`mktinsight`: Market Insights R Package

Benjamin D Ziomek
University of Chicago Booth School of Business
bziomek@chicagobooth.edu

Abstract

Introduces `mktinsight`, or Market Insight, an R package focused on the contextual analysis of sentences for optimizing marketing email subject lines. This document first discusses the design, implementation, and goals of the package, then goes into detail around installation and usage.

1 Introduction

Despite email marketing being key to many firms' success, less than half of email marketers do any analysis of the quality of their subject lines. In large part, this is due to a lack of tooling for easily analyzing email subjects. The most popular existing email automation platforms lack built-in email optimization methods. Existing subject line optimization platforms often use dated rules gleaned from historical trends or are expensive AI platforms that require substantial manual tuning.[1] This situation is becoming problematic for marketers as the data-driven audience targeting techniques of the past ten years run out of steam. Especially with new EU privacy regulations coming on-line that limit what marketers can do using micro-targeting of audiences, optimizing marketing content is becoming ever more important to effective marketing. `mktinsight` is an initial step towards equipping today's marketers with the tools they need to apply quantitative analysis to marketing content.

[1] For example, Subjectline.com and Persado, respectively

The Market Insights R Package seeks to help marketers improve their email marketing by delivering:

1. **Segmentation** – Automatically splitting email contacts into high- and low-value groups to help optimize for responses from more senior, higher-value contacts
2. **Actionable Insights** – Extracting the topics of email subjects and providing analytics that identify the subjects that high-value readers are most interested in, and how they are expressed

Market Insights has two primary components that accomplish these tasks: An audience segmentation tool, `titlesort`, and a subject line analysis tool, `mail.subject`.

2 Implementation

The `mktinsight` package provides two functions that are useful for email marketers: `titlesort` and `mail.subject`. Together they can automate the difficult and very manual tasks of segmenting email recipients and determining what subject lines worked best to get readership, and why.

2.1 `titlesort`

`titlesort` is a function that takes a job title and outputs the probability that the job title is management-level. It relies on a Naïve Bayesian estimator and the creation of document frame matrices from job titles. `titlesort` is built on a binary classifier trained

on a proprietary dataset of nearly 12,000 job titles, labeled based on seniority (management or non-management level).

2.2 `mail.subject`

`mail.subject` is the core function in `mktinsight`. It allows marketers to analyze the success of their email subject lines without having to resort to old-fashioned sets of rules or leverage the features of massive, unwieldy marketing automation platforms. `mail.subject` contains three key analytical steps:

1. Identifying email topics
2. Identifying the topics that align with successful emails, as measured by open rates
3. Finding the words that contextualize those topics

To extract topics, the function generates dependency trees from email subjects and parses them to find the most important topic. The function then calculates the odds that an email with a certain topic is opened and read versus that with a different topic, and finds words associated with that topic to guide future email creation.

`mail.subject` also has the option of determining the level of a contact using `titlesort`, then presenting the top predictors of an email being disproportionately opened by senior rather than junior contacts, to help marketers target senior customers. All this is done in a single command aid in accelerating marketing workflows.

3 NLP Techniques

3.1 `titlesort`

At its core, `titlesort` is a binary classifier that takes a job title and outputs the probability that the job title is management-level. The function returns a vector of 1s and 0s corresponding to the predicted seniority of input job titles. A prediction of 1 corresponds to a director-level-or-higher position, and a contact with a 0 is predicted to be an individual contributor or junior manager. Based on a bag of words method, it has good accuracy most English-language job titles (see table 1), with a slight bias towards the technology industry. `titlesort` Also works reasonably well for German and French titles, but should not be used for other languages. `titlesort` is embedded inside `mail.subject` and does not need to be used separately with that function.

This function was somewhat difficult to implement mainly because job titles can vary widely, and both one- and two-word as well as very long titles are common ("Director" or "Technical Director" compared to "Vice President of Regional Software and Patent Licensing Operations, United Kingdom, Ireland, and Nordics," for example). Because of this, parsing job titles as sentences or phrases didn't work well. Instead, I took a bag of words approach: The function tokenizes the data, remove stop words and punctuation, and then create a document-feature matrix that supports further bag of words analysis.

Multiple classifiers were tested on the data, including Naïve Bayesian, Class Affinity, and WordScores models. Results are provided in the attached tables. A Naïve Bayesian model was selected for the function due to its simplicity and high performance out of the box. Even moderately tuned versions of Class Affinity and WordScores could at best match the performance of the Naïve Bayesian model.

		Predicted Class	
		Junior	Senior
Actual Class	Junior	1120	165
	Senior	47	1005

Table 1: Naïve Bayesian Confusion Matrix

		Predicted Class	
		Junior	Senior
Actual Class	Junior	167	1118
	Senior	962	90

Table 2: Class Affinity Confusion Matrix

		Predicted Class	
		Junior	Senior
Actual Class	Junior	1169	116
	Senior	144	908

Table 3: WordScores Confusion Matrix

3.2 `mail.subject`

This function evaluates the effectiveness of email subjects in attracting readers either for an entire group of contacts or based on recipients' seniority. After optionally running `titlesort` to predict contact seniority, `mail.subject` uses dependency parsing to extract the topics and tone of email subject lines. The function then analyzes them in the context of their relative success at attracting readers. It outputs a data frame of extracted topics, their effectiveness (as measured by odds ratios of the email being read if it includes that topic), and words associated with that topic, to guide future email creation.

For the first step, the function leverages spaCy, a deep learning-based dependency parser, to generate dependency trees from emails. Because many traditional topic modeling techniques rely on a lot of data, I instead use the dependency tree to identify the most foundational head noun or verb within the parse tree, ideally one that refers to a Named Entity, and tag that as the topic of the email. Because email subjects are very short, one or two words is sufficient to represent an email's topic well.

For the second step, the function trains a logistic regression model on a stripped-down bag of words consisting of the emails' topics and their open rates. A logistic regression was selected because this application prioritizes interpret-ability, speed, and portability, and the additional analytical power of tools such as neural networks and SVMs wasn't worth the additional complexity they bring.

Lastly, the model cross-indexes topics with the words that were subsidiary to them in the dependency trees of successful email subjects, select the most common words in successful emails, and presents them to the user. The output also contains with the odds of an email being opened if it contains the given words and topics, which is extracted from the embedded logistic regression.

As noted above, `mail.subject` also has the option of determining the level of a contact using `titlesort`, then presenting the top predictors of an email being disproportionately opened by senior rather than junior contacts, to help marketers target senior customers. All this is done in a single command to aid in accelerating marketing workflows.

4 Results

4.1 Output Format

Except for the trained Naïve Bayesian model embedded within `titlesort`, Market Insight is an analytics tool rather than a trained model, and as such does not contain any data except for the sample list of 12,000 job titles. That said, to demonstrate its value I have collected a list of 3,500 marketing emails from my own inbox, and analyzed them to understand what emails appealed to me. A sample output of the best topics in the sample dataset can be seen in Figure 1.

Even though the data is relatively low quality (it has a lot of unique emails, whereas in reality a marketer would send thousands of identical emails, aiding analysis) you can still see clear patterns, including my interest in exploring new opportunities prior to my MBA, responding to invitations, and joining Meetups. The main reason for the odds ratios being the same are that most of these emails had similarly small numbers of emails present, compressing the model's coefficients, which would not occur in a real use case.

The table in Figure 1 tells the user what email topics have driven the most engagement and what users are interested in. More directly, the `Odds Ratio` output gives a marketer some idea how much an email can drive additional engagement if it is on a certain topic. Finally, `mktinsight` also provides words that have been

	Root Word	Odds Ratio	Associated
223	Thursday	4645736334	NA
74	exploring	4645736332	are, Benjamin, opportunities
157	month	4645736324	a, free, Free
234	Tuesday	4645736310	NA
249	wondering	4645736307	Benjamin, doing, NA
111	Invitation	4645736302	Bonfire, Crawl, day
250	work	4645736284	before, did, where
219	Taste	4645736199	PNA, Winter, Ridge
152	Meetup	4645736198	Cider, Class
187	Request	4645736197	for, LinkedIn, NA

Figure 1: Sample output from `mail.subject`

frequently mentioned along with the key topic, giving more nuance to what works in an email subject.

The last feature of the output, which is not shown in the figure, is the ability to automatically segment emails by seniority. If job titles are specified as an input (see usage, below), the function first predicts if a contact is senior or junior, and then independently analyzes emails sent to junior and senior contacts. This allows a user to instantly see what topics appeal to senior versus junior contacts or export a full dataset to understand how to better tailor emails to reach senior customers. If `titlesort` is used in isolation, it works similarly: It takes in a character vector of job titles, and outputs a vector of 1s and 0s that are the predicted seniority of each title.

4.2 Actionable Insights

The power of this package's approach is its flexibility: Output can either be used as a single-stop email subject analysis tool or can be integrated into a broader campaign evaluation to determine what sort of messaging works best for the target audience.

This flexibility is captured in the `outputrows` argument in the `mail.subject` function. If `outputrows` is specified, the function outputs a truncated list of only the most significant email topics, how effective they are, and the other words associated with them. This means that a marketer can export a `.csv` of emails from a marketing automation platform, run one command on them in R, and receive a quantitative analysis of what's working and what's not within a few minutes. This combination of speed, simplicity and machine learning-based analysis is currently not available in a commercial product, and shows the promise of a dependency-based approach for email analysis.

For more advanced use cases, if `outputrows` is omitted, the mail.subject function outputs the entire data frame resulting from its analysis. This means that a dedicated marketer can do analysis combining raw email data with the results of `mktinsights`. For example, this would allow a marketer to examine different subsets of highly effective emails (to determine why I like emails referring to Thursday, for example, as shown in Figure 1). In this way, `mktinsights` enables the combination of machine and human decision-making, and can be used by any marketer or data scientist who has basic knowledge of R.

4.3 Looking Forward

With the introduction of the new General Data Protection Regulation (GDPR) privacy laws in the EU, marketers' toolkits are going to have to change rapidly. For the past decade, digital marketing has focused on capturing as many customers as possible using digital methods, and micro-targeting users who react well to a firm's messaging. With GDPR limiting the types of data that can be collected about EU citizens, and how long that data can be held, firms operating in Europe are going to be restricted in their use of many foundational digital marketing techniques. In addition, because of the strict penalties associated with GDPR violations, many firms are extending their compliance with the regulation to their entire global user bases.

The outcome is going to be that digital marketers will spend more time on attracting users using effective messaging rather than micro-targeting users that already want to hear what a company has to say. Unfortunately, up until recently, the tools to use analytics to hone messaging lagged those to target users. For marketers, using NLP and computer vision to improve content is going to be as critical in teh near future as using machine learning to target

users was in the past. `Mktinsights` is one small step to building out a messaging-focused analytics toolkit for marketers that can extend the use of NLP techniques to such small decisions as how to write an email subject line.

4.4 Surprises

The process of creating `mktinsights` was relatively smooth. The biggest issues I ran into in writing the code was how long it ended up being (in excess of 1000 lines), which required learning how to manage a software project in its entirety, and dealing with interconnections between different programming languages.

While R works well for statistical analysis, trying to create an application that can be used by marketers brought its limitations into stark relief. From small things like packages not talking to each other in compatible formats, to larger issues like there not being a dependency parser that exists natively in R, it has become clear that to expand on this project and continue my study, Python is required.

All in all, the realization that I need to keep learning is the biggest thing I got out of this project. To create `mktinsight` I needed to learn how to write software rather than just scripts, think about my architecture, and learn how to refactor functions when things changed (as they always do). I also had to learn how to use Git, GitHub, and even LaTeXfor this write-up. The biggest surprise in building this package wasn't really a surprise: You always need to keep learning!

5 Using `mktinsight`

5.1 Preparing for Installation

More detailed instructions for installation can be found on the project's GitHub page[2]. For a brief explanation see below.[3]

First, ensure you have Python and R installed on your system and python is on your system path. Then, in a command prompt with sudo or admin rights, install the python dependencies:

```
pip install -U spacy
python -m spacy download en
```

After they are successfully installed, you can either install the package from a tarball, or you can install it from GitHub directly. The latter is suggested so you get the most up-to-date version of the package.

5.2 Installing from GitHub

Run the commands:

```
install.packages("devtools")
devtools::install_github("quanteda/
spacyr")
devtools::install_github("bziomek/
mktinsight")
```

5.3 Installing from a Tarball

Open R and install the R dependencies and the Market Insights package using the below commands. Substitute `path_to_file` for the actual path to the relevant tarball:

```
install.packages("quanteda")
devtools::install_github("quanteda/
spacyr")
install.packages(path_to_file/
mktinsight_0.2.1.tar.gz, repos = NULL,
type="source")
```

Now you're ready to analyze emails!

5.4 Usage Overview

To access the below and more documentation from within R, run the command:

```
package?mktinsight
```

[2] https://github.com/bziomek/mktinsight

[3] If copying and pasting commands from this document, be sure to remove any spacing added by your PDF reader.

5.4.1 Syntax

The core command provided by Market Insights is the R function `mail.subject`. Its syntax is:

```
  mail.subject(subjects, readflag,
jobtitles = NULL, minmails = 2,
outputrows = NULL)
```

Where:

- `subjects` in a vector of strings that should be email subjects
- `readflag` is a vector of 1s and 0s representing if an email has been read or not
- `jobtitles` is a vector of strings that is the job title of the email recipient (optional)
- `minmails` is an integer corresponding to the minimum instances of a subject line that will be included in the analysis (optional, default 2)
- `outputrows` is the number of rows of email topics that should be output (optional, defaults to outputting the entire data frame of results)

5.4.2 Output Detail

The output of `mail.subject` is a data frame with the format shown in Figure 1 section, which contains the columns:

1. The ID of each root word, assigned in the order they are identified
2. `Root Word` - Extracted email topics
3. `Odds Ratio` - The estimated odds ratio of how much more likely an email with a subject containing that topic is to be opened versus another email, resulting from logistic regression
4. `Associated` - Words associated with each topic to guide email creation

The default output is the entire data frame of extracted email topics, odds ratios, and associated words. If `jobtitles` is specified, the function will first run a model that predicts which contacts are senior (director or higher) and which are junior. It will then evaluate topic effectiveness separately for each group. If `outputrows` is specified, the function outputs the most effective topics (those with the highest odds ratios). The number of rows output for each of junior and senior contacts is equivalent to the number of output rows specified.

5.5 Troubleshooting

5.5.1 Installation

If either installation from GitHub or source fails, it is likely because R is bad at locating dependencies when a package is installed from an unofficial repository. This is usually solved by the code given in "Installation," above, but if not, try the following commands:

```
  remove.packages("mktinsights")
  remove.packages("spacyr")
  remove.packages(''quanteda'')
  devtools::install_github("quanteda/
spacyr")
  devtools::install_github("bziomek/
mktinsight")
```

If you get errors about another package not being found, run:

```
  install.packages(''package_name'')
```

This should solve any installation issues.

5.5.2 Usage

If you're having issues running `mail.subject` the issue likely lies in its python dependencies.

If R cannot find spacy, ensure that spacy is installed in your base python environment. This may require re-running the pip commands from the spacy website.

If R can find spacy, but won't boot, ensure that spacy is installed using pip, not conda.

Anaconda does not work well with spacy and is not recommended for this application.

References

[HubSpot(2018)] HubSpot. The ultimate list of marketing statistics for 2018, 2018. URL https://www.hubspot.com/marketing-statistics.

[jsowyrda(2017)] jsowyrda. Email subject performance and analysis, 2017. URL https://community.hubspot.com/t5/Email-Marketing/Email-subject-performance-and-analysis/td-p/34239.

[Perry and Benoit(2017)] Patrick O. Perry and Kenneth Benoit. Scaling text with the class affinity model. *arXiv:1710.08963 [stat.ML]*, 2017. URL https://arxiv.org/abs/1710.08963.

[Manning et al.(2008)Manning, Raghavan, and Schütze] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[Jurafsky and Martin(2017)] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 2017.

[Laver et al.(2003)Laver, Benoit, and Garry] Michael Laver, Kenneth R. Benoit, and John Garry. Extracting policy positions from political texts using words as data. *American Political Science Review*, 2003.

[Beauchamp(2012)] N. Beauchamp. Using text to scale legislatures with uninformative voting. 2012.

[Nenadic and Greenacre(2007)] O. Nenadic and M. Greenacre. Correspondence analysis in r, with two- and three-dimensional graphics: The ca package. *Journal of Statistical Software*, 2007. URL http://www.jstatsoft.org/v20/i03/.

[Benoit(2018)] Kenneth Benoit. Quenteda reference, 2018. URL http://docs.quanteda.io/reference/index.html.

[Explosion.AI(2018)] Explosion.AI. spacy usage, 2018. URL https://spacy.io/usage/.

[gdp(2018)] 2018 reform of eu data protection rules, 2018. URL https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en.