Prediction in Sentiment Analysis - Amazon Sample

Results - Insights - Methodology

Philippe Lambot

April 14, 2020

****

PREDICTION IN SENTIMENT ANALYSIS - AMAZON SAMPLE

****

I. EXECUTIVE SUMMARY

88 % prediction accuracy has been reached on the validation set, against 50 % with a baseline model. Data is an Amazon sample provided in UCI Machine Learning Repository.

In this sentiment analysis project, which factors have contributed towards that improvement with 38 percentage points?

Natural Language Processing has contributed 21.7 percentage points: corpus, lowercasing, punctuation handling, stopword

removal, stemming, tokenization from sentences into words, bag of words.

Text mining has brought additional accuracy improvement with 12.7 percentage points. The following insights have been determinant.

In decision trees predominate some tokens conveying subjective information; but other tokens containing subjective information have not been used in false negatives and false positives. Such ignored subjective information has been retrieved from random samples of false negatives and false positives, exclusively on the training set; customized lists have been established with tokens sorted as having either positive or negative sentiment orientation; occurrences of these tokens in reviews have been replaced with either a positive or a negative generic token. Polarization and text substitution have brought 10.3 percentage points out of the 12.7.

Another insight has been about negation impact: negation has been fruitfully integrated, contributing 2.4 percentage points towards the 12.7 improvement from text mining.

Machine learning optimization has been performed across 10 models. Testing has been conducted on accuracy distributions across bootstrapped resamples. eXtreme Gradient Boosting has emerged as the most performing model in this project and has boosted accuracy with 3.6 additional percentage points.

TAGS: sentiment analysis, natural language processing, text mining, subjective information, tokenization, bag of words, word frequency, wordclouds, decision trees, false negatives, false positives, text classification, polarization, lists of positive n-grams, lists of negative n-grams, text substitution, machine learning, binary classification, eXtreme Gradient Boosting, Monotone Multi-Layer Perceptron Neural Network, Random Forest, Stochastic Gradient Boosting, Support Vector Machines with Radial Basis Function Kernel, AdaBoost Classification Trees, bootstrapping, accuracy distributions across resamples, R

GITHUB: https://github.com/Dev-P-L/Sentiment-Analysis (https://github.com/Dev-P-L/Sentiment-Analysis)

II. FOREWORD to READERS

Dear Readers, you are most welcome to run the project on your own computer if you so wish.

This project is lodged with the GitHub repository https://github.com/Dev-P-L/Sentiment-Analysis (https://github.com/Dev-P-L/Sentiment-Analysis).

It is comprised of twelve files. All code is included in SA_Amazon_Code.Rmd. It does not show in the result report, called SA_Amazon_Insights&Results.html.

For your convenience, the dataset has already been downloaded onto the GitHub repository wherefrom it will be automatically retrieved by the code from SA_Amazon_Code.Rmd. If you so wish, you can also easily retrieve the dataset from https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences (https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences) and adapt the SA_Amazon_Code.Rmd code accordingly.

You can knit SA_Amazon_Code.Rmd (please in HTML) and produce SA_Amazon_Insights&Results.html on your own computer. On my laptop, running SA_Amazon_Code.Rmd takes approximately four hours. For information, here are some characteristics of my work environment:

- R version 3.5.1 (2018-07-02) – “Feather Spray”,
- RStudio Version 1.1.456,
- Windows 10.

Some packages are required in SA_Amazon_Code.Rmd. The code from SA_Amazon_Code.Rmd contains instructions to download these packages if they are not available yet.

Now, let’s get in touch with data.

III. GETTING IN TOUCH with DATA

As explained on the UCI Machine Learning Repository website, data is organized in a CSV file in two columns. In the first column, there are 1,000 Amazon product reviews (sentences). In the second column, there is a positive or negative evaluation; the ratio of positive evaluations is 50 %.

That file will be split into training reviews - two thirds of reviews - and validation reviews. Let’s have a quick look at a few training reviews.

	ILLUSTRATIVE SAMPLE OF TRAINING REVIEWS	SENTIMENT
61	**Essentially you can forget Microsoft’s tech support.**	**Neg**
187	**Design Flaw?.**	**Neg**

	ILLUSTRATIVE SAMPLE OF TRAINING REVIEWS	SENTIMENT
188	Great phone!.	Pos
218	I also didn't like the "on" button, it felt like it would crack with use.	Neg
247	Excellent wallet type phone case.	Pos
344	New Battery works great in phone.	Pos
523	I ended up sliding it on the edge of my pants or back pockets instead.	Pos
640	Disappointing accessory from a good manufacturer.	Neg
675	Then I had to continue pairing it periodically since it somehow kept dropping.	Neg
874	My sister has one also and she loves it.	Pos
893	Excellent product for the price.	Pos
970	I'm trying to return it for a refund.	Neg

Let's proceed to some NLP.

IV. NATURAL LANGUAGE PROCESSING

A. Corpus - Tokenization - Bag of Words

Training reviews will be transposed into a corpus. Then the corpus will be processed in NLP: words will be lowercased, punctuation marks will be removed as well as stopwords and finally words will be stemmed.

Tokenization will then take place, a bag of words being created. Applying a sparsity threshold will only leave tokens that appear in at least 0.5 % of reviews. The bag of words takes the form of a Document Term Matrix: the 668 rows correspond to the 668 training reviews; there is a column for each token selected across the sparsity process. At the junction of each row and each column, there is a frequency number representing the occurrence of the corresponding token in the corresponding review.

As a pre-attentive insight, a wordcloud will show the most frequent tokens.

There are topic-related tokens such as "phone", tokens conveying subjective information such as "great", etc. Before analyzing token categories, let's check up the technical adequacy of results from the NLP process.

B. Checking up NLP Output

Some tokens were not expected, such as “dont” or “ive”, since they seem to originate in short forms and were expected to have been eliminated as stopwords. Let’s start investigating with “dont”. How many rows contain " dont “?

	OCCURRENCE OF “dont”
Bag of Words from Training Reviews	**17**

Let’s have a look at the first review that has produced “dont”.

	FIRST REVIEW GENERATING “dont”
Set of Training Reviews	**Dont waste your money…**

“dont” contains a spelling error or is “alternative” grammar. Nevertheless, ideally, it should be treated as a short form from standard grammar, i.e. as the short form “don’t”. Consequently, if we want to eradicate short forms as being stopwords, we’ll have to complement stopwords with variants such as “dont”, “couldnt”. This will be done in the next section “C. Fine Tuning NLP”.

Let’s see the second review that, after NLP, has generated “dont”.

	SECOND REVIEW GENERATING “dont”
Set of Training Reviews	**That’s a huge design flaw (unless I’m not using it correctly, which I don’t think is the case).**

This is another scenario: “don’t” has been written in a standard way, but all punctuation marks have been removed, consequently it has become “dont” and is no longer identical to the stopword “don’t” and, very logically, it has not been removed.

This will be corrected not just for this instance but because it is systemic (you only have to change the system and not to correct manually) and systematic (it could happen several or even many times).

Consequently, short forms should be removed before removing punctuation. Or punctuation marks should be removed except for apostrophes before removing short forms. An appropriate solution will be applied in the next section “C. Fine Tuning NLP”.

While examining other tokens coming from training reviews, other oddities have been discovered. There are several unigrams that seem to originate from two words, e.g. “brokeni” at row 24.

24	**broke**	**broken**	**brokeni**	**brows**	**browser**

Which training review does “brokeni” come from?

	REVIEW PRODUCING “brokeni”
Set of Training Reviews	**I got the car charger and not even after a week the charger was broken…I went to plug it in and it started smoking.**

What happened? Well, “broken…I” was first lowercased to “broken…i”, then punctuation was removed by the function removePunctuation(), which does not insert any white space character, and “broken…i” has become “brokeni”.

In further text mining and machine learning steps, if this shortcoming were not corrected, “brokeni” would be treated differently than “broken”, which can be seen on the same row, i.e. on row 24 of the tokens from training reviews.

This has to be corrected of course not just for “brokeni” but because this flaw is systemic (you only have to change the system once) and systematic (it could happen several or even many times).

C. Fine Tuning NLP

Instead of the function removePunctuation() from the package tm, specific for loops will be developed, preprocessing reviews according to the needs stated above and in a stepwise way:

- punctuation marks other than apostrophes will be replaced with white space characters instead of just being removed;
- short forms will be removed;
- remaining apostrophes will be replaced with white space characters;
- other stopwords will be removed (it is done in step 4 and not in step 2 in order to do it when absolutely all punctuation marks have been removed: please see example with “brokeni” where two words and one punctuation mark are stuck together…).

Among stopwords, short forms (contractions) need to be specifically treated. Additional needs of breakdown might also emerge. Starting from the stopword list delivered by the function stopwords(“english”) from the package tm, four CSV files will be produced.

These are the four files: - short_forms_pos.csv, with all positive short forms from stopwords(“english”) such as “she’s”, a few additional ones and numerous misspelled variants such as “she s” or “shes”; - short_forms_neg.csv, in the same approach, for short forms such as “isn’t”, “daren’t” but also “isn t”, “isnt”, etc.; - negation.csv, with seven negational unigrams such as “not” or “no”; - stopwords_remaining.csv, which is self-explanatory.

The 4 files have been uploaded to the GitHub repository https://github.com/Dev-P-L/Sentiment-Analysis (https://github.com/Dev-P-L/Sentiment-Analysis). They are going to be downloaded now and integrated into NLP pre-processing.

Let’s rebuild the corpus, the bag of words and the wordcloud.

In the wordcloud, there is no more token originating from short forms. Let’s have a broader look, building up a presentation table and checking whether abovementioned oddities have disappeared. Let’s check up in the bag of words whether “dont” has indeed disappeared.

50	divis	dna	dock	dollar	done

Yes, indeed, “dont” has disappeared. Let’s check up in the same way for “ive”!

96	issu	item	jabra	jack	jawbon

“ive” has also disappeared. Now “brokeni”.

21	break	breakag	broke	broken	brows

“brokeni” has vanished as well, just as many other oddities. I leave uncorrected some spelling errors, such as “disapoint” or “dissapoint”, because this is no repetitive structure and occurrence seems marginal.

Let’s have a first try at predicting sentiment on the basis of sentSparse_av0, which is our training set.

D. Measuring NLP Impact on Prediction Accuracy

The chosen machine learning model will be CART: it runs rather quickly and delivers clear decision trees. Running function rpart() on the training set delivers the accuracy level mentioned hereunder.

	ACCURACY ON THE TRAINING SET
Model: CART	**0.7784**

Now let’s train the rpart method with the train() function from the package caret.

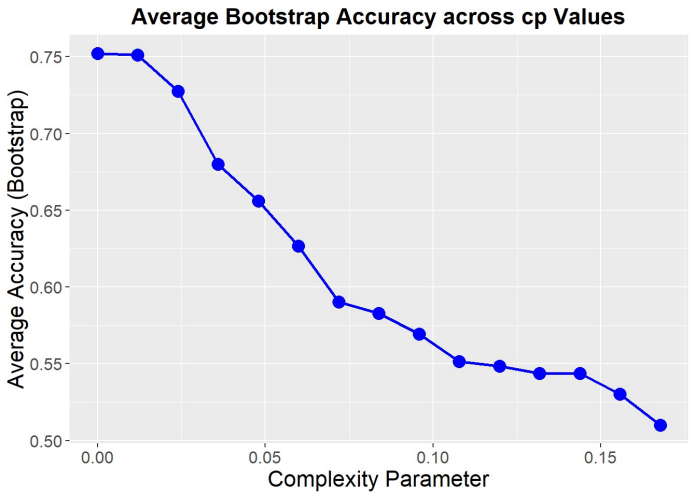
By default, the train() function would train across 3 values of cp (the complexity parameter) and 25 bootstrapped resamples for each tuned value of cp. As far as the number of tuned values is concerned, let’s upgrade it to 15 to increase the odds of improving accuracy, especially as rpart runs rather quickly.

The default resampling method is bootstrapping, samples being built with replacement, some reviews being picked up twice or more and some other reviews not being selected. This method seems especially appropriate here because the size of each resample will be the same of the size of the training set, which is already limited, i.e. 668. Working with e.g. K-fold cross-validation would imply further splitting the training set.

Will accuracy improve?

	ACCURACY ON THE TRAINING SET
Model: CART + cp Tuning	**0.7859**

Accuracy increases somewhat, i.e. from 77.8 to 78.6. For the record, let’s have a look at a graph showing how accuracy evolves across the 15 cp values chosen by the train() function.



On the graph above, maximum accuracy is a bit lower than the level previously indicated. Why is it different? Because, on the graph, it is, for each cp value, the average accuracy on the 25 bootstrapped resamples, while accuracy previously given related to the whole training set.

The optimal value of cp is near zero. Is it really zero?

	OPTIMAL cp VALUE
Model: CART + cp Tuning	**0.0000**

Yes, it is zero. This means that the train() function has kept the decision tree as complex as possible by assigning a zero value to the complexity parameter.

On the whole training set, the rpart model without tuning delivers approximately 78 % accuracy and the rpart model with tuning 79 %. Both levels are substantially higher than accuracy provided by the baseline model. The baseline model would predict a positive evaluation for all reviews (or alternatively a negative evaluation for all reviews) since prevalence is 50 %. Prevalence should show in the accuracy level delivered by the baseline model on the training set. Let's check it up.

	ACCURACY ON THE TRAINING SET
Model: Baseline	**0.5000**

Let's summarize results from the three models, not only with accuracy but also with additional performance metrics.

MODEL ID	SHORT DESCRIPTION	ACCURACY	SENSITIVITY	NEG PRED VAL	SPECIFICITY	POS PRED VAL
~~**baseline**~~	~~**baseline model**~~	~~**0.5000**~~	~~**1.0000**~~	~~**Div. by 0**~~	~~**0.0000**~~	~~**0.5000**~~
cart_av0	**CART**	**0.7784**	**0.6617**	**0.7257**	**0.8952**	**0.8633**
cart_tuned_av0	**CART + tuning**	**0.7859**	**0.7126**	**0.7493**	**0.8593**	**0.8351**

In the table above, on row 1, fonts have been stricken through to indicate that this model is discarded because if delivers only 50 % accuracy and looks like a dead-end path.

The other two models should be seen as a cumulative process bringing accuracy improvement in a stepwise and incremental way, with the one on green background being the best in accuracy. Models 2 and 3 deliver higher accuracy but also asymmetry between other performance metrics: sensitivity and negative predictive value are lower than specificity and positive predictive value. This reflects false negatives being more numerous than false positives. False negatives are predictions pointing to "Neg" while the reference value is " Pos". This is an insight for text mining: perusing false negatives and coming with actionable findings.

In order to confirm that false negatives are more numerous than false positives, let's have a look at the confusion matrix for both models. First, the confusion matrix from the rpart model without tuning.

	Actually positive	Actually negative
Predicted positive with CART	**TP = 221**	**FP = 35**
Predicted negative with CART	**FN = 113**	**TN = 299**

The weak point lies in the first column, on green background: the relatively high number of false negatives and, as a corollary, the relatively low number of true positives. On the reference positive class (" Pos" in label), predicting seems problematic or at the very least challenging since false negatives are rife. On the contrary, on the reference negative class ("Neg" in label), predicting has run smoothly, with a satisfactorily low number of false positives.

The tuned rpart model is expected to slightly reduce the excess in false negatives.

	Actually positive	Actually negative
Predicted positive with CART + tuning	**TP = 238**	**FP = 47**
Predicted negative with CART + tuning	**FN = 96**	**TN = 287**

With the tuned rpart model, accuracy has slightly improved: the sum of numbers on the main diagonal is larger.

On the green background, predicting on the reference positive class is less prolific in false negatives and, as a corollary, true positives are more predominant. On the secondary diagonal, imbalance between false negatives and false positives is less marked, not only because there are less false negatives but also because there are more false positives. Nevertheless false negatives remain the weak point, still being twice as numerous as false positives.

False negatives - and false positives - will be perused through text mining in the next section, looking for new insights towards accuracy improvement.

V. INSIGHTS from TEXT MINING

Let's now have a look at the false negatives, which constitute a challenge, from the CART model with tuning.

A. Visualizing False Negatives

	Review Corresponding to a FALSE NEGATIVE with CART + Tuning	After NLP	Usable Subjective Information
1	The phone loads super!	phone load super	super
2	This phone is very fast with sending any kind of messages and web browsing is significantly faster than previous phones i have used.	phone fast send kind messag web brows signific faster previous phone use	fast + faster
3	The color is even prettier than I thought it would be, and the graphics are incredibly sharp.	color even prettier thought graphic incred sharp	prettier + sharp
4	I'm still infatuated with this phone.	still infatu phone	infatu
5	Now I know that I made a wise decision.	now know made wise decis	wise
6	Jawbone Era is awesome too!	jawbon era awesom	awesom
7	Virgin Wireless rocks and so does this cheap little phone!	virgin wireless rock cheap littl phone	rock
8	I've had no trouble accessing the Internet, downloading ringtones or performing any of the functions.	troubl access internet download rington perform function	[no trouble]
9	Their Research and Development division obviously knows what they're doing.	research develop divis obvious know	[knows what they're doing]
10	Product is exactly as described.	product exact describ	[as described]
11	It seems completely secure, both holding on to my belt, and keeping the iPhone inside.	seem complet secur hold belt keep iphon insid	secur
12	Seller shipped quickly and much cheaper than the competitors.	seller ship quick much cheaper competitor	quick

There are three scenarios, each illustrated on a different background color. Let's get started with the first scenario on the blue background.

B. Subjective Information Unigrams

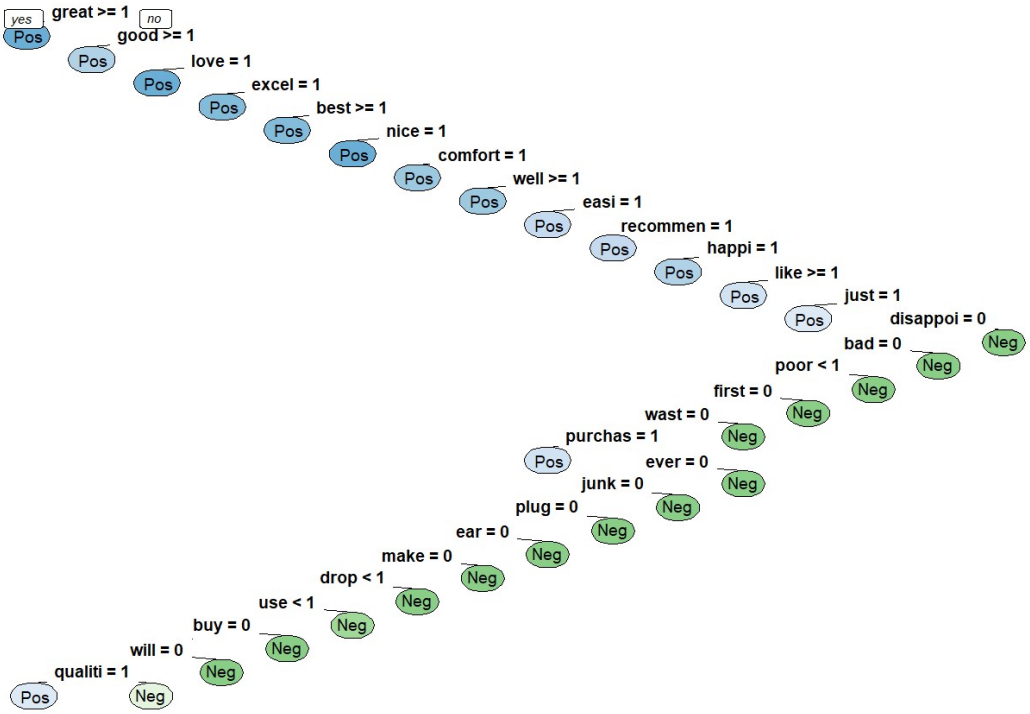
On the blue background in the table above, there are 9 cases out of 12. In each of these 9 reviews, there is at least one token with subjective information that is clear from a human point of view. Let's just pinpoint "super", "fast", "faster", "prettier", "infatuated", "wise", "awesome", "rocks", etc., or, after stemming "super", "fast", "faster", "prettier", "infatu", "wise", "awesom", "rock", etc.

These words, and the related standardized tokens, can be classified in two categories:

- compliance-related tokens, expressing compliance or incompliance with expectations, requirements or advertisements ("super", "faster", "rocks", etc.),
- sentiment-related tokens other than in previous category ("infatuated").

The difference between the two categories can sometimes be tricky and both categories will be referred to altogether in this project using phrases such as “subjective information” or “tokens conveying subjective information”.

“Unfortunately”, the tokens pinpointed among the false negatives do not show in the decision trees. Let’s visualize the decision tree from the CART model with tuning.



What types of tokens can be seen in the decision tree?

There is a majority of tokens conveying subjective information (“great”, “comfort”, “love”, “like”, “disappoi”, etc.). They are usually the highest ranked.

There are also other types of tokens, but at a lower level: - intent-related tokens (“purchas”, “buy”) or - topic-related tokens (“plug”, “ear”).

Which is an interesting insight. In CART, tokens conveying subjective information predominate, which is not at all surprising! This points to solutions allocating higher priority to tokens conveying subjective information.

Although a majority of tokens are conveying subjective information in the decision tree, we do not find the many tokens with subjective information pinpointed among false negatives. It is probably a matter of word (or token) frequency. This can be first checked up in the wordcloud that has already been visualized.

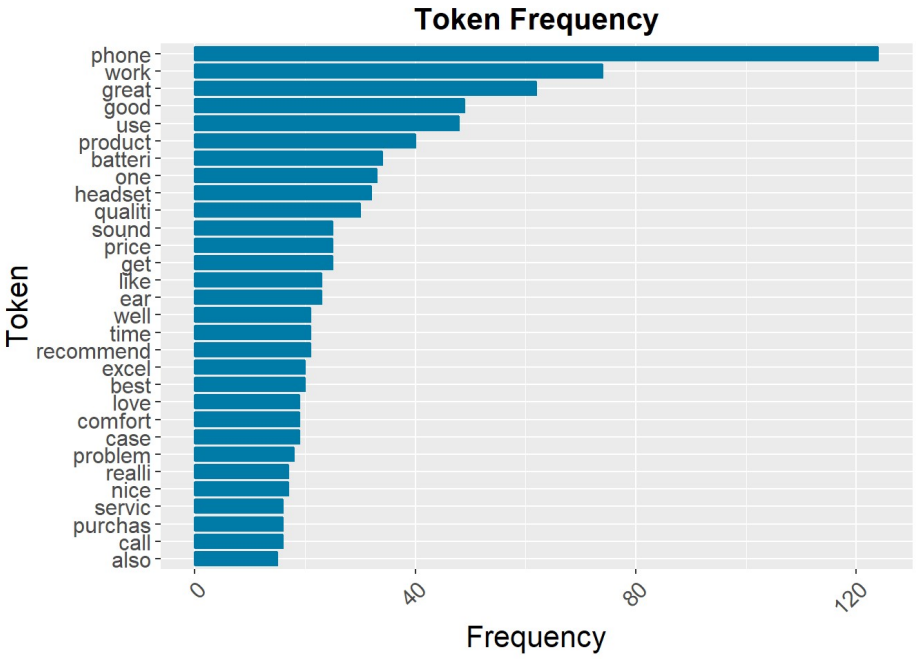
The wordcloud, which requires a minimal frequency of 10, is not comprised of the tokens with subjective information pinpointed among the false negatives.

Among tokens depicted in the wordcloud, there are - topic-related tokens (“phone”, “batteri”, “headset”, “sound”, “ear”, etc.), - intent-related tokens (“purchas”, “buy”, “bought”), - compliance-related tokens, expressing compliance or incompliance with expectations, requirements or advertisements (“fit”, “comfort”, “problem”, etc.), - sentiment-related tokens other than in previous category (“love”, “like”, etc.).

“phone”, which is topic-related, is the token with the highest frequency. Other topic-related tokens rank among the highest frequencies: “batteri”, “product”, “headset”, “sound”, etc.

“phone” has the highest frequency in the wordcloud but does not show in the decision tree. The same holds for other highly ranked topic-related tokens.

For illustrative purposes, tokens can be visualized in decreasing order of frequency in the graph below.



This looks like an interesting insight. Token frequency should not be the only criterion. Actually, there should be some hierarchy between types of tokens (subjective information, intent-related, topic-related). This does not necessarily mean a strict hierarchy between categories at a first level and then between tokens at a second level; there could be some intermixture, token frequency could coexist with prioritization of subjective information.

In conclusion, it might be impactful to garner subjective information conveyed by tokens such as “super”, “prettier”, “infatu”, “awesom”, etc. Since CART doesn’t do it, why not replace such tokens with a generic token either positive or negative? This would empower subjective information by building high frequency generic tokens only typified by sentiment orientation.

In this project, polarity of some tokens conveying subjective information will be inserted in additional files. That is one avenue of improvement that will be investigated in VI. INFORMATION RETRIEVAL thanks to INSIGHTS.

C. Negation Handling

Let’s go back to the table above that is comprised of 12 reviews corresponding to false negatives from the model cart_tuned. More particularly, let’s retrieve the review with a negational unigram.

	Review Corresponding to a FALSE NEGATIVE with CART + Tuning	After NLP	Usable Subjective Information

	Review Corresponding to a FALSE NEGATIVE with CART + Tuning	After NLP	Usable Subjective Information
8	I've had no trouble accessing the Internet, downloading ringtones or performing any of the functions.	troubl access internet download rington perform function	[no trouble]

The bigram "no trouble" is clear from a human point of view but this bigram has become "troubl", the negational token "no" having been removed with all other stopwords. Even if "troubl" is polarized under a generic negative token, as suggested above, the right polarity of "no trouble" wouldn't show. Two avenues are opened up: the whole bigram "no trouble" could be converted into a generic token with positive orientation or, more generally, negational stopwords such as "not" or "no" could no longer be removed, which is another avenue for improvement.

In this sample, there is only one case out of 12 with an ignored negational phrase. Is it worthwhile heading for properly dealing with negational unigrams? Let's integrate negational unigrams into the bag of words and produce a wordcloud in order to get some pre-attentive insight about frequency of negational unigrams such as "not" or "no".

"not" appears in the wordcloud and is highly ranked, just below "phone". This is an insight advocating inclusion of negational unigrams.

Let's have a look at word associations with "not": does "not" often reverse review polarity?

Token	Correlation with "not"
impress	0.21
compani	0.15
enough	0.14
plug	0.14
motorola	0.13
unit	0.13
will	0.13
charger	0.12
item	0.12
broke	0.11
connect	0.10
charg	0.09

Token	Correlation with “not”
contact	0.09
player	0.09
drop	0.08
minut	0.08
problem	0.08
replac	0.08
sinc	0.08
went	0.08
design	0.07
work	0.07
car	0.06
never	0.06
new	0.06
start	0.06
tri	0.06
cell	0.05
wear	0.05

On the list, the first stemmed unigram associated with “not” is “impress”. Let’s localize the reviews producing “impress” to get an idea of frequency.

	Training Review Containing “impressed”
1	He was very impressed when going from the original battery to the extended battery.
2	Not impressed.
3	great headset, very impressed - H500.
4	I am very impressed with this headset from Plantronics.
5	I am not impressed with this and i would not recommend this item to anyone.
6	All it took was one drop from about 6 inches above the kitchen counter and it was cracked.I am not impressed and I am not laughing.

In the three training reviews on green background, there are three reviews containing “not impressed”. Moreover, in the second of these reviews, there is also “not recommend” and in the third “not laughing”. This is an additional illustration of the usefulness of including negational unigrams in one way or another. It will be given a try in VI. INFORMATION RETRIEVAL thanks to INSIGHTS.

Up to now, negation has been shown in negational unigrams but negation can also be encapsulated into short forms (also called “contractions”). The previous sample of false negatives, which has been selected at random, is not comprised of negative short forms. But the same random procedure, applied to false positives, delivers several of them.

	Review Corresponding to a FALSE POSITIVE with CART + Tuning	After NLP	Usable Subjective Information
1	Not a good bargain.	good bargain	not a good
2	All I can do is whine on the Internet, so here it goes.The more I use the thing the less I like it.	can whine internet goe use thing less like	whine + ? the less
3	I still maintain that monkeys shouldn’t make headphones, we just obviously don’t share enough DNA to copy the design over to humans.	still maintain monkey make headphon just obvious share enough dna copi design human	shouldn’t

	Review Corresponding to a FALSE POSITIVE with CART + Tuning	After NLP	Usable Subjective Information
4	I put the latest OS on it (v1.15g), and it now likes to slow to a crawl and lock up every once in a while.	put latest os v g now like slow crawl lock everi	slow + crawl + lock-up
5	The plug did not work very well.	plug work well	did not work well
6	Still Waiting…… I'm sure this item would work well.. if I ever recieve it!	still wait sure item work well ever reciev	still waiting
7	My experience was terrible….. This was my fourth bluetooth headset, and while it was much more comfortable than my last Jabra (which I HATED!!!	experi terribl fourth bluetooth headset much comfort last jabra hate	terrible
8	Also difficult to put on.I'd recommend avoiding this product.	also difficult put recommend avoid product	difficult
9	I only used it two days, and it wasn't always easy to hear with.	use two day alway easi hear	wasn't always easy
10	I am sorry I made this purchase.	sorri made purchas	sorry
11	Not as good as I had hoped.	good hope	not as good
12	I don't like this Nokia either.	like nokia either	don't like

In the table above, there are seven negation forms, which machine learning could not take into account since these negation forms had been removed through NLP. So, "Not a good bargain." has become "good bargain" and "don't like" has mutated into "like"!

On the blue background, "not" comes thrice in three training reviews.

On the green background, there are four negative short forms.

This shows the importance of negation, be it under the form of negational unigrams ("not", etc.) or under the form of negational short forms.

In the table above, interpretation of some training reviews is rather complex. They will be analyzed in the next section, as well as other complex reviews shown before.

D. Subjective Information Multigrams

Sentiment can also be expressed through associations of words beyond negation cases already treated.

In some cases, these are rather stereotyped phrases. Let's go back to two examples already provided by false negatives.

	Review Corresponding to a FALSE NEGATIVE with CART + Tuning	After NLP	Usable Subjective Information
9	Their Research and Development division obviously knows what they're doing.	research develop divis obvious know	[knows what they're doing]
10	Product is exactly as described.	product exact describ	[as described]

In the table above, there are two rather stereotyped phrases with usable information: "knows what they're doing" (competency statement) or "as described" (compliance statement). Some of these phrases will be listed (with spelling variants) among positive multigrams and instances of them in reviews will be replaced with a generic token with positive orientation.

False positives have delivered some more difficult cases: reviews with figurative wording, sarcasm, irony, metaphors, multifaceted reviews, etc.

	Review Corresponding to a FALSE POSITIVE with CART + Tuning	After NLP	Usable Subjective Information

	Review Corresponding to a FALSE POSITIVE with CART + Tuning	After NLP	Usable Subjective Information
2	All I can do is whine on the Internet, so here it goes.The more I use the thing the less I like it.	can whine internet goe use thing less like	whine + ? the less
3	I still maintain that monkeys shouldn't make headphones, we just obviously don't share enough DNA to copy the design over to humans.	still maintain monkey make headphon just obvious share enough dna copi design human	shouldn't
4	I put the latest OS on it (v1.15g), and it now likes to slow to a crawl and lock up every once in a while.	put latest os v g now like slow crawl lock everi	slow + crawl + lock-up
7	My experience was terrible….. This was my fourth bluetooth headset, and while it was much more comfortable than my last Jabra (which I HATED!!!	experi terribl fourth bluetooth headset much comfort last jabra hate	terrible

The table above gives four examples of more complex wording:

- figurative word such as "whine" instead of e.g. "disappointed";
- figurative word such as "crawl";
- sarcasm and metaphors about "monkeys";
- multifaceted review such as "My experience was terrible….. This was my fourth bluetooth headset, and while it was much more comfortable than my last Jabra (which I HATED!!!".

What will be done here: listing some unigrams or multigrams such as "whine", "shouldn't", "lock up" as negatively oriented and replacing instances of them in training reviews with one generic negative token.

E. Topic-related Tokens

It has been seen that some topic-related tokens have high frequency but do not appear in the decision tree. So, "phone" has the highest frequency but is not present in the decision tree. "batteri", "headset" do not even appear in the decision tree either.

This difference between wordcloud and decision tree might be insightful. Should topic-related tokens be discarded except the few ones that appear in the decision tree? Should they be excluded to make the bag of words less messy? But would they systematically be excluded as well in other machine learning models?

To get some confirmation or infirmation, let's apply another algorithm. A Random Forest model is chosen because it allows for some rough ranking of predictor impact. First, here is the accuracy level on the training set.

	ACCURACY ON THE TRAINING SET
Model: Random Forest	0.9626

Would the high accuracy level mean that this model solves all issues and that further text mining and further machine learning are both useless? I do not think so: very often, at least on the basis of my experience, rf has a tendency to overfitting. Results on the validation set might be substantially lower. Consequently, text mining remains meaningful and worth performing.

Let's check whether topic-related tokens that do not show in the CART decision tree are positioned at a relatively high level in the predictor importance list from Random Forest.

Stemmed Token	Predictor Importance in Random Forest
great	100.00
love	40.42
good	38.37
excel	36.58
best	32.70
nice	26.88
comfort	25.84
well	23.93

Stemmed Token	Predictor Importance in Random Forest
recommend	18.27
price	17.34
sturdi	16.74
easi	16.70
phone	16.57
happi	16.45
like	15.93
seem	11.26
fine	10.85
awesom	10.64
disappoint	10.62
product	10.61

Actually, the Random Forest importance list is rather different than the CART decision tree: e.g. “phone”, which does not appear in the CART decision tree, is ranked in the 13th position in the Random Forest importance list. In the first 13 positions on the Random Forest list, there are two topic-related tokens (“price” and “phone”) while there is none in the first 13 positions in the CART decision tree. In a snapshot, two topic-related tokens get some importance in Random Forest and the Random Forest algorithm performs better: that is a reason to keep topic-related tokens in the bag of words. Especially so since other models than CART will be trained in the machine learning section.

Consequently, topic-related tokens that do not show in the CART decision tree will not be discarded.

By the way, Random Forest has outperformed CART in accuracy but there are false negatives and positives with Random Forest as well, even if less numerous. Here is the confusion matrix from Random Forest.

	Actually positive	Actually negative
Predicted positive with Random Forest	319	10
Predicted negative with random Forest	15	324

As expected, the number of false negatives and false positives is more limited than with CART since accuracy is larger. It is also true for each group separately: less false negatives in Random Forest and less false positives.

Let’s have a look at the false negatives, which are more numerous. Here is a sample of the false negatives from Random Forest.

Review Corresponding to a FALSE NEGATIVE with Random Forest	After NLP	Usable Subjective Information
)Setup couldn’t have been simpler.	setup simpler	simpler
Gets the job done.	get job done	job done
Incredible!.	incred	incred
Someone shouldve invented this sooner.	someon shouldv invent sooner	shouldv invent sooner
This PDA rocks.	pda rock	rock
The noise shield is incrediable.	nois shield incredi	incredi
This fixes all the problems.	fix problem	fix problem
These are fabulous!	fabul	fabul
Perfect for the PS3.	perfect ps	perfect
2 thumbs up to this seller	thumb seller	thumbs up
I have read other’s reviews here but I haven’t had any problem with it.	read s review problem	any problem

Review Corresponding to a FALSE NEGATIVE with Random Forest	After NLP	Usable Subjective Information
Leopard Print is wonderfully wild!.	**leopard print wonder wild**	**wonder**

On the blue background, there are training reviews with usable stemmed unigrams indicating positive polarity: “fabul”, “perfect”, etc.

On the green background, there are positively oriented multigrams such as “thumbs up”.

This supports previous insights.

F. Conclusion on Text Mining Insights

In text mining, insights have been obtained

- by comparing token frequency in the bag of words (wordcloud and histogram) and in token lists from CART and Random Forest
- and by perusing false negatives and positives in CART and Random Forest.

Among insights, let’s mention:

- topic-related tokens predominate in the bag of words;
- topic-related tokens show in limited number and at a lower level in the CART decision tree and in the predictor importance list from Random Forest;
- subjective information tokens predominate in the CART decision list and in the predictor importance list from Random Forest;
- many subjective information tokens show in false negatives or false positives but neither in the CART decision tree nor in the predictor importance list from Random Forest (first 20 positions);
- many negational n-grams are present in reviews giving false negatives or positives and often reverse sentiment polarity of the reviews but they cannot be made actionable in machine learning since they have been removed from the bag of words as stopwords;
- these negational n-grams can be negational unigrams such as “not” or negative short forms such as “isn’t”.

In the next section, these text mining insights will be tentatively transposed into NLP and machine learning actions towards more accuracy.

Three avenues of improvement have been opened up:

- integrating negational unigrams (“not”, etc.);
- integrating negative short forms (“isn’t”, etc.);
- establishing polarized lists of subjective information tokens and replacing instances of these tokens in reviews with one generic token, either positive or negative.

Stepwise, the three avenues will be quantitatively tested.

The whole research has been performed only on training reviews without any kind of intermixture with validation reviews.

VI. INFORMATION RETRIEVAL thanks to INSIGHTS

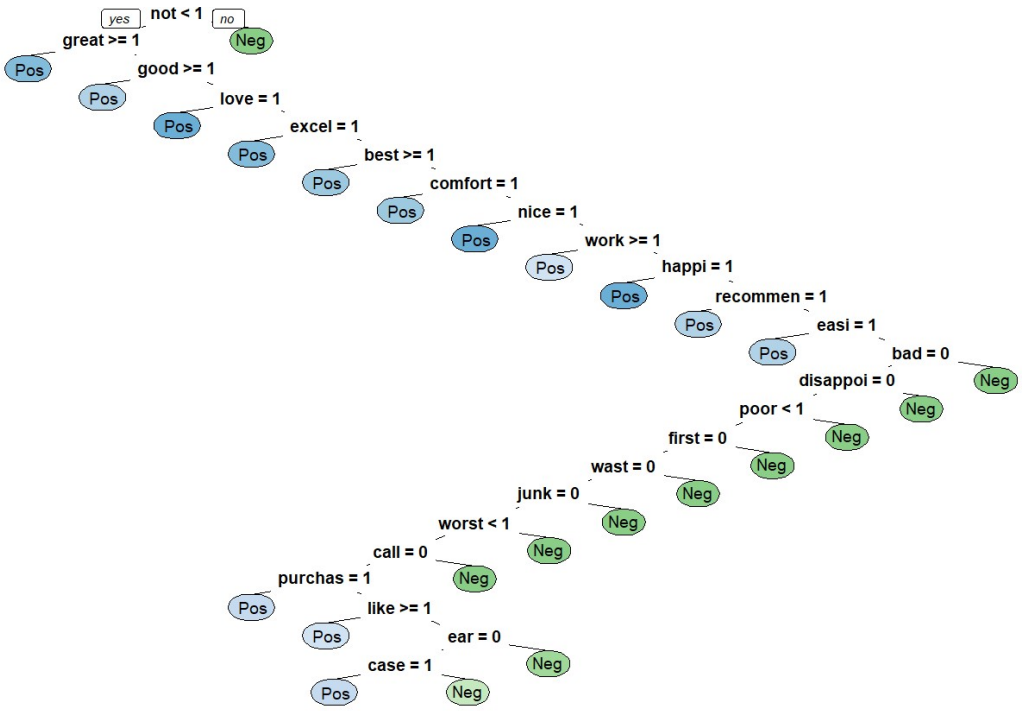
A. Integrating Negational Unigrams

Negational unigrams have been introduced, NLP has be rerun as well as the CART model with tuning, which is used as a performance yardstick. Here are the results.

	ACCURACY ON THE TRAINING SET
Model: Neg Short Forms + CART + Tuning	**0.8083832**

There is accuracy improvement approximately from 79 % to 81 %. Consequently, negational unigrams such as “not” will be kept in the corpus.

For the record, does “not” show in the decision tree?



Yes, indeed it does and rather predominantly!

B. Retrieving Information Included in Negative Short Forms

1. Adding Negative Short Forms to the Bag of Words

First, negative short forms will no longer be removed from the corpus and will thus enter the bag of words. Impact on accuracy will be tested.

	ACCURACY ON THE TRAINING SET
Model: Negation + Neg Short Forms + CART + Tuning	**0.8083832**

Adding negative short forms such as “isn’t” does not impact accuracy level. Consequently, another try will be done with negative short forms: instead of being added, negative short forms will be replaced with “not”.

2. Replacing Negative Short Forms with " not "

Positive short forms will still be removed from the corpus with the function removeWords() from the package tm, which removes separate words and no substrings.

Negative short forms will be searched for with the function gsub(). To avoid picking up substrings as well, one white space character will be added in front of all negative short forms and at the end of each of them. Indeed, let’s not forget that misspelled short forms such as “dont” have been deliberately introduced as well among negative short forms to take into account “alternative grammar”, which is omnipresent in reviews. And words such as “dont” can be substrings out of other words. The gsub() function will replace the “escorted” short forms with " not “.

Let’s have a look at the new accuracy level.

	ACCURACY ON THE TRAINING SET
Model: Negation + [Neg Short Forms = “not”] + CART + Tuning	**0.7979042**

Replacing negative short forms with " not " downgrades accuracy. This path will not be followed.

C. Polarization - Text Classsification - Text Substitution

In samples of false negatives and false positives, analysis has pinpointed unigrams and multigrams that convey subjective information.

In the line of these insights, these n-grams have been listed and classified as positive and negative. They have been inserted into four files and the files have been uploaded in the GitHub repository https://github.com/Dev-P-L/Sentiment-Analysis (https://github.com/Dev-P-L/Sentiment-Analysis) :

- subj_pos_multigrams.csv,
- subj_pos_unigrams.csv,
- subj_neg_multigrams.csv,
- subj_neg_unigrams.csv.

Here are a few examples from each file of polarized n-grams.

Positive sentiment oriented unigrams from subj_pos_unigrams.csv (stemmed): “super”, “awesom”, etc.

Some positive multigrams from the file sub_pos_multigrams.csv (not stemmed): “no trouble”, “5 stars”, “thumbs up”, “it’s a ten”, “as described”, “know what they’re doing”, “must have”. Possible variants have usually been added, including variants originating from spelling errors or “alternative grammar”: “no troubles”, “not any trouble”, “not any troubles”, “no problem”, “no problems”, etc.; “five stars”, “five star”, “5-star”, “5star”, “5 star”; “it’s a 10”, “it’s a ten”, “its a 10”, etc.; “know what theyre doing”, “know what they are doing”, etc.

Some negative unigrams (after stemming) from the file subj_neg_unigrams.csv: “horribl”, “crap”, “whine”, etc.

Some negative multigrams (not stemmed) from the file sub_neg_multigrams.csv: “1 star”, “one star”, “not good”, “no good”, “shouldn’t” (often associated with negative context), “pretty piece of junk”, etc.

In the training reviews, instances of the positive n-grams will be replaced with " subjpo " and instances of negative n-grams with " subjneg “.

Efficacy-minded rules will be applied in this NLP process.

First, the polarized n-grams will be preceded and followed by one white space character when looking for instances in reviews. Otherwise, in the bag of words, the n-gram “most inconvi” would become “most in subjpo” (because “convi” is a polarized unigram in subj_pos_unigrams.csv) and then " subjpo " (because “most” and “in” are stopwords in stopwords_remaining.csv)! A negatively oriented multigram would become a positively oriented unigram! Consequently, one white space character is added in front of and at the end of each polarized n-gram before looking for matching instances in NLP-transformed reviews, in order to avoid replacing substrings.

Second, as a consequence, a white space character has to be added at the beginning and at the end of each NLP-transformed review! Otherwise, polarized n-grams, which are preceded and followed by one white space character can never match an instance that is positioned at the beginning or at the end of a review.

Third, " subjpo " and " subjneg " contain one white space character at the beginning and at the end, in order to prevent amalgamation. Indeed, what would happen if white space characters were not added? Let’s take our well known example of " conveni “: if it were replaced with just”subjpo" in the n-gram " most conveni “, then it would produce” mostsubjpo“, which would no longer be a generic positive unigram! Transformation would be useless if not annoyingly counterproductive!

Fourth, multiple inter word white space characters have to be reduced to a single inter word white space character: indeed, listed multigrams only have one white space character between words and could never match multigrams from reviews with several white space characters between words.

Fifth, in training reviews, negative multigrams have got to be replaced before positive multigrams. Let’s take the example of " not a good bargain “, which is a negatively polarized multigram from the file subj_neg_multigrams.csv: if matching with instances in reviews begins with positively polarized n-grams, then” not a good bargain " in a review becomes " not a subjpo “, which might be less clear in machine learning than” subjneg “! For similar reasons, positive multigrams are matched before negative unigrams and positive unigrams.

Sixth, negative or positive polarized multigrams should be tentatively matched in decreasing order in for loops. Why? Let’s take the example of " no good bargain " in one review. In sub_neg_multigrams.csv, there are two negatively polarized multigrams: " no good bargain " and " no good “; if these are considered in decreasing order, then, in the review,” no good bargain " is replaced with " subjneg “, which looks appropriate; otherwise” no good bargain " is replaced with " subjneg bargain " and then " subjneg subjpo “: consequently, instead of having one negative generic unigram we would get one positive and one negative generic unigrams!

NLP will be rerun again. In each training review, all n-grams that match positive n-grams from subj_pos_multigrams.csv or subj_pos_unigrams.csv will be replaced with a generic positive token (" subjpo “); all n-grams that match negative n-grams from subj_neg_multigrams.csv or subj_neg_unigrams.csv will be replaced with a generic negative token (” subjneg “).

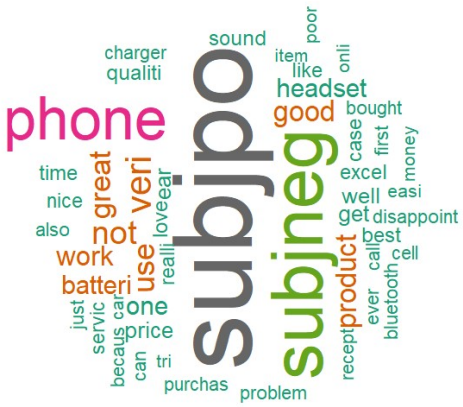
The utf8 package will be used to normalize punctuation: there has been some trouble with curly apostrophes instead of straight apostrophes.

	ACCURACY ON THE TRAINING SET

	ACCURACY ON THE TRAINING SET
Model: Negation + Polarization + CART + Tuning	**0.9176647**

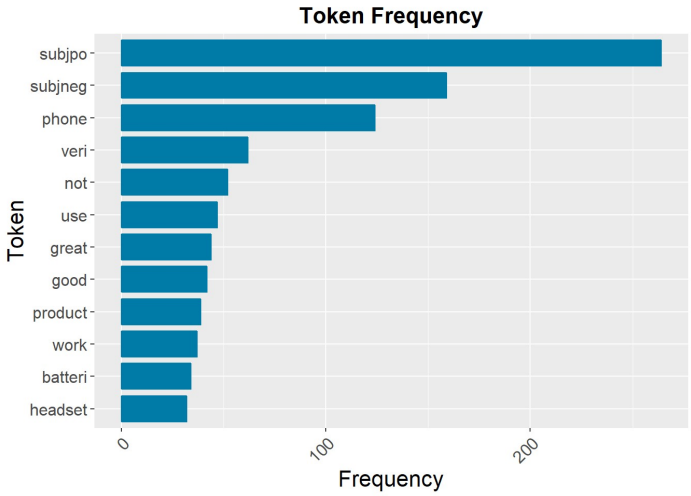
Polarizing n-grams that convey subjective information has dramatically boosted accuracy from approximately 81 % to 92 %. Nevertheless, let us stay realistic: text mining has been performed on the training set and is not necessarily, or at least not necessarily entirely, transposable to the validation set.

To get some visual representation of changes, let's have a look at a wordcloud from the new bag of words.

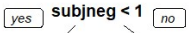


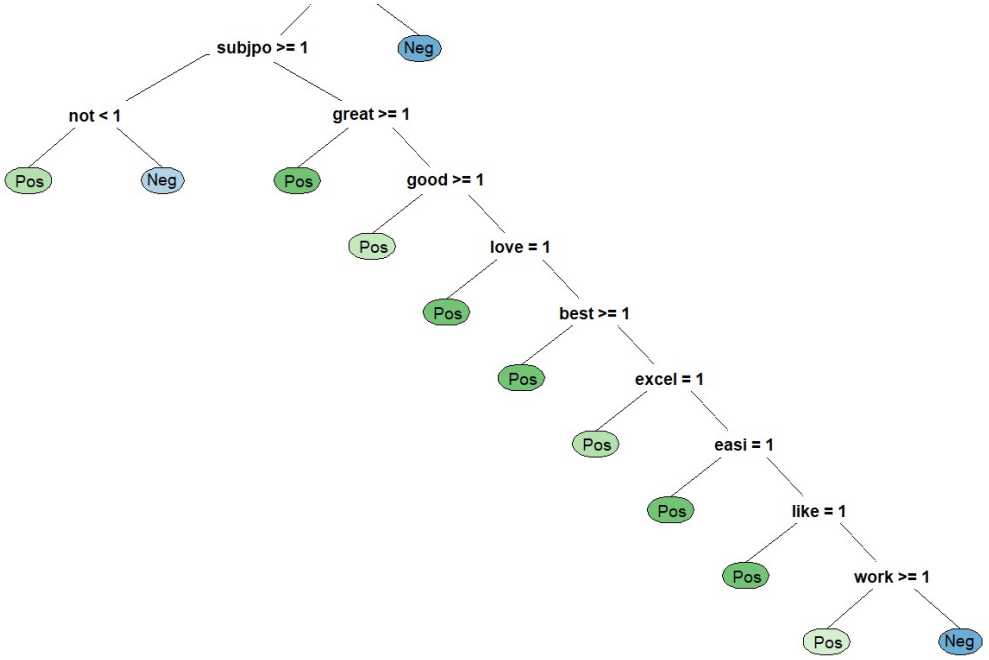
The whole picture has changed. The most frequent tokens are now "subjpo" and "subjneg". "not" is present, at the "fourth level" (graphically).

The following histogram adds some quantitative insight.



Is there any parallel evolution in the decision tree?





Yes, indeed there is now more parallelism between wordcloud/histogram and decision tree! The most frequent tokens in the wordcloud are now at the top of the decision tree. Indeed, the first node is occupied by “subjneg”; then comes “subjpo” and at the same level “not” and “great”. Many individual tokens that were previously in nodes of the tree, have disappeared.

The next table summarizes accuracy results obtained so far.

MODEL ID	SHORT DESCRIPTION	ACCURACY	SENSITIVITY	NEG PRED VAL	SPECIFICITY	POS PRED VAL
~~baseline~~	~~baseline model~~	~~0.5000~~	~~1.0000~~	~~Div. by 0~~	~~0.0000~~	~~0.5000~~
cart_av0	CART	0.7784	0.6617	0.7257	0.8952	0.8633
cart_tuned_av0	CART + tuning	0.7859	0.7126	0.7493	0.8593	0.8351
cart_tuned_av1_a	CART + tuning + negation	0.8084	0.7156	0.7601	0.9012	0.8787
~~cart_tuned_av1_b~~	~~CART + tuning + negation + neg short forms~~	~~0.8084~~	~~0.7156~~	~~0.7601~~	~~0.9012~~	~~0.8787~~
~~cart_tuned_av1_c~~	~~CART + tuning + negation + neg short forms = “not”~~	~~0.7979~~	~~0.6587~~	~~0.733~~	~~0.9371~~	~~0.9129~~
cart_tuned_av2	CART + tuning + negation + polarization	0.9177	0.8862	0.893	0.9491	0.9457

In the table above, on rows 1, 3 and 4, fonts have been stricken through to indicate that these models have been discarded. The other two models should be seen as a cumulative process bringing accuracy improvement in a stepwise and incremental way.

As shown in the last two rows, thanks to polarization, accuracy has jumped approximately from 81 % up to 92 %, which is impressive.

More impressive: sensitivity has sprung from 66 % to 89 %. This is linked to false negative management. False negatives have been a recurrent weak point in machine learning results up to now. But special attention has been paid to them in debriefing previous machine learning results and in perusing random samples of false negatives and false positives.

Let’s have a look at the numbers of remaining false negatives and positives in the following confusion matrix.

	Actually positive	Actually negative
Predicted positive AFTER POLARIZING	296	17
Predicted negative AFTER POLARIZING	38	317

With respect to the last accepted model, the number of false negatives has been crushed from 114 to 38; parallelwise, the number of true negatives has climbed from 220 to 296. There is also a decrease in false positives, much more modest though, from 22 to 17.

After improving accuracy thanks to NLP and text mining, new accuracy improvements will be looked for in the next section through machine learning optimization.

VII. MACHINE LEARNING OPTIMIZATION on the TRAINING SET

Ten models are going to be applied.

A. 10 Machine Learning Models

Ten machine learning models have been chosen. Here is the list, with an identifier and a short description for each model.

MODEL ID	MODEL	NAME IN CARET	# TUNING VALUES	# BOOTSTRAPPED RESAMPLES
adaboost_03	**AdaBoost Classification Trees**	**adaboost**	**3**	**25**
cart_03	**CART**	**rpart**	**3**	**25**
gbm_03	**Stochastic Gradient Boosting**	**gbm**	**3**	**25**
monmlp_03	**Monotone Multi-Layer Perceptron Neural Network**	**monmlp**	**3**	**25**
rf_03	**Random Forest**	**rf**	**3**	**25**
svm_03	**Support Vector Machines with Radial Basis Function Kernel**	**svmRadialCost**	**3**	**25**
xgb_03	**eXtreme Gradient Boosting**	**xgbLinear**	**3**	**25**
cart_15	**CART**	**rpart**	**15**	**25**
gbm_15	**Stochastic Gradient Boosting**	**gbm**	**15**	**25**
svm_15	**Support Vector Machines with Radial Basis Function Kernel**	**svmRadialCost**	**15**	**25**

All models will be trained with the train() function from the package caret.

For four models, by default, training has been done on 25 bootstrapped resamples and on 3 values of the parameters tuned. These models are “adaboost_03”, “rf_03”, “xgb_03” and “monmlp_03”. For three models that run faster on my PC, two options have been applied: either default training (“cart_03”, “svm_03” and “gbm_03) or training with tuning on 15 values across parameters tuned (”cart_15“,”svm_15" and “gbm_15”).

The names of the parameters tuned by the train() function are available in http://topepo.github.io/caret/available-models.html (http://topepo.github.io/caret/available-models.html).

B. Accuracy Results on the Training Set

The accuracy results are summarized in the next table.

MODEL ID	MODEL	# TUNING VALUES	# BOOTSTRAPPED RESAMPLES	ACCURACY ON THE TRAINING SET
rf_03	**Random Forest**	**3**	**25**	**0.9850**
adaboost_03	**AdaBoost Classification Trees**	**3**	**25**	**0.9820**

MODEL ID	MODEL	# TUNING VALUES	# BOOTSTRAPPED RESAMPLES	ACCURACY ON THE TRAINING SET
monmlp_03	Monotone Multi-Layer Perceptron Neural Network	3	25	0.9820
svm_15	Support Vector Machines with Radial Basis Function Kernel	15	25	0.9790
svm_03	Support Vector Machines with Radial Basis Function Kernel	3	25	0.9760
xgb_03	eXtreme Gradient Boosting	3	25	0.9461
gbm_15	Stochastic Gradient Boosting	15	25	0.9371
gbm_03	Stochastic Gradient Boosting	3	25	0.9237
cart_15	CART	15	25	0.9177
cart_03	CART	3	25	0.7754

Not surprisingly, most models seem to overfit. As pointed out in literature, it is a rather widespread tendency, especially so with complex algorithms which have many parameters they can minimize the loss function on.

Consequently, the ranking figuring in the table above is no appropriate tool to separate models.

Accuracy cannot be tested on the validation set since the validation set is only to be used at the last step; if some testing is performed on the validation set, it is no longer a validation set.

Moreover, the whole sample being very limited (1,000 reviews), it might have been counter-productive to divide it into training, test and validation sets instead of training and validation as has been done.

C. Testing on Bootstrap Resample Accuracy Distributions

But there are some unused resources. The train() function from caret has automatically trained the models on 25 bootstrapped resamples, for all values of the tuned parameters, i.e. on three values of the tuned parameters except where extra tuning on 15 values had been asked, i.e. CART, svm and gbm (chosen because running time was limited).

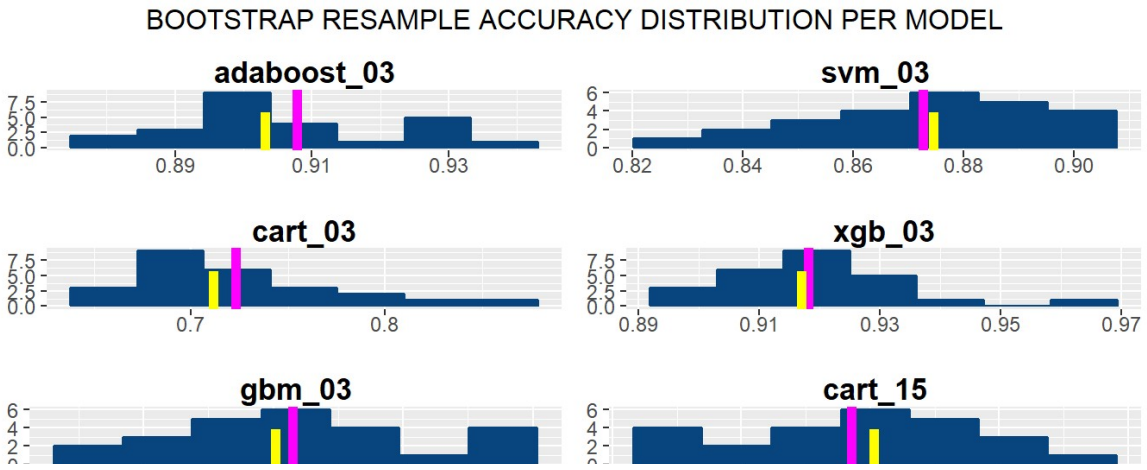
In each of the 10 models, accuracy is available for each of the 25 resamples: for each model and for each resample it is the accuracy level with the optimizing parameter value(s), i.e. the parameter value(s) that deliver(s) the highest accuracy level on average on all the 25 resamples.

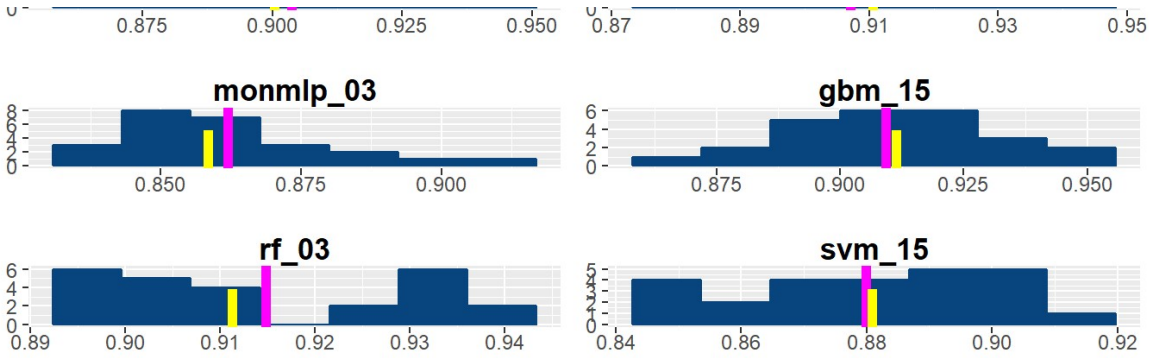
This produces for each model an empirical distribution of accuracy, i.e. a set of 25 accuracy levels corresponding to the optimizing parameter value(s).

Each resample is a sample of the training set with replacement. It has the same number of rows as the training set. That is why the bootstrapping method has been preferred to e.g. the K-fold cross-validation, which would have implied further splitting of an already small dataset.

For each model, the bootstrapped accuracy distribution can deliver precious indications: what's the average, the median, the maximum? It can tell about the generalizability of the accuracy levels computed.

First, accuracy distributions will be extracted from the 10 models. Ten graphs will be drawn with the resample accuracy distribution from the ten models.





On these graphs, the mean appears as a vertical magenta line and the median as a vertical dashed yellow line.

What's new in comparison with information delivered by predicting on the whole training set? Generally speaking, accuracy level has decreased, sometimes dramatically. This was expected.

Some of these empirical distributions are left-skewed and some are right-skewed.

From that point of view, rf_03, which was a solid candidate on the whole training set, shows a completely split empirical distribution with an empty middle, where the mean is isolated, and two blocks on the left and on the right. The rather concentrated left block seems problematic in terms of predictiveness on a validation set. A strong end of the left tail is problematic from a risk management viewpoint since several resamples show accuracy levels that are substantially lower than the average and the median.

To further evaluate the merits of the remaining models, let's produce a comparative table.

ID	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
xgb_03	0.8943	0.9061	0.9170	0.9182	0.9295	0.9608
rf_03	0.8943	0.9000	0.9113	0.9148	0.9309	0.9380
adaboost_03	0.8794	0.8984	0.9031	0.9078	0.9227	0.9380
cart_15	0.8735	0.8953	0.9106	0.9071	0.9183	0.9378
gbm_15	0.8653	0.8984	0.9113	0.9093	0.9227	0.9490
gbm_03	0.8653	0.8889	0.9004	0.9037	0.9142	0.9451
svm_15	0.8434	0.8661	0.8809	0.8799	0.8963	0.9095
monmlp_03	0.8313	0.8500	0.8584	0.8620	0.8723	0.9052
svm_03	0.8273	0.8584	0.8745	0.8727	0.8902	0.9024
cart_03	0.6426	0.6870	0.7114	0.7227	0.7617	0.8494

For five models, even the maximum resampled accuracy is lower than the training set accuracy: "rf_03", "ada_03", "monmlp_03", "svm_15", "svm_03".

For every model, the mean is lower than the accuracy level on the whole training set, even if means and medians are above 90 % for six models. The sharpest fall can be seen for the neural network model monmlp_03 (- 12 percentage points), followed by svm_15 and svm_03 (- 10 percentage points). The most resilient model is cart_15, which only loses 1 percentage point; it doesn't score badly, being at 2 percentage points in average from xgb_03, which has the highest average with almost 92 %.

On the whole training set, rf_03 was at the top but it has now been passed by xgb_03 in mean, median, maximum and first quartile value, but not in third quartile value; in minimum, they are even.

From a risk management point of view, attention should be paid, among others, to the minimum. That is why models have been ranked according to minimum, in decreasing order. To separate models, the second criterion is median (rather than mean since several distributions are skewed). According to this double criterion, xgb_03 is on top. Moreover, rf_03, which equals xgb_03 in minimum but not in median, has an already commented annoying concentration at the very left of its empirical distribution.

xgb_03 is the model selected to get validated on the validation set.

VIII. RESULTS on the VALIDATION SET

A. Constructing the Validation Set

In order to predict on the validation set, the training set and the validation set have to contain the same columns, i.e. the same labels (dependent variable) and the same predictors. For labels, there is no problem. Let's see about predictors.

The training set already exists, it is sentSparse_av2, which has been built up through NLP and with text mining insights (negational unigrams integrated and generic tokens replacing sentiment polarized tokens). The training set cannot be changed to match the validation set since this would imply, in one way or another, retrieving some information from the validation and instilling it into the training set, which is contrary with the very status of a validation set.

The validation set will be constructed in the same way as the training set, but independently: NLP, negational unigrams and polarization. Columns from the validation set will be aligned on the training set, of course not in content but in headers, in column names (i.e. in tokens). Columns that are in the validation set but not in the training set will be removed. Columns that are in the training set but not in the validation set will be added as null vectors to the validation set.

B. Predicting on the Validation Set

The xgb_03 model can now be validated on the validation set: predictions will be computed on the validation set and then accuracy.

	ACCURACY ON THE VALIDATION SET
Baseline Model	**0.5000**
Final Model: NLP + Text Mining + XGBoost + Default Tuning	**0.8795**

On the validation set, xgb_03 provides an accuracy level of 88 %, which is substantially higher than the 50 % accuracy level from a baseline model.

C. Attributing Accuracy Improvement per Methodological Layer

On the validation set, the baseline delivers an accuracy level of 50 % (just as on the training set). On the validation set again, the final model provides accuracy of 88 %, which is substantially higher.

Where does the improvement come from? From which step in the whole process: NLP, text mining or machine learning optimization? Splitting contribution to results needs to be done on the validation set since results on the training set can be boosted by overfitting. To evaluate input from each layer, a simple approach will we followed.

On the validation set without polarization and without negation, prediction will be conducted and accuracy produced. The method will be cart_15, i.e. CART with 15-value tuning and 25 bootstrapped resamples: this model is rather fast and can be considered as a yardstick, having proved resilient although not optimal. This will help evaluate the impact of NLP.

Then, on the validation set with negation, a second evaluation will be conducted with cart_15: this will help evaluate the impact of integrating negation.

On the validation set with negation and polarization, a third evaluation will be conducted with cart_15: this will help evaluate the impact of polarization.

The accuracy difference between this last result and the result from eXtreme Gradient Boosting will measure the impact of machine learning optimization.

METHODOLOGY	ACCURACY ON THE VALIDATION SET
Baseline Model	**0.5000**
NLP + cart_15	**0.7169**
NLP + Negation + cart_15	**0.741**
NLP + Negation + Polarization + cart_15	**0.8434**
NLP + Negation + Polarization + xgb_03	**0.8795**

88 % prediction accuracy has been reached on the validation set, against 50 % with a baseline model. Which factors have contributed towards that improvement with 38 percentage points?

Natural Language Processing has contributed 21.7 percentage points.

Text mining has brought additional accuracy improvement with 12.7 percentage points.

Machine learning optimization has boosted accuracy with 3.6 additional percentage points.

IX. SUMMARY & CONCLUSION

In this sentiment analysis project, a three-tier approach has lifted accuracy out of a baseline 50 % to 88 %: NLP (22 %), text mining (13 %) and machine learning optimization (4 %).

The Executive Summary, at the very beginning of this document, provides a nice overview. A dynamic table of content allows easy access to more detailed information. In particular, the main insights from text mining can be found in “VI. INFORMATION RETRIEVAL USING INSIGHTS, C. Polarization - Text Classification - Text Substitution”. Instead of using existing dictionaries, customized lists of polarized tokens have been established from perusing unused subjective information available in false negatives and false positives. In reviews, instances matching these polarized tokens have been replaced by a generic token either positive or negative, boosting use of subjective information.

This method has showed rather resilient. With insights limited to the training set, it has brought 10 % accuracy improvement on the validation set, i.e. almost as much as the 11 % accuracy increase on the training set.

Machine learning optimization has been conducted across ten models. eXtreme Gradient Boosting has emerged as the most performing model in this project and has boosted accuracy with 4 additional percentage points. Testing has been performed on bootstrapped accuracy distributions.

Dear Readers,

Thank you for reaching the end. Please don’t hesitate to get in touch with me through my GitHub email address or on LinkedIn. I am interested in all kinds of comments.

X. REFERENCES

Availability has been checked up on March 30, 2020.

A. Sentiment Analysis

https://www.edx.org/course/the-analytics-edge (https://www.edx.org/course/the-analytics-edge)

https://www.tidytextmining.com/sentiment.html (https://www.tidytextmining.com/sentiment.html)

https://medium.com/@annabiancajones/sentiment-analysis-of-reviews-text-pre-processing-6359343784fb (https://medium.com/@annabiancajones/sentiment-analysis-of-reviews-text-pre-processing-6359343784fb)

https://cran.r-project.org/web/packages/SentimentAnalysis/vignettes/SentimentAnalysis.html (https://cran.r-project.org/web/packages/SentimentAnalysis/vignettes/SentimentAnalysis.html)

https://monkeylearn.com/sentiment-analysis/ (https://monkeylearn.com/sentiment-analysis/)

https://towardsdatascience.com/basic-binary-sentiment-analysis-using-nltk-c94ba17ae386 (https://towardsdatascience.com/basic-binary-sentiment-analysis-using-nltk-c94ba17ae386)

https://medium.com/@datamonsters/sentiment-analysis-tools-overview-part-1-positive-and-negative-words-databases-ae35431a470c (https://medium.com/@datamonsters/sentiment-analysis-tools-overview-part-1-positive-and-negative-words-databases-ae35431a470c)

B. Text Mining

https://www.tidytextmining.com/tidytext.html (https://www.tidytextmining.com/tidytext.html)

https://monkeylearn.com/text-mining/ (https://monkeylearn.com/text-mining/)

C. About Resample Distribution of Accuracy

https://books.google.be/books?id=GgmqDwAAQBAJ&pg=PA80&lpg=PA80&dq#v=onepage&q&f=false (https://books.google.be/books?id=GgmqDwAAQBAJ&pg=PA80&lpg=PA80&dq#v=onepage&q&f=false)

https://www.edx.org/course/data-science-machine-learning (https://www.edx.org/course/data-science-machine-learning)

D. Something Simple about Overfitting

https://www.r-bloggers.com/machine-learning-explained-overfitting/ (https://www.r-bloggers.com/machine-learning-explained-overfitting/)

****

****