Machine Learning for Text

August 2, 2015

1 Intro to Supervised Learning

Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find hidden insights without being explicitly programmed where to look

ML pipeline, setup, feature matrix:
- data given with labels
- but learning setup up to us
- ML algorithm up to us

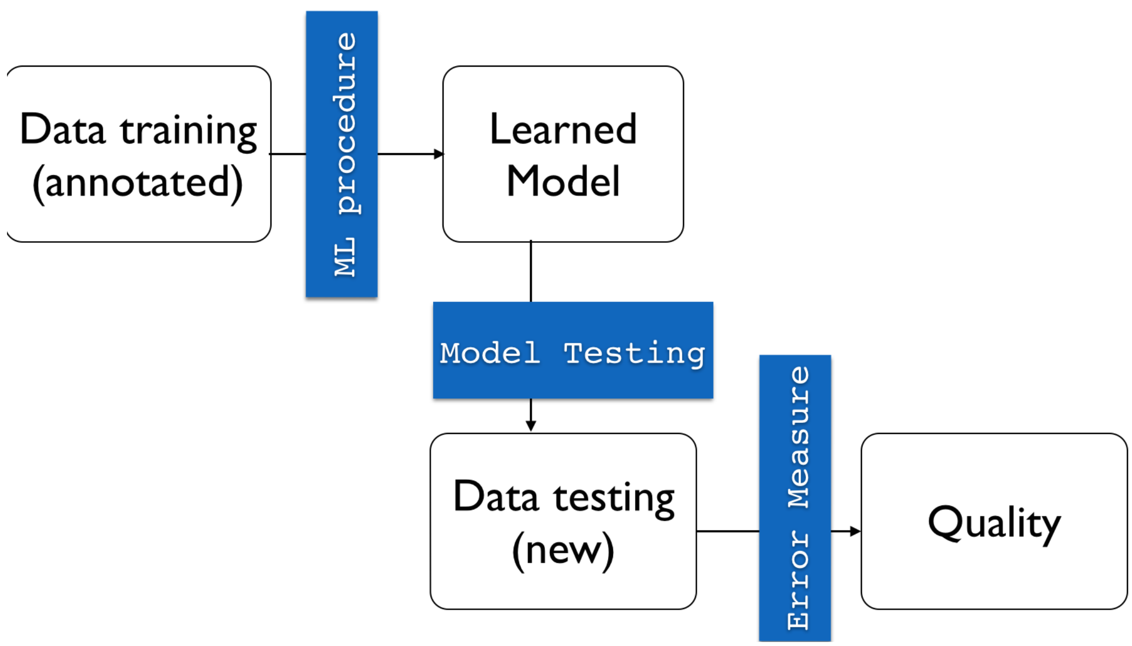


Figure 1: Machine Learning Typical workflow

Data is typically organized for machine learning into a “feature matrix” where rows are datapoints in a vectorial representation, and columns are “features” of data like “age”, “length”, “number of rooms”, “blood pressure", "color", "presence of word `China`" — depending on the actual data. These might be very different for datasets of patients, datasets of text articles, datasets of images etc.

It is important that features are consistent in meaning, i.e. the second column always indicates "age".

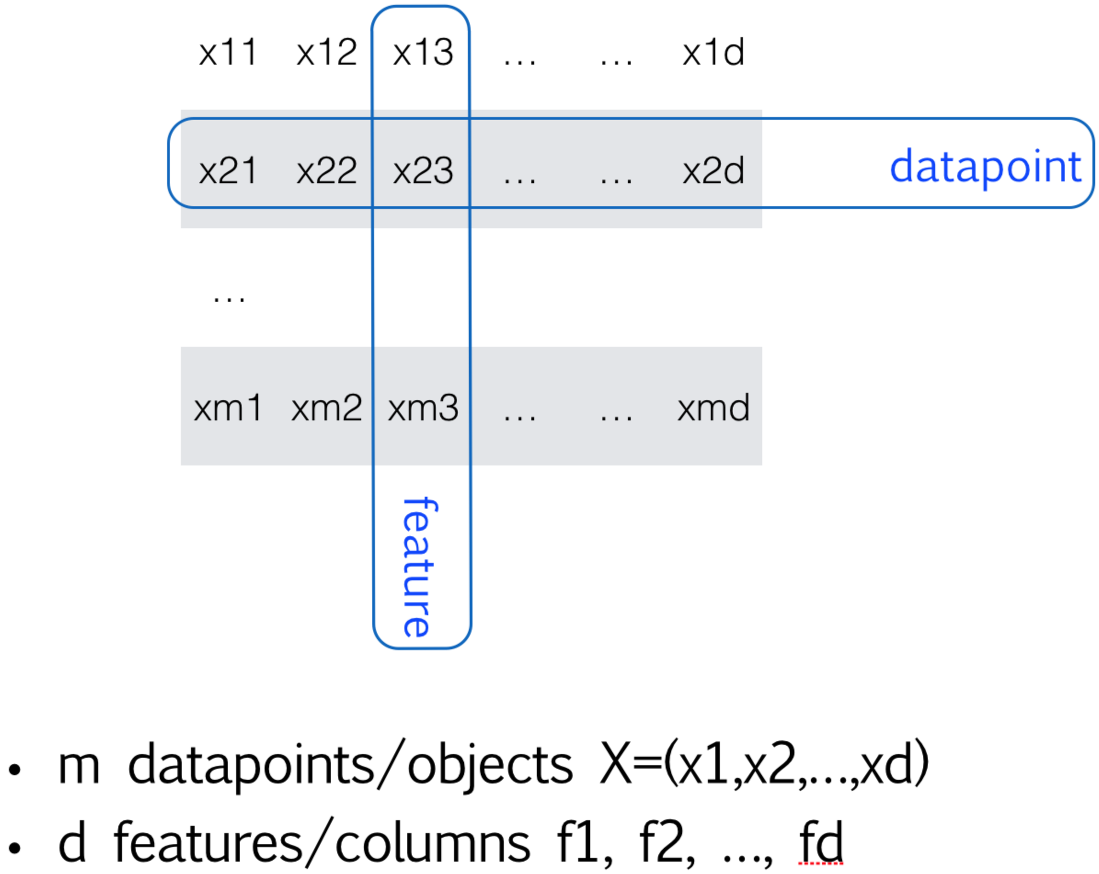
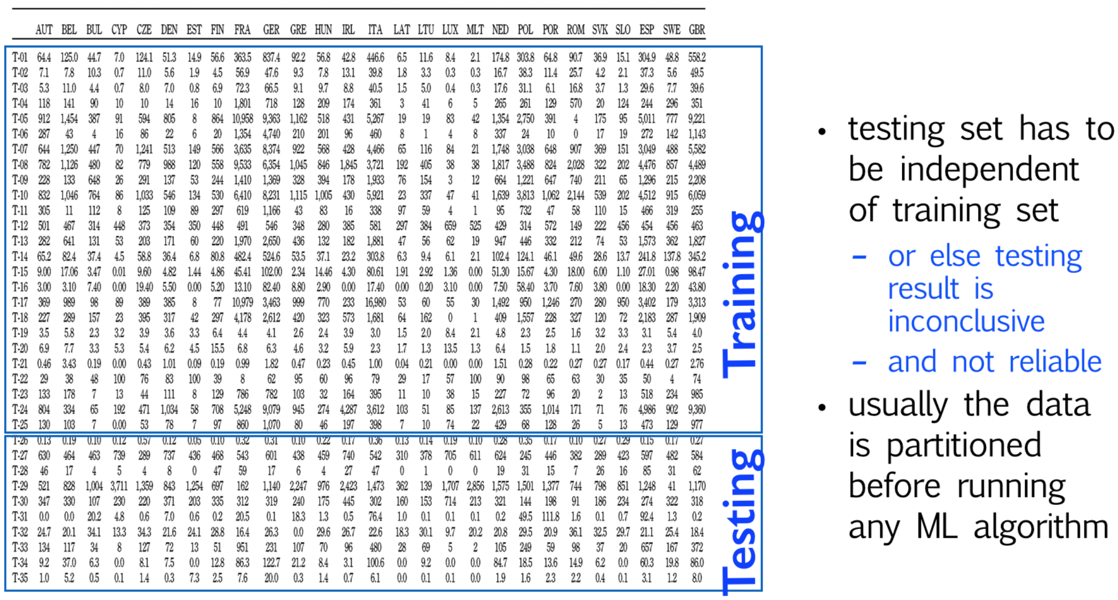


Figure 2: Machine Learning Typical workflow

Heuristic rules. People have used heuristic rules to make decisions since thousands of years ago. These rules are not learned from data, but rather verified over time by trial and error. A machine learning model essentially learns such rules from data, adapting to particular datasets (for example liver cancer patients in Africa might be different than the ones in Europe). Here are some examples of such heuristic rules, not learned from data:

- If fever>100, patient has flu
- If email contains words "free" or "porn", it is spam
- If a web page contains ngram "Michael Jackson", it is relevant to the user
- If age<22 and sex=F and highschool_diploma=Yes, then eligible for application
- If income_per_capita$<$ \$1000, region prone to civil war
- If romantic=Yes and comedy=Yes and Orlando_Bloom=Yes, then movie success among females aged 20-40
- If Nasdaq_Computer_Index=Gain and Apple announces new Ipad, then AAPL_Stock=Buy

For research and testing purposes, usually data is partitioned, randomly, into Training and testing sets. A popular split is Training 90% and testing 10%, or Training 80% and testing 20%.



	AUT	BEL	BUL	CYP	CZE	DEN	EST	FIN	FRA	GER	GRE	HUN	IRL	ITA	LAT	LTU	LUX	MLT	NED	POL	POR	ROM	SVK	SLO	ESP	SWE	GBR
T-01	64.4	125.0	44.7	7.0	124.1	51.3	14.9	56.6	363.5	837.4	92.2	56.8	42.8	446.6	6.5	11.6	8.4	2.1	174.8	303.8	64.8	90.7	36.9	15.1	304.9	48.8	558.2
T-02	7.1	7.8	10.3	0.7	11.0	5.6	1.9	4.5	56.9	47.6	9.3	7.8	13.1	39.8	1.8	3.3	0.3	0.3	16.7	38.3	11.4	25.7	4.2	2.1	37.3	5.6	49.5
T-03	5.3	11.0	4.4	0.7	8.0	7.0	0.8	6.9	72.3	66.5	9.1	9.7	8.8	40.5	1.5	5.0	0.4	0.3	17.6	31.1	6.1	16.8	3.7	1.3	29.6	7.7	39.6
T-04	118	141	90	10	10	14	16	10	1,801	718	128	209	174	361	3	41	6	5	265	261	129	570	20	124	244	296	351
T-05	912	1,454	387	91	594	805	8	864	10,958	9,363	1,162	518	431	5,267	19	19	83	42	1,354	2,750	391	4	175	95	5,011	777	9,221
T-06	287	43	4	16	86	22	6	20	1,354	4,740	210	201	96	460	8	1	4	8	337	24	10	0	17	19	272	142	1,143
T-07	644	1,250	447	70	1,241	513	149	566	3,635	8,374	922	568	428	4,466	65	116	84	21	1,748	3,038	648	907	369	151	3,049	488	5,582
T-08	782	1,126	480	82	779	988	120	558	9,533	6,354	1,045	846	1,845	3,721	192	405	38	38	1,817	3,488	824	2,028	322	202	4,476	857	4,489
T-09	228	133	648	26	291	137	53	244	1,410	1,369	328	394	178	1,933	76	154	3	12	664	1,221	647	740	211	65	1,296	215	2,208
T-10	832	1,046	764	86	1,033	546	134	530	6,410	8,231	1,115	1,005	430	5,921	23	337	47	41	1,639	3,813	1,062	2,144	539	202	4,512	915	6,059
T-11	305	11	112	8	125	109	89	297	619	1,166	43	83	16	338	97	59	4	1	95	732	47	58	110	15	466	319	255
T-12	501	467	314	448	373	354	350	448	491	546	348	280	385	581	297	384	659	525	429	314	572	149	222	456	454	456	463
T-13	282	641	131	53	203	171	60	220	1,970	2,650	436	132	182	1,881	47	56	62	19	947	446	332	212	74	53	1,573	362	1,827
T-14	65.2	82.4	37.4	4.5	58.8	36.4	6.8	80.8	482.4	524.6	53.5	37.1	23.2	303.8	6.3	9.4	6.1	2.1	102.4	124.1	46.1	49.6	28.6	13.7	241.8	137.8	345.2
T-15	9.00	17.06	3.47	0.01	9.60	4.82	1.44	4.86	45.41	102.00	2.34	14.46	4.30	80.61	1.91	2.92	1.36	0.00	51.30	15.67	4.30	18.00	6.00	1.10	27.01	0.98	98.47
T-16	3.00	3.10	7.40	0.00	19.40	5.50	0.00	5.20	13.10	82.40	8.80	2.90	0.00	17.40	0.00	0.20	3.10	0.00	7.50	58.40	3.70	7.60	3.80	0.00	18.30	2.20	43.80
T-17	369	989	98	89	389	385	8	77	10,979	3,463	999	770	233	16,980	53	60	55	30	1,492	950	1,246	270	280	950	3,402	179	3,313
T-18	227	289	157	23	395	317	42	297	4,178	2,612	420	323	573	1,681	64	162	0	1	409	1,557	228	327	120	72	2,183	287	1,909
T-19	3.5	5.8	2.3	3.2	3.9	3.6	3.3	6.4	4.4	4.1	2.6	2.4	3.9	3.0	1.5	2.0	8.4	2.1	4.8	2.3	2.5	1.6	3.2	3.3	3.1	5.4	4.0
T-20	6.9	7.7	3.3	5.3	5.4	6.2	4.5	15.5	6.8	6.3	4.6	3.2	5.9	2.3	1.7	1.3	13.5	1.3	6.4	1.5	1.8	1.1	2.0	2.4	2.3	3.7	2.5
T-21	0.46	3.43	0.19	0.00	0.43	1.01	0.09	0.19	0.99	1.82	0.47	0.23	0.45	1.00	0.04	0.21	0.00	0.00	1.51	0.28	0.22	0.27	0.27	0.17	0.44	0.27	2.76
T-22	29	38	48	100	76	83	100	39	8	62	95	60	96	79	29	17	57	100	90	98	65	63	30	35	50	4	74
T-23	133	178	7	13	44	111	8	129	786	782	103	32	164	395	11	10	38	15	227	72	96	20	2	13	518	234	985
T-24	804	334	65	192	471	1,034	58	708	5,248	9,079	945	274	4,287	3,612	103	51	85	137	2,613	355	1,014	171	71	76	4,986	902	9,360
T-25	130	103	7	0.00	53	78	7	97	860	1,070	80	46	197	398	7	10	74	22	429	68	128	26	5	13	473	129	977
T-26	0.13	0.19	0.10	0.12	0.57	0.12	0.05	0.10	0.32	0.31	0.10	0.22	0.17	0.36	0.13	0.14	0.19	0.10	0.28	0.35	0.17	0.10	0.27	0.29	0.15	0.17	0.27
T-27	630	464	463	739	289	737	436	468	543	601	438	459	740	542	310	378	705	611	624	245	446	382	289	423	597	482	584
T-28	46	17	4	5	4	8	0	47	59	17	6	4	27	47	0	1	0	0	19	31	15	7	26	16	85	31	62
T-29	521	828	1,004	3,711	1,359	843	1,254	697	162	1,140	2,247	976	2,423	1,473	362	139	1,707	2,856	1,575	1,501	1,377	744	798	851	1,248	41	1,170
T-30	347	330	107	230	220	371	203	335	312	319	240	175	445	302	160	153	714	213	321	144	198	91	186	234	274	322	318
T-31	0.0	0.0	20.2	4.8	0.6	7.0	0.6	0.2	20.5	0.1	18.3	1.3	0.5	76.4	1.0	0.1	0.1	0.1	0.2	49.5	111.8	1.6	0.1	0.7	92.4	1.3	0.2
T-32	24.7	20.1	34.1	13.3	34.3	21.6	24.1	28.8	16.4	26.3	0.0	29.6	26.7	22.6	18.3	30.1	9.7	20.2	20.8	29.5	20.9	36.1	32.5	29.7	21.1	25.4	18.4
T-33	134	117	34	8	127	72	13	51	951	231	107	70	96	480	28	69	5	2	105	249	59	98	37	20	657	167	372
T-34	9.2	37.0	6.3	0.0	8.1	7.5	0.0	12.8	86.3	122.7	21.2	8.4	3.1	100.6	0.0	9.2	0.0	0.0	84.7	18.5	13.6	14.9	6.2	0.0	60.3	19.8	86.0
T-35	1.0	5.2	0.5	0.1	1.4	0.3	7.3	2.5	7.6	20.0	0.3	1.4	0.7	6.1	0.0	0.1	0.1	0.0	1.9	1.6	2.3	2.2	0.4	0.1	3.1	1.2	8.0

Figure 3: Machine Learning Typical workflow

To avoid various biases that can appear due to randomness from such partitions, a more robust way to test ML algorithms is to use Cross-Validation:
- split data randomly into K folds, for example K=10 folds each with 10% of data
- train on K-1 folds and test on the one fold not included in training
- repeat train/test for each fold, K separate runs
- average measurements

1.1 Popular Machine Learning libraries

- **LibLinear**
- **LibSVM**
- **SVM light, SVM rank**
- **WEKA**
- **Mahout**
- **Python scikit-learn**
- **ntlk**
- **spark**
- **Mallet**

2 setup for ranking documents

First, documents dont come with many natural features like "age", "bloodtest" etc, so a necessary step is fature extraction. Usually the meaning of text is what matters most in building a good model, so it is necessary to extract features such as words (unigrams), bigrams, trigrams, other expressions etc. The feature values can be simply binary (presence/absence of term) or counts (TF values).

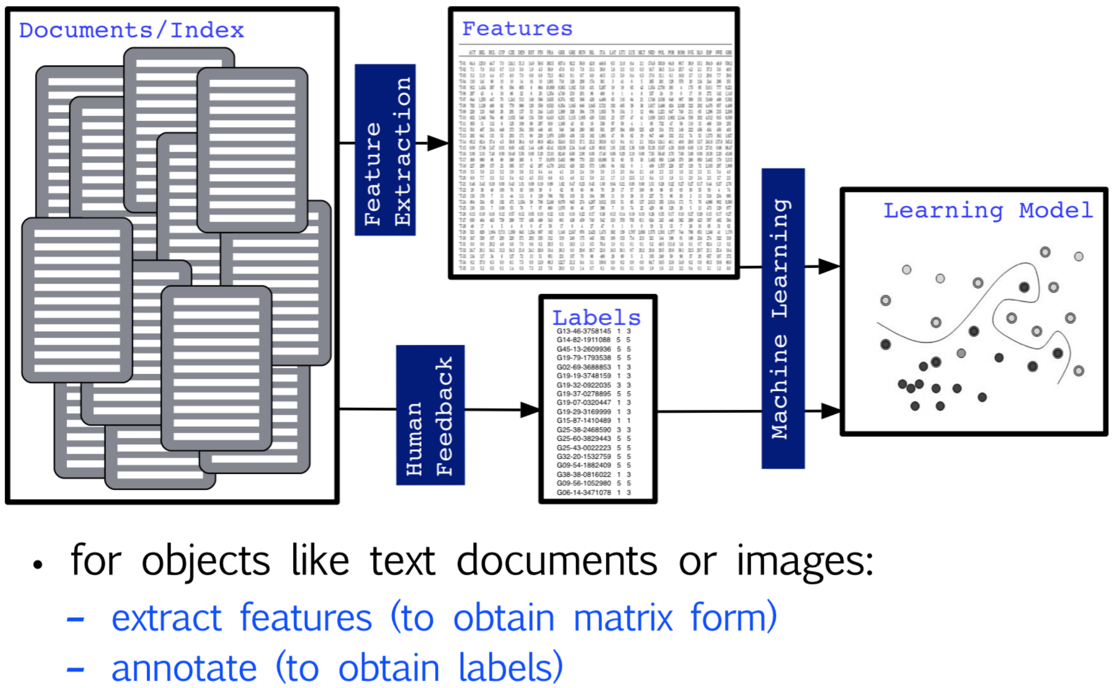


Figure 4: Machine Learning Typical workflow

In Information Retrieval, when data is given as (query,document) datapoints, such as the TREC documents with queryIDs and QREL assessments, we want to create training and testing sets following queries. In other words, we typically dont want to have the same query on both training and testing sets; instead each query will send all its documents to either training set, or all its documents to the testing set.

nominal output. If the ML algorithm is nominal (like usual) with a prediction per datapoint, then we *sort the output for each test query* and measure typical IR performance such as Average Precision or nDCG.

pairwise output. If the output of the learning algorithm is pairwise, then we get an output for pairs (doc_1, doc_2) indicating which document is better. For each test query, will have to use an aggregation algorithm (similar to metasearch fusion, of graph sorting) to obtain a ranking from all pairs. Once we have a ranking, we can compute IR measures.

3 ML algorithms for supervised learning

- **Decision Trees and Regression Trees.** See separate notes.
- **Linear and Logistic Regression.** See separate notes.
- **pairwise ML algorithms.** RankBoost, LambdaMart, RankNet

4 ML with LibLinear package, evaluation, cross validation

4.1 running LibLinear

Liblinear

LibLinear is a public Machine Learning package for linear classifier for data with millions of instances and features. You could download the package from `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

Once you have the unzipped folder of LibLinear, you need first compile the files. To do this, just run "make" in your terminal.

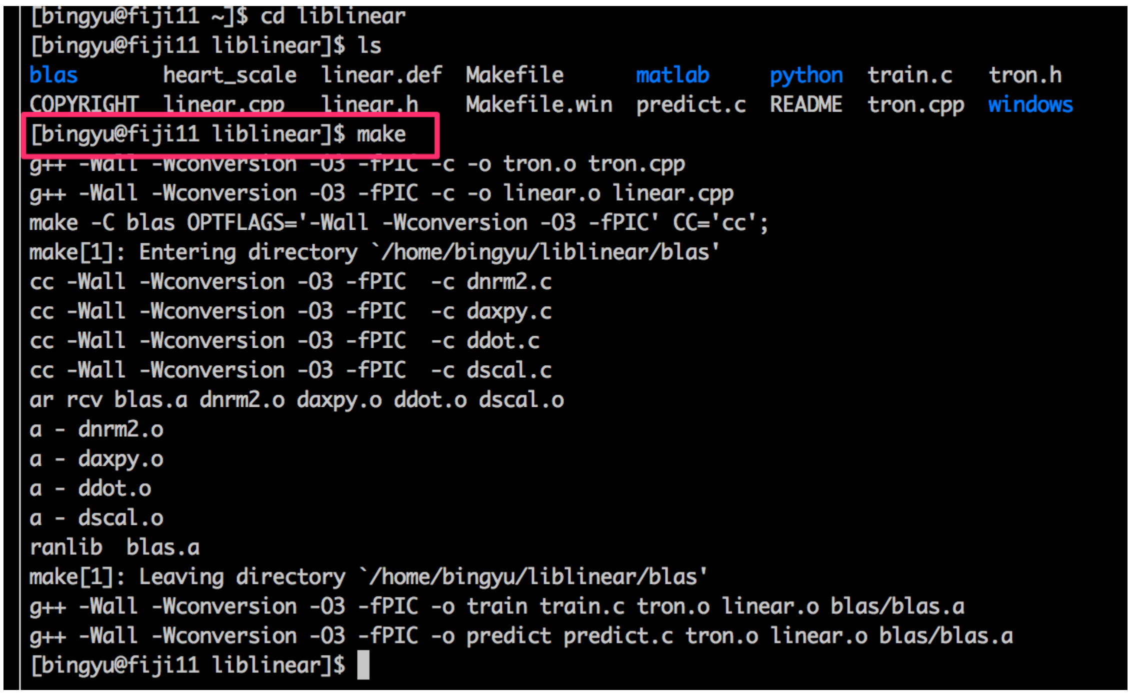


Figure 5: Compile the LibLinear Package

After compiling, you could do training and prediction process. Suppose your input feature matrix named "train.txt" and "test.txt". To do the simple training and prediction:

training

Run "./train train.txt linear.model" in your terminal.

"train.txt" is your input training feature matrix file. "linear.model" is the output model name, you could name it.

prediction

Run "./predict test.txt linear.model linear.predict".

"test.txt" is your input testing file. "linear.model", this is the model you got from training process as an input here. "linear.predict" is the output prediction results for testing data. You could name it.

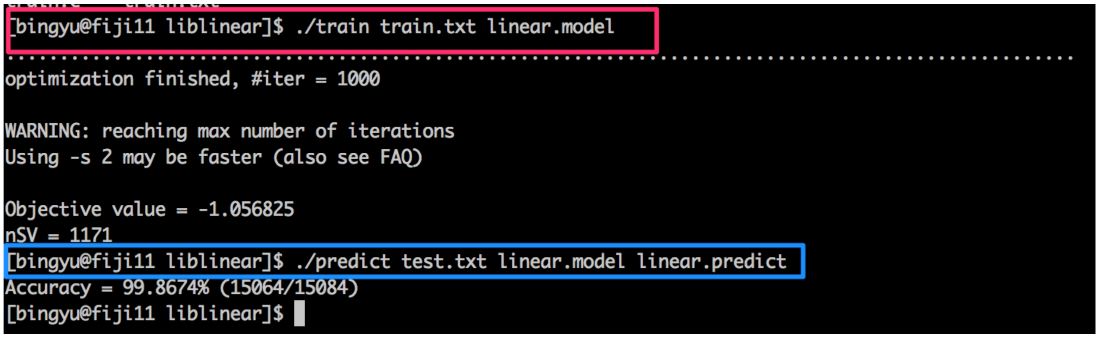


Figure 6: Training and Prediction by LibLinear Example

4.2 ML evaluation

To evaluate your Machine Learning algorithms, first you need select a training set and testing set. There is no overlapping between these two groups, which means you could not include any testing samples in your training, or any training samples in your testing. Random selecting from data into training and testing could be a simple way.

After getting training and testing, the Machine Learning algorithms will be trained on training set and will be evaluated on the testing set. Remember that the model training process is not allowed to access any testing set.

Once you have the trained model, you could do prediction on testing set using the trained model. To evaluate the predictions on the testing set, based on different Machine Learning tasks, there are different performance measures. For classification, we could choose accuracy (total number of correct predictions on testing divided by the total number of testing samples.) Furthermore, to dig more information from accuracy, you could refer confusion matrix(`https://en.wikipedia.org/wiki/Confusion_matrix`) information. For example, in spam classification, people may care more about the false positives. For regression problem, you may consider the Root Mean Squared Error as the measure(`https://en.wikipedia.org/wiki/Mean_squared_error`).

4.3 Cross Validation

A more sophisticated way than evaluate ML only using one training and testing set is trying to randomly split the entire dataset into a few folders evenly, let's say 5 folders. Then you could do following evaluations:

- Train on folder 1,2,3,4; Test on 5
- Train on folder 2,3,4,5; Test on 1
- Train on folder 3,4,5,1; Test on 2
- Train on folder 4,5,1,2; Test on 3
- Train on folder 5,1,2,3; Test on 4

	AUT	BEL	BUL	CYP	CZE	DEN	EST	FIN	FRA	GER	GRE	HUN	IRL	ITA	LAT	LTU	LUX	MLT	NED	POL	POR	ROM	SVK	SLO	ESP	SWE	GBR	
T-01	64.4	125.0	44.7	7.0	124.1	51.3	14.9	56.6	363.5	837.4	92.2	56.8	42.8	446.6	6.5	11.6	8.4	2.1	174.8	303.8	64.8	90.7	36.9	15.1	304.9	48.8	558.2	Fold 1
T-02	7.1	7.8	10.3	0.7	11.0	5.6	1.9	4.5	56.9	47.6	9.3	7.8	13.1	39.8	1.8	3.3	0.3	0.3	16.7	38.3	11.4	25.7	4.2	2.1	37.3	5.6	49.5	
T-03	5.3	11.0	4.4	0.7	8.0	7.0	0.8	6.9	72.3	66.5	9.1	9.7	8.8	40.5	1.5	5.0	0.4	0.3	17.6	31.1	6.1	16.8	3.7	1.3	29.6	7.7	39.6	
T-04	118	141	90	10	10	14	16	10	1,801	718	128	209	174	361	3	41	6	5	265	261	129	570	20	124	244	296	351	
T-05	912	1,454	387	91	594	805	8	864	10,958	9,363	1,162	518	431	5,267	19	19	83	42	1,354	2,750	391	4	175	95	5,011	777	9,221	Fold 2
T-06	287	43	4	16	86	22	6	20	1,354	4,740	210	201	96	460	8	1	4	8	337	24	10	0	17	19	272	142	1,143	
T-07	644	1,250	447	70	1,241	513	149	566	3,635	8,374	922	568	428	4,466	65	116	84	21	1,748	3,038	648	907	369	151	3,049	488	5,582	
T-08	782	1,126	480	82	779	988	120	558	9,533	6,354	1,045	846	1,845	3,721	192	405	38	38	1,817	3,488	824	2,028	322	202	4,476	857	4,489	
T-09	228	133	648	26	291	137	53	244	1,410	1,369	328	394	178	1,933	76	154	3	12	664	1,221	647	740	211	65	1,296	215	2,208	Fold 3
T-10	832	1,046	764	86	1,033	546	134	530	6,410	8,231	1,115	1,005	430	5,921	23	337	47	41	1,639	3,813	1,062	2,144	539	202	4,512	915	6,059	
T-11	305	11	112	8	125	109	89	297	619	1,166	43	83	16	338	97	59	4	1	95	732	47	58	110	15	466	319	255	
T-12	501	467	314	448	373	354	350	448	491	546	348	280	385	581	297	384	659	525	429	314	572	149	222	456	454	456	463	
T-13	282	641	131	53	203	171	60	220	1,970	2,650	436	132	182	1,881	47	56	62	19	947	446	332	212	74	53	1,573	362	1,827	Fold 4
T-14	65.2	82.4	37.4	4.5	58.8	36.4	6.8	80.8	482.4	524.6	53.5	37.1	23.2	303.8	6.3	9.4	6.1	2.1	102.4	124.1	46.1	49.6	28.6	13.7	241.8	137.8	345.2	
T-15	9.00	17.06	3.47	0.01	9.60	4.82	1.44	4.86	45.41	102.00	2.34	14.46	4.30	80.61	1.91	2.92	1.36	0.00	51.30	15.67	4.30	18.00	6.00	1.10	27.01	0.98	98.47	
T-16	3.00	3.10	7.40	0.00	19.40	5.50	0.00	5.20	13.10	82.40	8.80	2.90	0.00	17.40	0.00	0.20	3.10	0.00	7.50	58.40	3.70	7.60	3.80	0.00	18.30	2.20	43.80	
T-17	369	989	98	89	389	385	8	77	10,979	3,463	999	770	233	16,980	53	60	55	30	1,492	950	1,246	270	280	950	3,402	179	3,313	
T-18	227	289	157	23	395	317	42	297	4,178	2,612	420	323	573	1,681	64	162	0	1	409	1,557	228	327	120	72	2,183	287	1,909	
T-19	3.5	5.8	2.3	3.2	3.9	3.6	3.3	6.4	4.4	4.1	2.6	2.4	3.9	3.0	1.5	2.0	8.4	2.1	4.8	2.3	2.5	1.6	3.2	3.3	3.1	5.4	4.0	
T-20	6.9	7.7	3.3	5.3	5.4	6.2	4.5	15.5	6.8	6.3	4.6	3.2	5.9	2.3	1.7	1.3	13.5	1.3	6.4	1.5	1.8	1.1	2.0	2.4	2.3	3.7	2.5	
T-21	0.46	3.43	0.19	0.00	0.43	1.01	0.09	0.19	0.99	1.82	0.47	0.23	0.45	1.00	0.04	0.21	0.00	0.00	1.51	0.28	0.22	0.27	0.27	0.17	0.44	0.27	2.76	
T-22	29	38	48	100	76	83	100	39	8	62	95	60	96	79	29	17	57	100	90	98	65	63	30	35	50	4	74	
T-23	133	178	7	13	44	111	8	129	786	782	103	32	164	395	11	10	38	15	227	72	96	20	2	13	518	234	985	
T-24	804	334	65	192	471	1,034	58	708	5,248	9,079	945	274	4,287	3,612	103	51	85	137	2,613	355	1,014	171	71	76	4,986	902	9,360	
T-25	130	103	7	0.00	53	78	7	97	860	1,070	80	46	197	398	7	10	74	22	429	68	128	26	5	13	473	129	977	Fold K-1
T-26	0.13	0.19	0.10	0.12	0.57	0.12	0.05	0.10	0.32	0.31	0.10	0.22	0.17	0.36	0.13	0.14	0.19	0.10	0.28	0.35	0.17	0.10	0.27	0.29	0.15	0.17	0.27	
T-27	630	464	463	739	289	737	436	468	543	601	438	459	740	542	310	378	705	611	624	245	446	382	289	423	597	482	584	
T-28	46	17	4	5	4	8	0	47	59	17	6	4	27	47	0	1	0	0	19	31	15	7	26	16	85	31	62	
T-29	521	828	1,004	3,711	1,359	843	1,254	697	162	1,140	2,247	976	2,423	1,473	362	139	1,707	2,856	1,575	1,501	1,377	744	798	851	1,248	41	1,170	Fold K
T-30	347	330	107	230	220	371	203	335	312	319	240	175	445	302	160	153	714	213	321	144	198	91	186	234	274	322	318	
T-31	0.0	0.0	20.2	4.8	0.6	7.0	0.6	0.2	20.5	0.1	18.3	1.3	0.5	76.4	1.0	0.1	0.1	0.1	0.2	49.5	111.8	1.6	0.1	0.7	92.4	1.3	0.2	
T-32	24.7	20.1	34.1	13.3	34.3	21.6	24.1	28.8	16.4	26.3	0.0	29.6	26.7	22.6	18.3	30.1	9.7	20.2	20.8	29.5	20.9	36.1	32.5	29.7	21.1	25.4	18.4	
T-33	134	117	34	8	127	72	13	51	951	231	107	70	96	480	28	69	5	2	105	249	59	98	37	20	657	167	372	
T-34	9.2	37.0	6.3	0.0	8.1	7.5	0.0	12.8	86.3	122.7	21.2	8.4	3.1	100.6	0.0	9.2	0.0	0.0	84.7	18.5	13.6	14.9	6.2	0.0	60.3	19.8	86.0	
T-35	1.0	5.2	0.5	0.1	1.4	0.3	7.3	2.5	7.6	20.0	0.3	1.4	0.7	6.1	0.0	0.1	0.1	0.0	1.9	1.6	2.3	2.2	0.4	0.1	3.1	1.2	8.0	

- split data in K folds
- execute K independent learning trials:
  - train on K-1 folds
  - test on remaining fold
  - measure testing performance
- average results across K trials

Figure 7: Machine Learning Typical workflow

Finally you could take an average of your five times performance measures on testing set as your final evaluation. We call this method as Cross Validation.

The number of folders is chosen based on the size of your dataset. We usually choose 5 or 10.

5 ML for text, IR data

5.1 Document Features

We could regard documents or text as sequences of words so far, but documents have much richer structure and information. Let us see some other extra features we can use as clues of relevance.

Structural Features

Some of these features are structural: the document's organization gives clues about the topic:

- The title, headings, and menu give fine-grained topic and subtopic information.
- Links and their anchor text provide clues about the relevance of other pages related to this one.

http://en.wikipedia.org/wiki/Susan_Dumais

Figure 8: Structural Features Example

Topical Features

Other features are topical: the document's text may contain special words and phrases that pertain to a certain topic.

- Named entities(people, companies, places, events ...) are strong topical clues.
- Topic modeling discovers the vocabulary that tends to be used when speaking of a certain topic.

Figure 9: Topical Features Example

Features from ML

Tools from Machine Learning can be used to generate additional features for a page.

- Document classifiers are used to identify news articles, blogs, reviews, and other types of specialized pages.
- Document clustering can find very similar pages, which is useful for providing diverse result lists and "more like this" functions (e.g. clustering news articles by story).

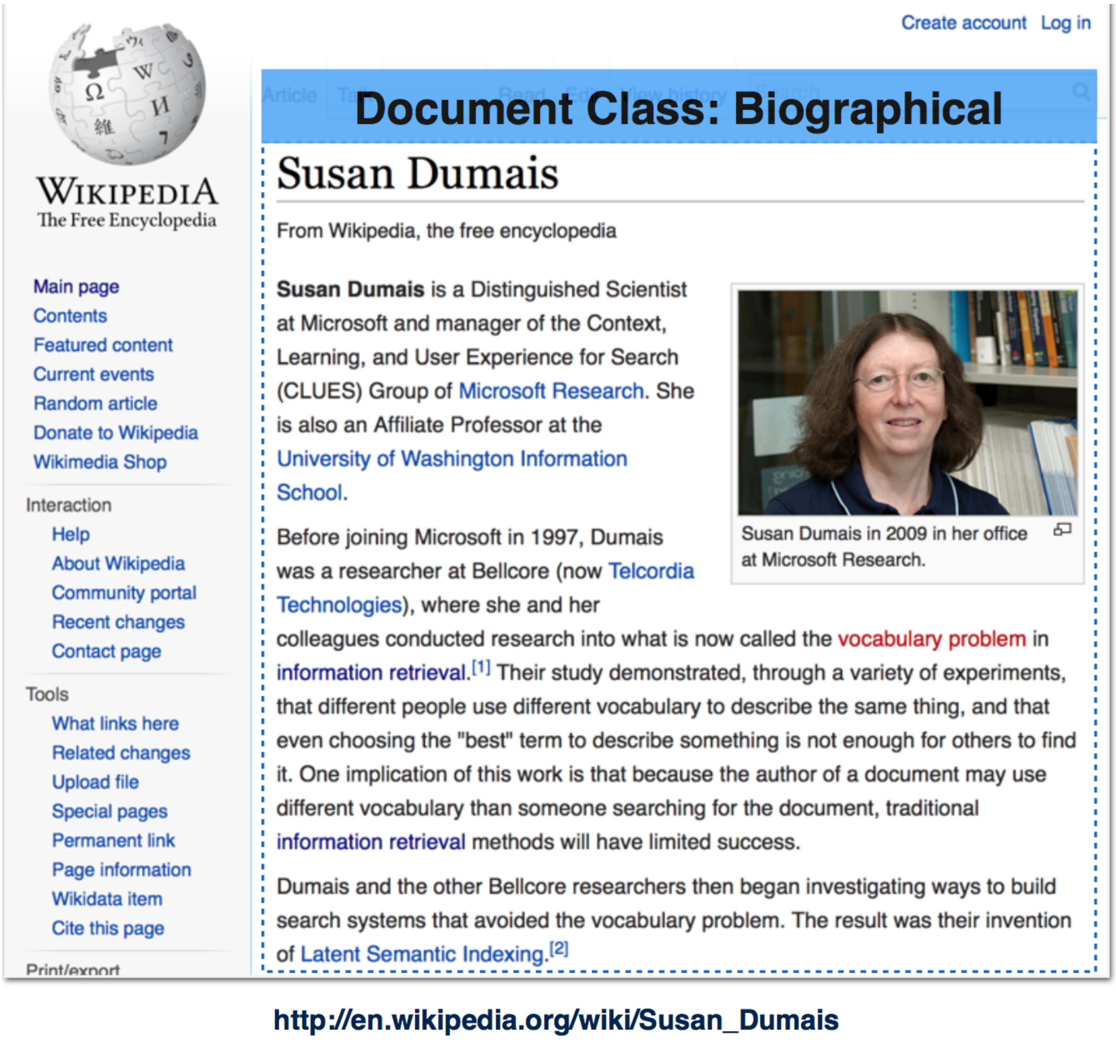


Figure 10: Features from ML Example

5.2 Feature Matrix

When we have collected all the document features we are interested in, we can use standard Machine Learning classifiers to learn how to predict document relevance from document features. This makes scoring functions such as BM25 simply one component of a more complicated relevance predictor. And all documents and queries can be converted into numeric vectors that provide this rich information to learning algorithms. This allows us to leverage the best ML techniques for document ranking. A example is shown as below:

DocID	Body BM25	Title BM25	In-links	...	Rel?
1	1.23	1.2	3		1
2	1.2	1.4	12		1
3	17.3	13.2	2		0
4	10.55	0	207		0

Figure 11: Features Matrix Example

6 ES feature value collection, sparse Feature Matrix

See Cheng's and Bingyu's demos.

This is a Demo on IMDB dataset, where documents are movie reviews. Labels are annotations "good" or "bad" for each review obtained from ratings. The purpose of a model is to predict the "good" or "bad" from the review text. It is essentially the same problem as predicting "spam" or "not spam" from the email text.

————————————train ————————-

(pos) I highly recommend this movie.
(neg) I do not recommend this movie to anybody.
(neg) It is a waste of time.
(pos) Good fun stuff !
(neg) It's just not worth your time.

———————————————————

————————————- test ————————

(neg) I do not recommend this movie unless you are prepared for the biggest waste of money and time of your life.
(neg) This movie was the slowest and most boring so called horror that I have ever seen.
(neg) The film is not worth watching.
(pos) A wonderful film
(pos) This is a really nice and sweet movie that the entire family can enjoy.

————————————————————-

Two steps: 1. gather ngrams, 2. computing matching scores

6.1 enumerate unigrams, bigrams

Gather ngrams only once from the training set. Use these ngrams to compute matching scores for both training set and test set. Make sure the same ngrams are used and the orders are the same. Enumerating ngrams: Scan all documents. For each document, pull out the term vector. Get sorted list. Scan the list.

Procedure:
connected to index
there are 10 documents in the index.
number of training documents = 5
there are 2 classes in the training set.
label distribution in training set:
neg:3, pos:2,
LabelTranslator{intToExt={0=neg, 1=pos}, extToInt={neg=0, pos=1}}
fields to be considered = [body]
gathering 1-grams from field body with slop 0 and minDf 0
gathered 22 1-grams
gathering 2-grams from field body with slop 0 and minDf 0
gathered 21 2-grams
there are 43 ngrams in total
creating training set
allocating 43 columns for training set
training set created
data set saved to `/huge1/people/chengli/projects/pyramid/archives/exp35/imdb_toy/1/train`
creating test set
allocating 43 columns for test set
test set created
data set saved to `/huge1/people/chengli/projects/pyramid/archives/exp35/imdb_toy/1/test`

6.2 Sparse Feature Matrix

The format that we want: an on-disk sparse matrix In each line, the first number is the label. The rests are feature index: feature value pairs. The feature index starts at 0. Since the feature matrix is very sparse, only non-zero feature values are stored. We expect features not listed to have value 0.

Fundamental constraint: cannot hold the entire dense matrix in memory

sparse matrix options:

1. use a sparse matrix library

python: numpy sparse matrix

`http://docs.scipy.org/doc/scipy/reference/sparse.html`

java: Mahout sparse matrix or Guava table

`http://mahout.apache.org/`

`http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/collect/Table.html`

WARNING: Be careful with complexity of the operations

2. write your own data structure array of hash maps

6.3 skip-grams, slop, and "span near query"

Computing matching scores:

For unigrams, one can use binary feature values (present or not), tf, tfidf, or the scores provided by Elasticsearch. For ngrams or skip-grams, one can use binary feature values (present or not), phrase frequency, or the scores provided by Elasticsearch. Please refer to the "Span Near" query [1] documentation for details. A 0 slot corresponds to ngrams, while a non-zero slop corresponds to skip-grams.

6.4 Running Cheng's Learning Algorithms

System requirement: Java 8.

Each data set is a folder, which includes two mandatory files "feature_matrix.txt", "config.txt" and some optional files.

The "config.txt" file looks like:

numClasses=2

numDataPoints=5

missingValue=false

numFeatures=43

Before running the code, please first modify the paths in run.sh and exp33.config. Then in command line type:

./run.sh exp33.config

[1] `https://www.elastic.co/guide/en/elasticsearch/reference/1.6/query-dsl-span-near-query.html`