DAR practicum 2

Lukas Donkers: 5523265, Roel van Warmerdam: 4300556

June 24, 2016

1 Inleiding

The main concern of this assignment is: finding a model that best fits the data, and using it to predict relevance on future query-product pairs. Both of these can be done most practically in R, as it allows features to be defined easily, and has many built-in functions to create models and use those to predict values.

First, we preprocess the data with preparatory treatments, which make the creation of the model both faster and more accurate. We then apply ten features that we believe to be useful in creating such a model. Lastly, we use the polr and multinom functions in R to create the models, and use the predict function on separate data to find the accuracy of our model.

In this paper, we describe our features in detail, along with the expected results and their real outcome. We conclude this paper with a discussion about the accuracy of our model, and future improvements that could be made.

2 Data

The data used is interesting in a number of ways: First of all, the relevance data is based on an average of 3 votes. This means a lot of data is lost already, as a relevance vote of 1-2-3 is the same as a vote of 2-2-2. Next, we forego a lot of relevance data because the average is not an integer. After preprocessing, only queries where the relevance vote averaged to an integer (1, 2 or 3) are used. When inspecting relevance data, we notice that average relevance is quite rarely 1. It should not surprise us if the model does not predict 1 very often. Furthermore, a lot of queries contain numbers. This leads us to believe that numbers must be important, and a feature that uses those numbers would have a positive impact. Lastly, it is not uncommon for the queries to contain spelling errors. This means that text matching will never be a perfect indicator, and would require advanced text matching algorithms to find those matches.

3 Preparatory Treatments

We use multiple preparatory treatments to create different ways of as not all of our features can use the same data as we will describe in all subsections below. When talking about the query and description below, then it will also be applicable to the query and title.

3.1 Separating training set and test set

Because we have only one dataset, we separate the dataset into a training set and a test set. We take the first 2/3 elements of the dataset to create the training set, and the remainder for the test set. This way there is no overlap between the two, which means the resulting model should be a good fit for our dataset. However, this method does not guarantee anything regarding the accuracy on future data. It should also be noted that this way of separating the dataset is biased: if there is a significant difference between the first and latter part of the dataset, our model will not fit correctly. In that case, it would be best to take a random sample from the entire dataset as our training set, and keep the remainder as the test set.

3.2 Removing unnecessary queries and descriptions

The first thing which we did was to get rid of all of the unnecessary queries and descriptions. As was stated, the queries with the relevance score of 1, 2 or 3 are relevant, which makes every other query unnecessary and thus they can be removed. Also the table with descriptions has way more products then the amount used in the queries also making them unnecessary and thus we removed them making the processes a lot faster. In this process all of the queries were sorted by id and a list of descriptions was made to easily access the query and corresponding description.

3.3 The separation of numbers and words

Also to reduce the amount of time we decided to separating the words and numbers at the beginning instead of in every feature, because our features use either the numbers of the words but never both. This is done by taking the entire string and first selecting only numbers, and then taking the entire string and selecting everything but digits, punctuation and spaces, to make sure no other symbols are missed. Removing the punctuation also allows some abbreviations to be found as a/c can also be written as ac or a.c. and so on.

3.4 Making all words single

A important issue is that many words are written in plural in the query and single in the description or the other way around, making it quite difficult to check whether these words are the same. Our program changes the end to make all words single by changing words ending with -ies to ending them with -y, words ending with -ves with -fe and otherwise removing -(e)s. This takes care of most of the plural versions however this idea does not consider the irregular words such as goose - geese, but the amount of words which have this problem is insignificant to this problem.

3.5 Making all words lower case

Another problem is dealing with words which are sometimes written with capitals which can mess up checks to see if that word is included in both the query and description. This problem is easily solved by changing all characters to lower case.

4 Analysis

Our program uses 10 features to predict what kind of score the query should get for the found object, in which each of them were used separately. Also with in some feature we use (almost) the same functionality however instead of the titles we can use the descriptions and vice versa.

When applying stepAIC on our two models, the results are as follows:

- polr starts at an AIC of 32709.4, but does not converge due to an error
- multinom starts at an AIC of 32446.03, and converges to 16296.03

Finally, calculating the root mean square error on our two models:

- for polr, the mean square error cannot be calculated (NaN)
- for multinom, the mean square error is 0.151

4.1 Allterms

Allterms is a feature that was already suggested during the lectures. It simply finds how many terms in the query appear in the title of the found product. This amount is then divided by the total amount of terms, resulting in a value between 0 and 1.

- The coefficient of Allterms is 1.093 in polr, and 1.496 / 2.253 in multinom.
- The standard error of Allterms is 0.066 in polr, and 0.171 / 0.167 in multinom.
- The t value of Allterms is 16.466 in polr.

These values make sense - if the ratio of matching terms is higher, the rating of the query-product match should be higher as well. Of course, this is not a perfect indicator, as the product name could match the query, but could still be something the user did not intend to find. Also the t value makes it significant to the effect of the predictions and thus a good way to predict as expected.

4.2 Alltermsdesc

Alltermsdesc is the description variant of Allterms, which looks for query-terms in the description of the found product. This is, again, divided by the amount of terms, resulting in a value between 0 and 1.

- The coefficient of Alltermsdesc is 1.074 in polr, and 0.237 / 1.297 in multinom

- The standard error of Alltermsdesc is 0.064 in polr, and 0.153 / 0.150 in multinom
- The t value of Alltermsdesc is 16.850 in polr.

These values are similar to Allterms, except for the standard error, which is lower. This indicates that the description is less error prone for cases where the query matches the description, possibly because the description is often more extensive. Also the t value makes it significant to the effect of the predictions and thus a good way to predict as expected.

4.3 Allnumbers

The Allnumbers feature, checks how many of the numbers used in the query are used in the and the title of the found product similar as to the Allterms, however the Allnumbers feature also checks if a numbers is used multiple times and whether the title does contain that number multiple times assuming that people do not use a number twice if only needed once and if something has a size of 2 by 2, the query is better if it also contains two two's. This number this again is divided by all numbers in the query resulting in a value between 0 and 1.

- The coefficient of Allnumbers is 0.465 in polr, and -0.371 / 0.202 in multinom
- The standard error of Allnumbers is 0.0737 in polr, and 0.153 / 0.150 in multinom
- The t value of Allnumbers is 6.327 in polr.

These values make sense - if the ratio of numbers which have been used in the query are higher then the query will most likely, fit the title better. This can be decieving as many product can have the same numbers and be completely different. Also the t value makes it significant to the effect of the predictions and thus a good way to predict as expected.

4.4 Allnumbersdesc

Allnumbersdesc is the description variant of Allnumbers, which looks for the numbers matching in the description of the found product. And as was done with Allnumbers, this is also divided by all of the numbers in the queries.

- The coefficient of Allnumbersdesc is -0.426 in polr, and 0.538 / -0.202 in multinom
- The standard error of Allnumbersdesc is 0.080 in polr, and 0.183 / 0.178 in multinom
- The t value of Allnumbersdesc is -5.369 in polr.

These results are actually quite surprising as they are almost the opposite of the Allnumbers which is weird, considering that this is the same function only used with the description. This would mean that if less numbers which are used in the query are also used in the description that this is bad and thus a lower score while it should mean the opposite. The standard error however is almost the same which does make sense as the Allnumbers. Although it may have a weird coefficient to what we suspected, it still is significant as can be seen by looking at the t value and thus relevant for the prediction.

4.5 Allorders

Allordes is a feature which checks how many of the words in the query are in the same order as in the title of the product. This also include skips to make sure that if one word is out of order does not mean that if that word wasn't included then the order would be a lot better instead of the half at minimum. The value of Allorders is between 0 and n where n the amount of words in the query is.

- The coefficient of Allorders is 0.174 in polr, and 0.125 / 0.314 in multinom.
- The standard error of Allorders is 0.0238 in polr, and 0.0723 / 0.0709 in multinom.
- The t value of Allorders is 7.300 in polr.

This coefficient does make sense because if the order is the same the words are more likely to give a better result. What is uncertain is if this actually has to the with the order or just with the fact that if the words are in order, they are also in the Allterms (words can't be in order if they aren't even used). Also the t value makes it significant to the effect of the predictions and thus a good way to predict as expected. This feature is also relevant when looking to the t value but then again with this information only we can't tell whether it is due to the order of the words or the fact that it gets a better score for having the same words in the title as in the query.

4.6 Allordersdesc

Allordersdesc is the description variant of Allorders, which looks for words in the same order as words in the description. Allordersdesc also ranges from 0 to n where n is the amount of words in the query.

- The coefficient of Allordersdesc is 0.0773 in polr, and 0.0273 / 0.111 in multinom.
- The standard error of Allordersdesc is 0.0237 in polr, and 0.0652 / 0.0641 in multinom.
- The t value of Allordersdesc is 3.262 in polr.

This makes sense as well because the descriptions usually contains a lot more words than the title so the is a good chance that Allorderdesc gets a higher value than Allorder, because there are simply more words and more possibilities as a word may occur more than once. The standard error is close to the standard error in the Allorders which is logicol as it uses the same function and there is a large chance that words used in the title are also included in the description.This feature is also relevant when looking to the t value but then similar to Allorder, with this information only we can't tell whether it is due to the order of the words or the fact that it gets a better score for having the same words in the description as in the query.

4.7 Allabbr

The Allabbr feature checks whether there are (potential) abbreviations used in the query and the full word in the title. This is being done by firstly checking whether a word is between 2 and 4 characters long (assuming that abbreviations are no longer than 4 characters). Then for each of these potential abbreviations the algorithm checks the first character of the words to find words with similar starting characters which are the same as the abbreviation. Here it is extremely important that any punctiuation is removed because abbreviations can be written in many ways as stated in Section 3.3.

The value of Allabbr ranges from 0 to n where n is the amount of words (of 4 long to be more specific).

- The coefficient of Allabbr is -0.0241 in polr, and -0.267 / -0.261 in multinom.
- The standard error of Allabbr is 0.186 in polr, and 0.389 / 0.389 in multinom.
- The t value of Allabbr is -0.130 in polr.

The negative coefficient is quite unexpected as more abbreviations are found should increase the relevance score of a query. The reason of this might be a lot of false positives such as in. which stands for inches but will not be found, instead a other set of words starting with i and n might be found when inches should be found but isn't there, or just small words as by, causing a somewhat unsure way of checking for abbreviations. This feature also isn't significant due to the small t value, causing this feature to have a small influence the right prediction.

4.8 Allabbrdesc

Allabbrdesc is the description variant of Allabbr, here again used the assumption that the abbreviations are no longer than 4 characters and used the same way of checking if a word is a abbreviation.The value of Allabbrdesc ranges again from 0 to n where n is the amount of words (of 4 long to be more specific).

- The coefficient of Allabbrdesc is 0.0773 in polr, and 0.155 / -0.217 in multinom.
- The standard error of Allabbrdesc is 0.085 in polr, and 0.202 / 0.203 in multinom.
- The t value of Allabbrdesc is -3.813 in polr.

That this value is quite low is understandable, what is surprising though is that it is positive, while Allabbr is negative. one would suspect that Allabbrdesc is more susceptible to false positives as it usually has more words so more chance that the gets the right words in a row. We have no idea what could be the reason behind this. Even weirder is that considering the fact that this version is more susceptible to false positives but still is relevant with the t value, while the version which we expected to better isn't. this could be because the false positives still have a good effect or more full writings of abbreviations can be found in the description than in the title which does make some kind of sense.

4.9 Lengthsdesc

Lengthsdesc looks only at the amount of words in the description tied to the product found. The result is therefore an integer at or above 0. The idea behind this feature is that descriptions with more words contain more information, giving the user a better idea about the product he found.

- The coefficient of Lengthsdesc is -0.000864 in polr, and 0.00133 / 0.0000472 in multinom.
- The standard error of Lengthsdesc is 0.000187 in polr, and 0.000441 / 0.00437 in multinom.
- The t value of Lengthsdesc is -4.622 in polr.

Because descriptions are often quite long, the amount of words is often going to be a high amount. This explains the small coefficients, as they reduce the impact of those values. The interesting thing here is the difference in sign between polr and multinom. From our premise we would expect the sign to be positive, with longer descriptions being more useful. It is possible that longer descriptions may hide essential information, hence the negative sign. This is related to the next feature, which looks at the part of the description that is considered useful. The standard errors are quite low, which implies that the feature is a good indicator for classification. The t value here also indicates that the feature is relevant to the prediction.

4.10 Allreverseterms

Allreverseterms is the reverse of Alltermsdesc - rather than looking at the ratio between term matches and query terms, it looks at the ratio between term matches and description terms. As such, the value is still between 0 and 1, but becomes lower as the description becomes longer. This could be used to represent the amount of 'useful information' in the description - a lower ratio means a smaller part of the description is considered useful.

- The coefficient of Allreverseterms is -17.2 in polr, and 15.204 / -6.335 in multinom.
- The standard error of Allreverseterms is 0.00209 in polr, and 0.00493 / 0.00437 in multinom.
- The t value of Allterms is -8222.040 in polr.

Because descriptions are often quite long, the ratio is often going to be a lot closer to 0. This explains the high coefficients - small changes in the ratio still make an impact on the feature due to the larger coefficients. Here as well, the difference in sign between polr and multinom is interesting - we would expect a positive sign, because lower ratios mean the description is less useful, or that less terms match (like in Allterms). On the other hand, a longer description may be considered more useful as it contains more information, meaning that a lower ratio (divide by more terms) should imply a higher score, hence the negative sign. As you can see, this feature and the previous one are related in the sense that longer descriptions can have a positive or negative impact, depending on the amount of useful information in the description. The 'correct' sign for both coefficients is therefore hard to conclude. The standard errors are quite low here as well, which implies that the feature is a good indicator for classification. Here the t value implies that the feature is extremely relevant to the feature as it is almost certain that it will influence the prediction.

5 Conclusie

With all features combined, we create a model that can predict relevance with quite some accuracy. Out of the ten features described above, the features which had the highest impact were: Allreverseterms, Allterms and Allterms-desc. This makes sense, as the query-product match system is obviously optimized to give the best results based on text matching. Out of the two models (ordinal and multinomial), ordinal appeared to fit better, although by a small margin. We draw this conclusion based on the prediction on the test set using both models. The confusion matrices are as follows:

	Predicted 1	Predicted 2	Predicted 3
Actual 1	0	0	0
Actual 2	309	837	509
Actual 3	629	4273	4429

Table 1: POLR Confusion Matrix

	Predicted 1	Predicted 2	Predicted 3
Actual 1	0	0	0
Actual 2	178	539	287
Actual 3	760	4571	4651

Table 2: Multinom Confusion Matrix

With the test set consisting of 10986 queries, it follows that POLR has a success rate of (0+837+4429)/10986 = 47.9% And Multinom has a success rate of (0+539+4651)/10986 = 47.2% When comparing these percentages to a basic model, in which we always guess the relevance which occurs most (this would be a relevance of 2, which occurs in 5510 out of 10986 queries), we would only achieve an accuracy of 46.5%. Our model is, in that regard, not a large improvement, but an improvement nonetheless.

Furthermore, it is interesting to note that none of the models ever predict 1 for the relevance score. This can partially be attributed to the biased separation of training set and test set as discussed in section 3.1. The two tables above can be combined into a cross table, describing the overlap between predictions:

	POLR Predict 1	POLR Predict 2	POLR Predict 3
Multinom Predict 1	0	0	0
Multinom Predict 2	0	916	739
Multinom Predict 3	0	88	9243

Table 3: Cross Table of POLR and Multinom

From this table one can conclude that POLR guesses a relevance of 3 more often, but for the rest they are very similar.

The program could be improved by removing features which have little to no influence like abbreviations. Also the Allorder and Allorderdesc could be better researched if it has any influence on the program, or that it only influences the relevance because it is a stronger version of the Allterms (and Alltermsdesc respectively). Also, our program does not use any relational features which could be very beneficial to the relevance score but would also have to be researched to be definitive. Lastly, it could be import to check why the first class is never guessed, as this could improve the overall performance as more classes would be guessed lower and thus also more two guesses.