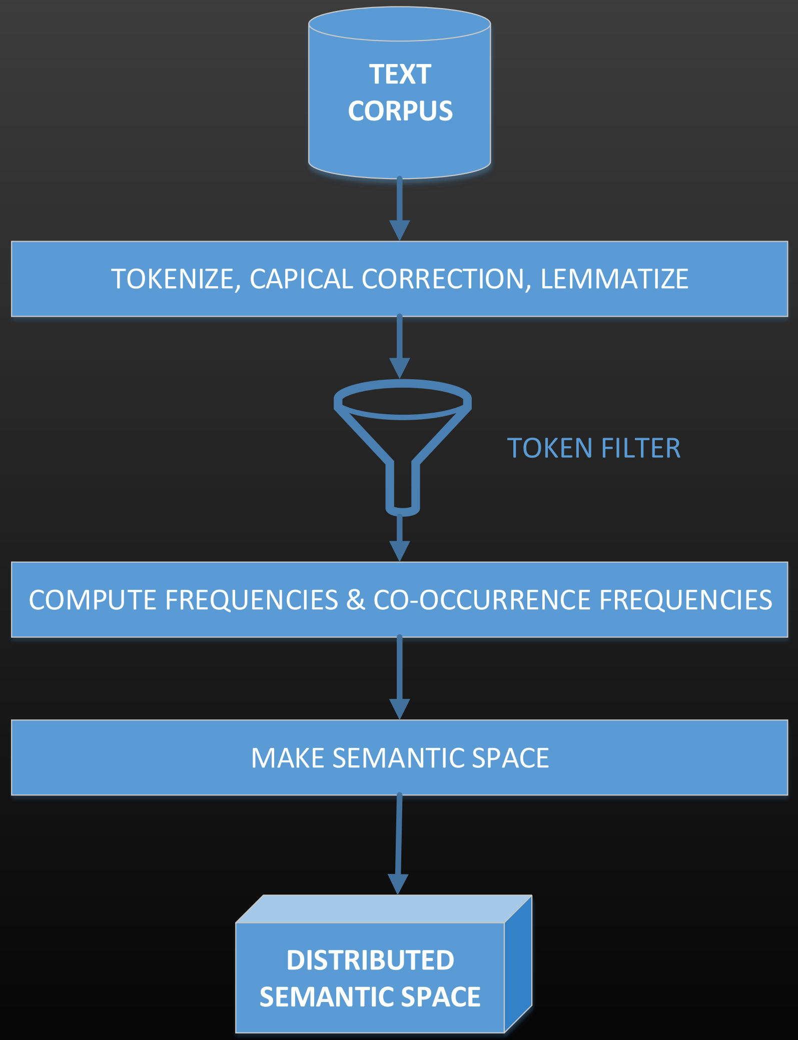


Fig9. Semantic space construction

2.2 Feature extraction: *headline vector*

Multiple **words** document composition

- Word representation:

```
>>> print space.get_neighbours('Trump', 10, CosSimilarity())
[('Trump', 1.0000000000000004), ('Donald', 0.79194684810397509), ('Carson',0.36608895821386095),
('Clinton',0.26288749608776324), ('Iowa',0.26155115776963084), ('Ben', 0.2610271126427458),
('Bush',0.26014531421917397), ('campaign', 0.26010256987586372), ('Fix',0.22267280244652179),
('Jeb', 0.2213852014605992)]
```

Fig10. Neighbors of keyword "Trump"

- Headline representation: addictive composition

```
>>> my_space.get_neighbours('Greece Blasts Shipwrecks Leave Refugees Dead', 5,CosSimilarity())
[('Greece Blasts Shipwrecks Leave Refugees Dead', 1.0000000000000018), ('Refugee Shipwrecks
Greece Leave Dead Days', 0.67752684921750417), ('Refugees Relocated Greece Head Luxembourg',
0.6354818436302625), ('Greece Shipwrecks Kill People Aegean Sea', 0.47656395819802827),
('Protests India Desecration Holy Text Leave two Dead', 0.47384161346021847)]
```

Fig11. Neighbors of headline 'Greece Blasts Shipwrecks Leave Refugees Dead'

2.2 Feature extraction: *time-window vector*

Multiple **documents** group composition

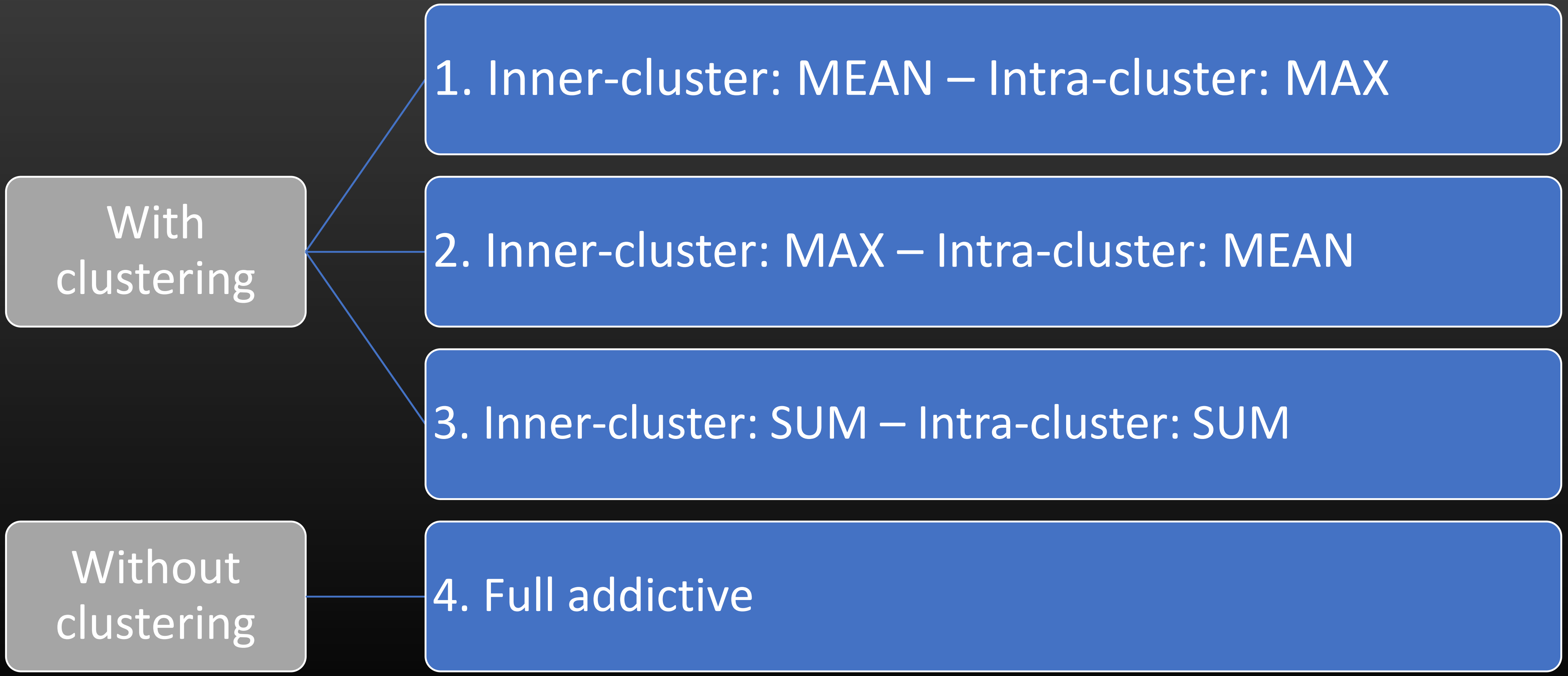
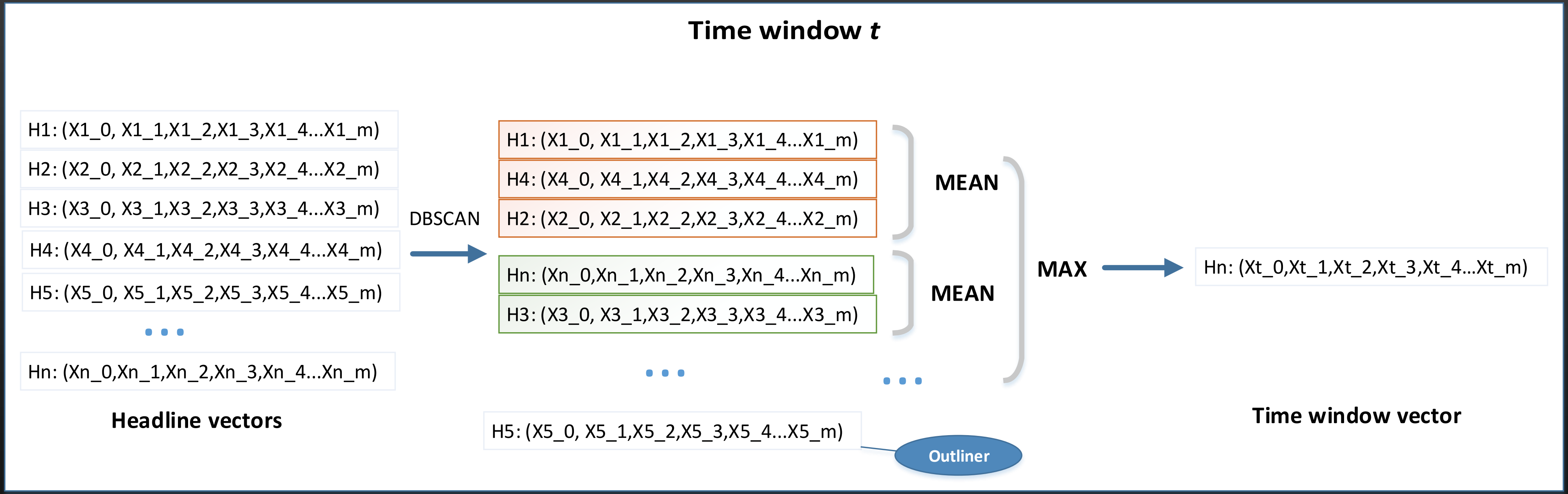


Fig12. Four time-window composition scenarios

2.2 Feature extraction: *time-window vector*

Multiple **documents** group composition

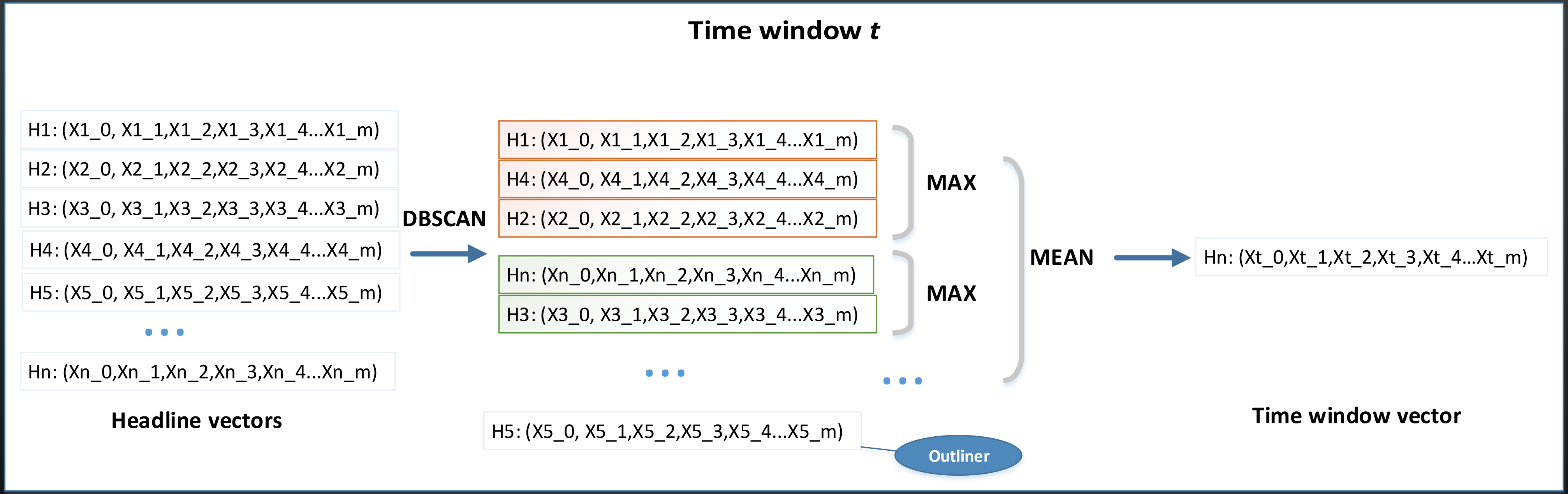


*Cosine distance

Fig12.1. Scen 1. Inner-cluster: MEAN – Intra-cluster: MAX

2.2 Feature extraction: *time-window vector*

Multiple **documents** group composition

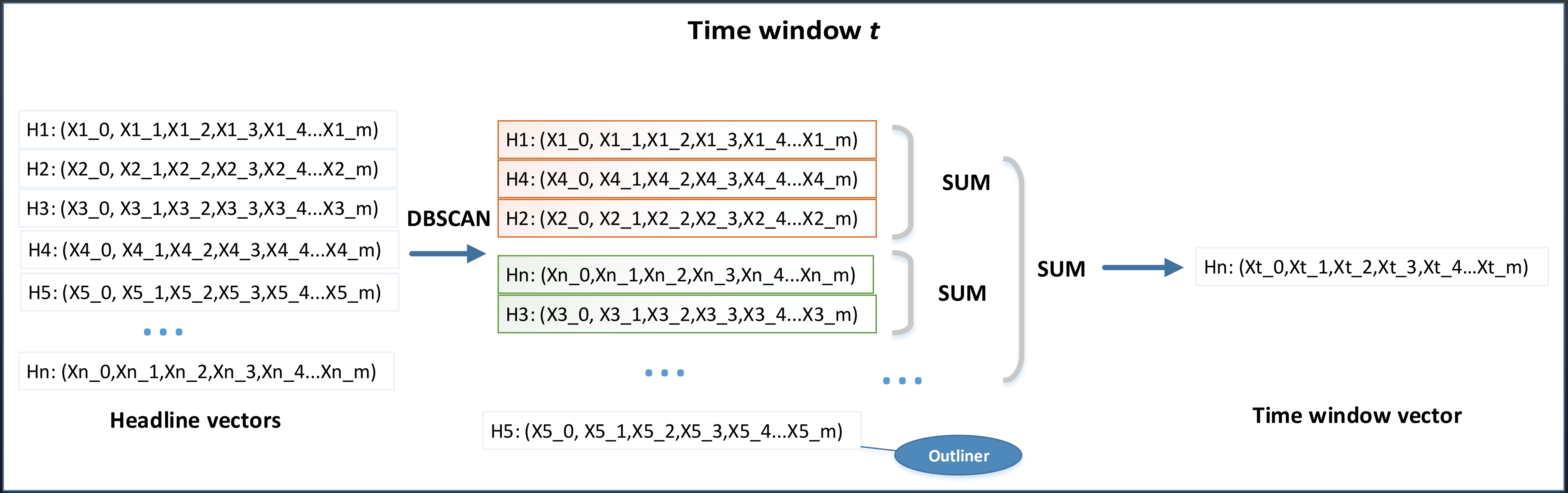


*Cosine distance

Fig12.2. Scen 2. Inner-cluster: MAX – Intra-cluster: MEAN

2.2 Feature extraction: *time-window vector*

Multiple **documents** group composition



*Cosine distance

Fig12.3. Scen 3. Inner-cluster: SUM – Intra-cluster: SUM

2.2 Feature extraction: *time-window vector*

Multiple **documents** group composition

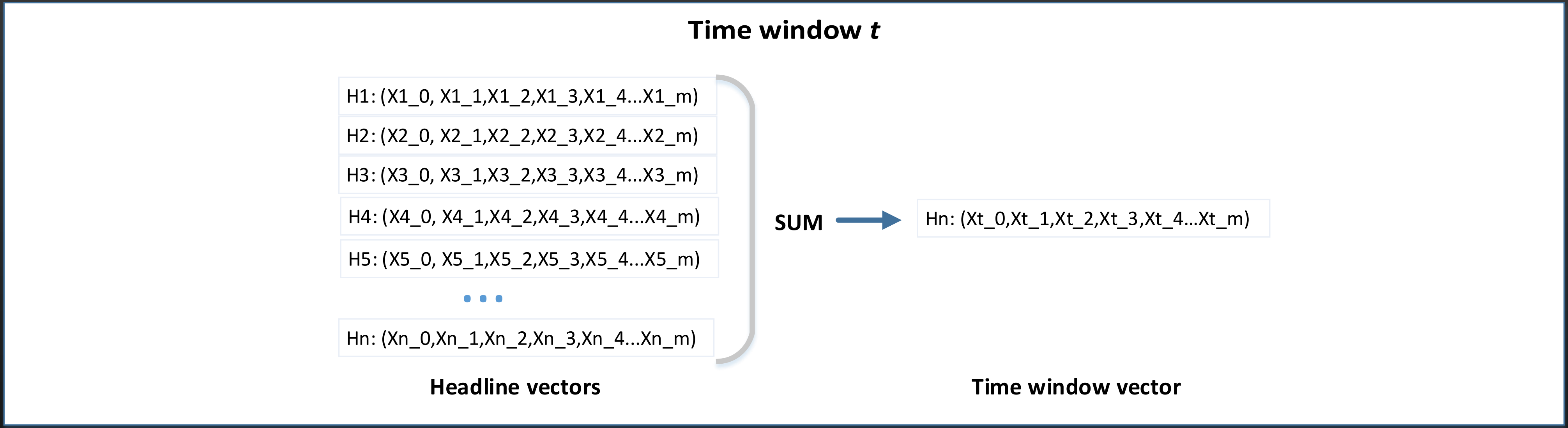


Fig12.4. Scen 4. Full addictive (Full sum)

2.2 Feature extraction: *Stock trend*

- **Definition**: stock trend is the *increase* or *decrease* of DJIA in a give time window
- Time window size: 1hr

```
Data: Stock database, target time window
Result: stock trend for the target time window
Query Open value at the beginning minute of the time window;
Query Close value at the ending minute of the time window;
if Close value equal to Open value then
   Ignore this time window;
else if Close value greater than Open value then
   Return positive label;
else
   Return negative label;
end
```

Alg1. Stock trend extraction

2.3 Model build: SVM classifier

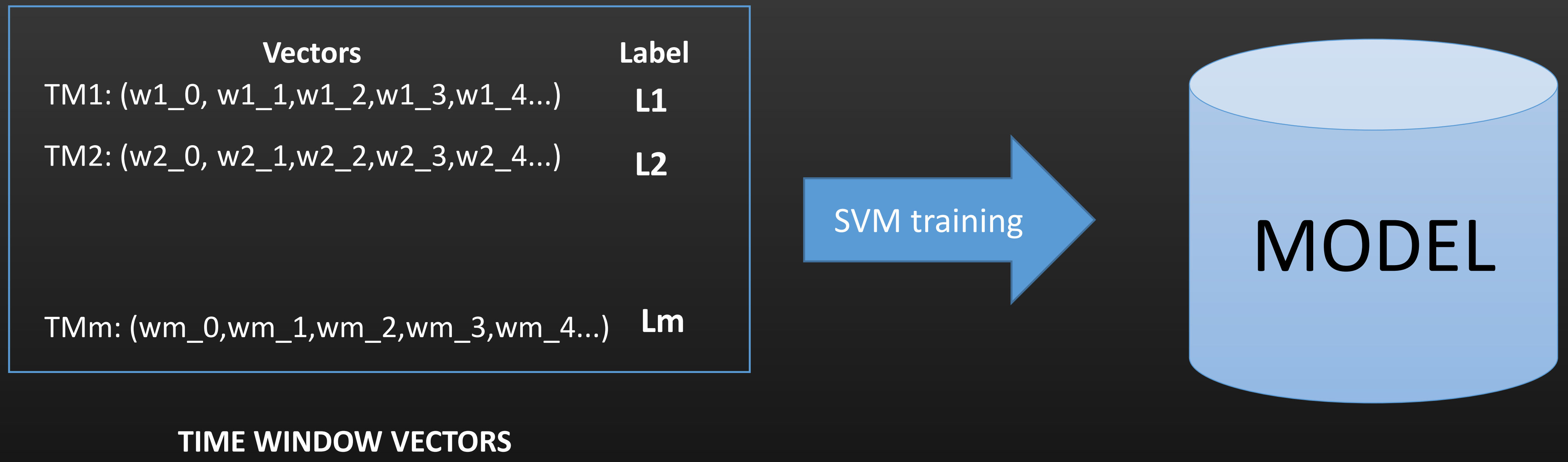


Fig13. Model construction

2.3 Model build: *Data preprocessing*

Scaling	Standardize a dataset along columns.Center to the mean and component wise scale to unit variance.
Normalization	Scale input vectors individually to unit norm (vector length).
Binarization	Boolean thresholding of array-like or scipy.sparse matrix. Feature values below or equal to 0 are replaced by 0, above it by 1.
Softmax by row value*	Apply **softmax** function (**normalized exponential)** with max of each vector
Softmax by the whole matrix value*	Apply **softmax** function (**normalized exponential**) with max the whole matrix
Cosine transformation*	Compute the cosine similarity between the vector to each axis of the semantic space

*Self implemented

Fig13. Model construction

2.3 Model build: *Parameter selection*

- Grid search: (on scaled data)
  - 3 kernels,
  - 6 C values,
  - 8 gamma values

```
>>> svc = GridSearchCV(SVC(), cv=5, param_grid={"kernel": ['poly', 'rbf',
'sigmoid'],"C": [1e-2,1e-1,1e0, 1e1, 1e2, 1e3],"gamma": np.logspace(-2, 2, 8)})

>>> svc.fit(dataset_vectors_mat,dataset_labels_mat)

>>> svc.best_params_{'kernel': 'rbf', 'C': 10.0, 'gamma': 1.9306977288832496}
```

Fig15. Grid search for appropriate parameters
(under 5 folds cross validation, accuracy as comparison metric)

2.4 Evaluation

- 5-folds cross validation, 10 iterations
- Estimators: *mean…*

Accuracy	$= \frac{\Sigma\, True\ positive\ +\ \Sigma\, True\ negative}{\Sigma\, Total\ population}$
Precision	$= \frac{\Sigma\ \text{True positive}}{\Sigma\ \Sigma\ \text{True positive} +\ \Sigma\, False\ \text{positive}}$
Recall	$= \frac{\Sigma\ \text{True positive}}{\Sigma\ \text{True positive} +\ \Sigma\ \text{False negative}}$
F1	$= 2\,\frac{\text{Precision}\ .\, Recall}{\text{Precision} + Recall}$

3. Experimental results

3.1 Experimental results

Accuracy comparison

*(Inner-cluster MEAN – Intra-cluster MAX: *Low performance)*

Composition Scenario	Kernel	Original data	Scaled data	Normalized data	Binarized data	Softmax row	Softmax full	Cosine transform
Max Mean	*poly*	0.5135	0.5016	0.5225	0.5000	0.5015	0.5015	**0.5315**
	rbf	**0.5390**	0.5015	0.5300	0.5150	0.5015	0.5015	**0.5300**
	sigmoid	0.4924	0.5076	0.5015	0.5015	0.5015	0.5015	0.5015
Sum Sum	poly	0.4761	0.4939	0.4385	0.4536	0.5015	0.5015	0.4790
	rbf	0.5015	0.5015	0.4536	0.5015	0.5015	0.5015	0.4536
	sigmoid	0.5015	0.4580	0.5015	0.5015	0.5015	0.5015	0.5015
Full Sum	*poly*	0.4669	0.4925	0.4955	0.4879	0.5015	0.5015	0.4624
	rbf	0.5015	0.5015	0.4834	0.5015	0.5015	0.5015	0.4834
	sigmoid	0.5015	0.4941	0.5015	0.5015	0.5015	0.5015	0.5015

3.1 Experimental results

Score details

2. Inner-cluster MAX – Intra-cluster MEAN

kernel	Scoring method	Original data	Scaled data	Normalized data	Binarized data	Softmax row	Softmax full	Cosine transform
poly	accuracy	0.5135	0.5016	0.5225	0.5000	0.5015	0.5015	**0.5315**
	f1	0.5389	0.6361	0.5347	0.6358	0.0000	0.0000	0.5510
	precision	0.5111	0.4996	0.5170	0.4990	0.0000	0.0000	0.5266
	recall	0.5723	0.8767	0.5601	0.8764	0.0000	0.0000	0.5814
rbf	accuracy	0.5390	0.5015	0.5300	0.5150	0.5015	0.5015	**0.5300**
	f1	0.5585	0.0000	0.5565	0.6218	0.0000	0.0000	0.5565
	precision	0.5340	0.0000	0.5239	0.5085	0.0000	0.0000	0.5239
	recall	0.5874	0.0000	0.5965	0.8011	0.0000	0.0000	0.5965
sigmoid	accuracy	0.4924	0.5076	0.5015	0.5015	0.5015	0.5015	0.5015
	f1	0.4796	0.4926	0.0060	0.0000	0.0000	0.0000	0.0000
	precision	0.4918	0.5038	0.2000	0.0000	0.0000	0.0000	0.0000
	recall	0.4785	0.4851	0.0030	0.0000	0.0000	0.0000	0.0000

3.1 Experimental results

Score details

3. Inner-cluster: SUM – Intra-cluster: SUM

kernel	scoring_method	Original data	Scaled data	Normalized data	Binarized data	Softmax row	Softmax full	Cosine transform
poly	accuracy	0.4761	0.4939	0.4385	0.4536	**0.5015**	**0.5015**	0.4790
	f1	0.4736	0.4124	0.4319	0.4395	0.0000	0.0000	0.4540
	precision	0.4709	0.5948	0.4335	0.4368	0.0000	0.0000	0.4766
	recall	0.4878	0.4717	0.4491	0.4645	0.0000	0.0000	0.4486
rbf	accuracy	**0.5015**	**0.5015**	0.4536	**0.5015**	**0.5015**	**0.5015**	0.4536
	f1	0.0000	0.0000	0.4832	0.0000	0.0000	0.0000	0.4832
	precision	0.0000	0.0000	0.4523	0.0000	0.0000	0.0000	0.4523
	recall	0.0000	0.0000	0.5430	0.0000	0.0000	0.0000	0.5430
sigmoid	accuracy	**0.5015**	0.4580	**0.5015**	**0.5015**	**0.5015**	**0.5015**	**0.5015**
	f1	0.0000	0.4542	0.0000	0.0000	0.0000	0.0000	0.0000
	precision	0.0000	0.4517	0.0000	0.0000	0.0000	0.0000	0.0000
	recall	0.0000	0.4612	0.0000	0.0000	0.0000	0.0000	0.0000

3.1 Experimental results

Score details

4. Full addictive

kernel	Scoring method	Original data	Scaled data	Normalized data	Binarized data	Softmax row	Softmax full	Cosine transform
poly	accuracy	0.4669	0.4925	0.4955	0.4879	**0.5015**	**0.5015**	0.4624
	f1	0.4745	0.4288	0.5208	0.4952	0.0000	0.0000	0.4672
	precision	0.4656	0.3974	0.4987	0.4907	0.0000	0.0000	0.4618
	recall	0.4908	0.5387	0.5546	0.5032	0.0000	0.0000	0.4817
rbf	accuracy	**0.5015**	**0.5015**	0.4834	**0.5015**	**0.5015**	**0.5015**	0.4834
	f1	0.0000	0.0000	0.5229	0.0000	0.0000	0.0000	0.5229
	precision	0.0000	0.0000	0.4840	0.0000	0.0000	0.0000	0.4840
	recall	0.0000	0.0000	0.6036	0.0000	0.0000	0.0000	0.6036
sigmoid	accuracy	**0.5015**	0.4941	**0.5015**	**0.5015**	**0.5015**	**0.5015**	**0.5015**
	f1	0.0000	0.4713	0.0000	0.0000	0.0000	0.0000	0.0000
	precision	0.0000	0.4828	0.0000	0.0000	0.0000	0.0000	0.0000
	recall	0.0000	0.4737	0.0000	0.0000	0.0000	0.0000	0.0000

3.1 Experimental results: comparison

	Method	Accuracy
[1]	• Rule-based system	Dow Jones Indus: 45% FT-SE 100: 46.7%
[2]	• Sentiment analysis from crawled headlines + paragraph • Naïve Bayes Classifier	60%
[3]	• Financial data + news database • BoW, tf-idf, SVM classifier + time series	51%
[4]	• Combines the information from both related news releases and technical indicators to enhance the predictability of the daily stock price trends • Support vector machines	Direct news: 62.5% Indirect news: 50.0% Combined news: 64.7%
[5]	• Twitter's data, S&P 100 index • LDA, DPM Model, time series analysis	47-61% (ts. param. depend)
	My method: News headlines, DS + SVM	53.9%

[1] Wuthrich, B., Cho, V., Leung, S., Permunetilleke, D., Sankaran, K., & Zhang, J. (1998, October). Daily stock market forecast from textual web data. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on* (Vol. 3, pp. 2720-2725). IEEE.

[2] Khare, R., Pathak, N., Gupta, S. K., & Sohi, S. (2004). Stock Broker P–sentiment extraction for the stock market. *WIT Transactions on Information and Communication Technologies*, *33*.

[3] Falinouss, P. (2007). Stock trend prediction using news articles. *Master's thesis, Lulea University of Technology*, 1653-0187.

[4] Zhai, Y., Hsu, A., & Halgamuge, S. K. (2007, June). Combining news and technical indicators in daily stock price trends prediction. In *International Symposium on Neural Networks* (pp. 1087-1096). Springer Berlin Heidelberg.

[5] Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H., & Deng, X. (2013). Exploiting Topic based Twitter Sentiment for Stock Prediction. *ACL (2), 2013*, 24-29.

4. Conclusion & Discussion

4. **Conclusion &** Discussion

Conclusion

- Built a system to predict the stock trend of Dow Jones
- Using Distributional Semantic & Machine learning
- Gain positive result

Discussion

- Multiple documents composition
- Optimization problem
- Time series problem? The trend of time window *t+1* and *t*

Thank you!

Word level

```
>>> print space.get_neighbours('Trump', 10, CosSimilarity())
[('Trump', 1.0000000000000004), ('Donald', 0.79194684810397509), ('Carson', 0.36608895821386095), ('Clinton', 0.26288749608776324), ('Iowa', 0.26155115776963084), ('Ben', 0.26102711264274758), ('Bush', 0.26014531421917397), ('campaign', 0.26010256987586372), ('Fix', 0.22267280244652179), ('Jeb', 0.2213852014605992)]

>>> print space.get_neighbours('white', 10, CosSimilarity())
[('white', 1.0000000000000002), ('Middle-Aged', 0.43434476295700952), ('middle-aged', 0.42538979498306823), ('imagine', 0.39324544217072011), ('excessive', 0.33277541325341797), ('Unclear', 0.33046849974251508), ('girl', 0.28624748790974963), ('conduct', 0.25877685895710345), ('wish-fulfillment', 0.23135498204858113), ('married', 0.19623971970301651)]

>>> print space.get_neighbours('gun', 10, CosSimilarity())
[('gun', 0.99999999999999911), ('responsible', 0.36225357781051504), ('Concealed', 0.3013834162184009), ('depress', 0.28923987568640214), ('control', 0.28590898402929354), ('mensch', 0.27584645652107687), ('holder', 0.27170801825035179), ('McAuliffe', 0.26057588403214715), ('destroy', 0.25531432678352373), ('grab', 0.24451666960944321)]

>>> print space.get_neighbours('language', 10, CosSimilarity())
KeyError

>>> print space.get_neighbours('climate', 10, CosSimilarity())
[('climate', 1.0), ('sperm', 0.38511877454655646), ('sheet', 0.29643865194745761), ('Nations', 0.28639908624548105), ('Caldron', 0.27418766399727196), ('Warming', 0.24587961277971465), ('change', 0.24514464584529705), ('Enceladus', 0.24300109518305371), ('Temperature', 0.23641619876912837), ('Insect', 0.23502173156673303)]


>>> print space.get_neighbours('space', 10, CosSimilarity())

[('space', 0.99999999999999967), ('astronaut', 0.25740411436493293), ('station', 0.24902884863755148), ('carmaker', 0.24508193055967009), ('carve', 0.2360852512993625), ('moon', 0.23526086356409037), ('Saturn', 0.23450790736648702), ('Enceladus', 0.23356266344842602), ('suggest', 0.2313421429377184), ('icy', 0.22584685427169413)]
```

Composition

```
>>> my_space.get_neighbours('Jeb Bush need debate touchdown come close', 5, CosSimilarity())
[('Jeb Bush need debate touchdown come close', 1.0000000000000007), ('Jeb Bush come back', 0.7812252843282168l), ('Jeb Bush RNC restore Telemundo debate', 0.7655739299734090l), ('next GOP debate make break Jeb Bush', 0.7331225921602955 2), ('Jeb Bush much rid tonight debate', 0.7293073741290431 2)]
>>> my_space.get_neighbours('CIT CEO John Thain Retire', 5, CosSimilarity())
[('John Thain Retire CIT CEO', 1.0000000000000004), ('CIT CEO John Thain Retire', 1.0000000000000004), ('CIT Thain Plans Retire March Succeeded Alemany', 0.7840539609287655 6), ('John Legend', 0.5138640976243451 5), ('Ford General Counsel Retire', 0.4919184776733251 9)]


>>> my_space.get_neighbours('Chris Rock talk host 8th Academy Awards', 5, CosSimilarity())
[('Chris Rock talk host 8th Academy Awards', 1.0000000000000011), ('Chris Rock host 016 Oscars', 0.7631929607165657 3), ('Chris Rock last time host Oscars', 0.7112440878828504 3), ('Chris Rock Host Oscars', 0.6491958665306620 8), ('Chris Rock Confirms Hosting Oscars 016', 0.6464226160849291 2)]
 >>> my_space.get_neighbours('Greece Blasts Shipwrecks Leave Refugees Dead', 5, CosSimilarity())
[('Greece Blasts Shipwrecks Leave Refugees Dead', 1.0000000000000018), ('Refugee Shipwrecks Greece Leave Dead Days', 0.6775268492175041 7), ('Refugees Relocated Greece Head Luxembourg', 0.635481843630262 5), ('Greece Shipwrecks Kill People Aegean Sea', 0.4765639581980282 7), ('Protests India Desecration Holy Text Leave two Dead', 0.4738416134602184 7)]
```

Composition

```
>>> my_space.get_neighbours('Amazon CEO Jeff Bezos become richest worldwide', 5, CosSimilarity())
[('Amazon CEO Jeff Bezos become richest worldwide', 1.0), ('become vegetarian', 0.46694138240927968), ('Amazon Removes Listings Apple Chromecast Add Amazon Fire', 0.46208520564853373), ('Evernote Jeff Shotts Startup Heading', 0.45750149518374256), ('final race Talladega Jeff Gordon win pole', 0.4572319440
0999145)]

>>> my_space.get_neighbours('Biden Says oppose Bin Laden Raid', 5, CosSimilarity())
[('Biden Says oppose Bin Laden Raid', 0.99999999999999978), ('Biden Says oppose Raid Killed Bin Laden', 0.96990209536021 51), ('Notebook Joe Biden Bin Laden Raid ambiguity recent history', 0.79755918561244643), ('first Draft Joe Biden Account Stance Bin Laden Raid conflict Others', 0.77997669581457629), ('Osama Bin Laden Raid Obama dministration Secret Legal Deliberations', 0.7313188897740629)]

>>> my_space.get_neighbours('White House Emphasizes Companies commitment cut Emissions', 5, CosSimilarity())
[('White House Emphasizes Companies commitment cut Emissions', 0.99999999999999989), ('House GOP Leaders Strike Budget Deal White House', 0.59117616010823182), ('big money White House Congress DC city hall', 0.57273381296120029), ('Congress White House Near Budget Deal', 0.57016828746619219), ('Congress White House Near Deal Budget', 0.57016828746619219)]
```

2.1 Dataset

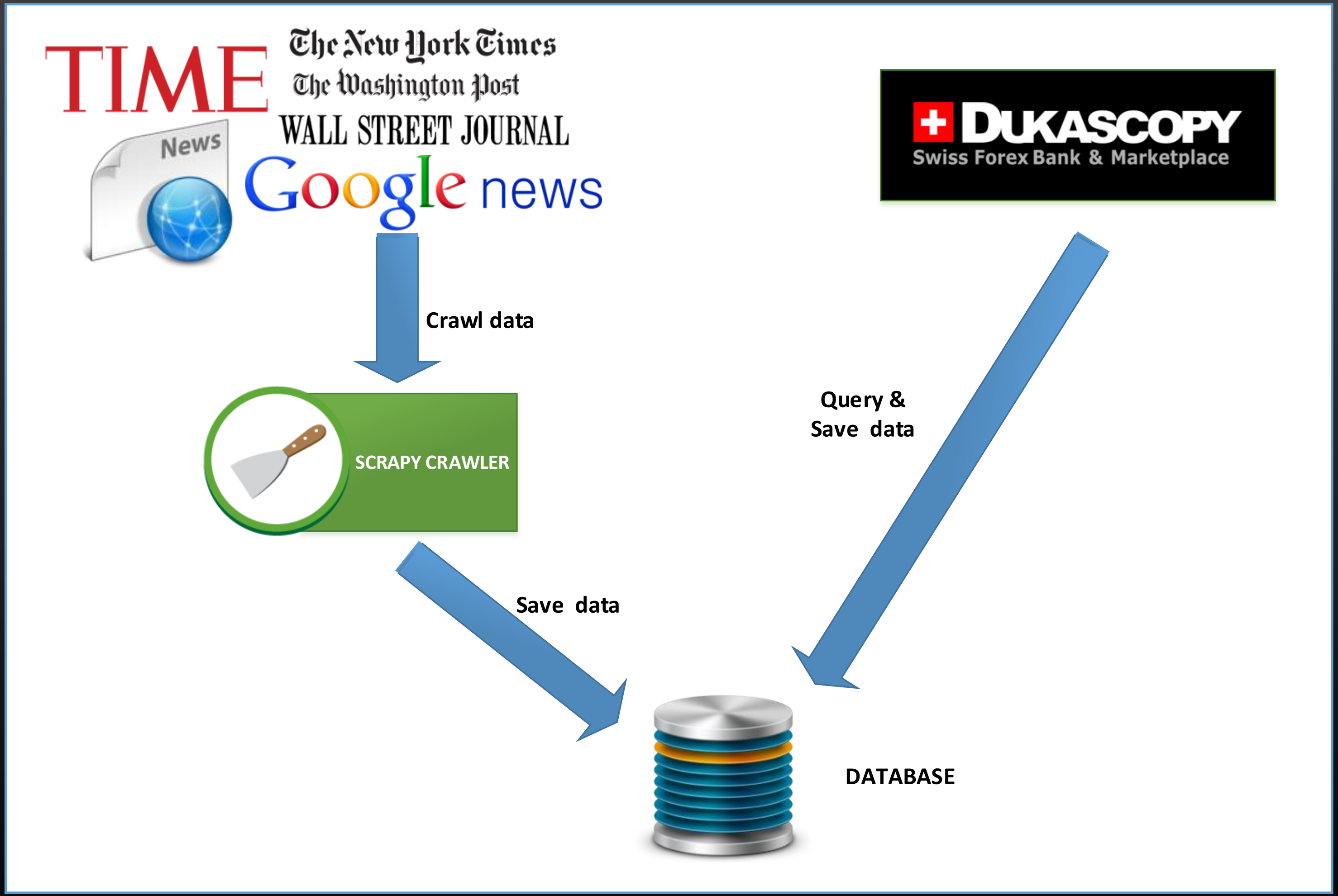
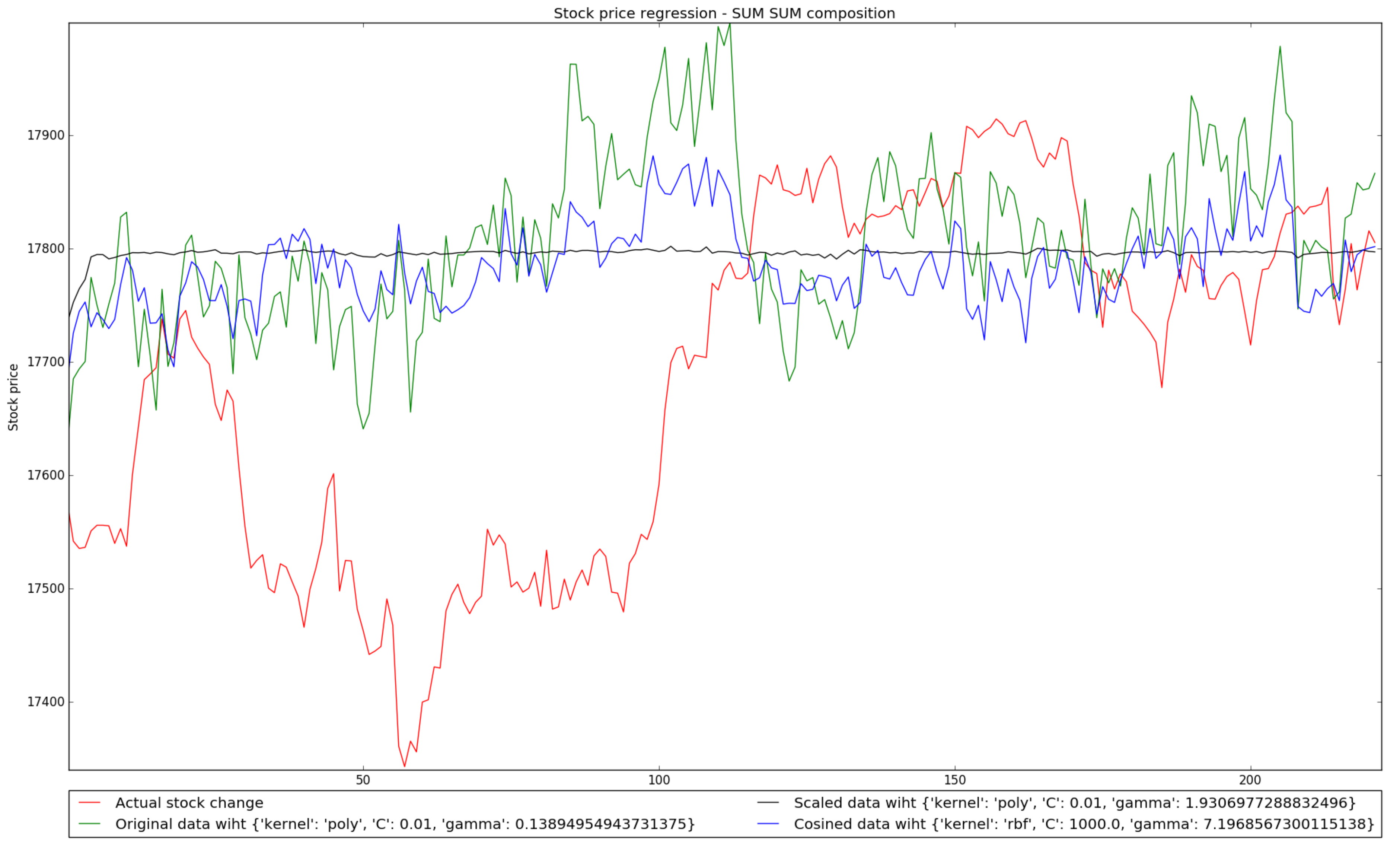


Fig6. data collecting methos

3.1 Experimental results: SumSum regression



3.1 Experimental results: MaxMean regression