Microsoft weeds out fake marketing leads with Naïve Bayes and Machine Learning Server

Companies increase operational efficiency through digital transformation. At Microsoft, we're digitally transforming our marketing and sales operations for greater efficiency and innovation. Core Services Engineering (CSE) helps create, support, and refine the services that marketing and sales teams use to nurture and track new sales leads.

To generate global demand for our products and services, our marketing and sales organization collects leads when people request information or access gated content with an online form. Our system collects leads via marketing vehicles such as:

- Trial products, services, and subscriptions such as Microsoft Azure and Microsoft Office.
- Events like conferences, webinars, and training sessions.
- Content downloads.

When people trial a product, send email, or download content, they become leads. These names—which can be a company or a person—become records in our marketing automation system. Then our marketers and sellers follow up with the leads and turn them into customers by selling them a product, service, or subscription.

However, when people sign up via online forms, they sometimes give a fake name, company name, email, or phone number. They may submit randomly typed characters (keyboard gibberish) or use profanity. Or, they may accidentally make a small typographical error, but otherwise the name is real—so we don't want to classify the lead as junk. The abundance of fake lead names across Microsoft subsidiaries results in:

- **Lost productivity for our global marketers and sellers.** Fake names waste an enormous amount of time since sellers rely on accurate information to follow up with leads.
- **Lost revenue opportunities.** Among thousands of fake lead names, there could be one legitimate opportunity.

Improving our data quality with machine learning

Each day, thousands of people sign up using thousands of web forms. But, in any month, many of the lead names—whether a company or a person—are fake. Improving data quality is critical. To do that, and to determine if names are real or fake, we built a machine learning solution that uses:

- Microsoft Machine Learning Server (previously Microsoft R Server).
- A data quality service that integrates machine learning models. When a company name enters the marketing system, the system calls our data quality service, which immediately checks if it's a fake name.

So far, machine learning has reduced the number of fake company names that enter our marketing system, at scale. Our solution has prevented thousands of names from being routed to marketers and sellers. Filtering out junk leads has made our marketing and sales teams more efficient, allowing them to focus on real leads and help customers better.

Operationalizing our model, at scale, was key

We needed a scalable way to eliminate fake names across millions of records and to build and *operationalize* our machine learning model—in other words, we wanted a systematic, automated approach with measurable gains. We chose Machine Learning Server, in part, because:

- It can handle our large datasets—which enables us to train and score our model.

- It has the computing power that we need.
- We can control how we scale our model and operationalize for high-volume business requests.
- Access is based on user name and password, which are securely stored in Azure Key Vault.
- It helps expose the model as a secure API that can be integrated with other systems and improved separately.

Moving from a rules-based model to machine learning

For a system to make automated decisions—such as identifying and weeding out fake names—it's common to create models based on:

- **Rules.** Experts create static rules to cover common scenarios. As new scenarios occur, new rules are written.
- **Machine learning.** Algorithms are used to train the model and make intelligent predictions.

A static, rules-based model can make it hard to capture varying types of keyboard gibberish (like akljfalkdjg). With static rules, our marketers must waste time sorting through the fake leads and deciphering misleading or confusing information. In contrast, machine learning algorithms help us build and train our model by labeling and classifying data at the beginning of the process. Then, as data enters the model, the algorithm categorizes the data correctly—saving our sellers valuable time.

Realizing how machine learning could help, we used the Naive Bayes classifier algorithm to categorize names as real/fake. Our algorithm is influenced by how LinkedIn detects spam names in their social networks.

Scenarios—How we use the information that the model gives us

Our business team identified the Microsoft subsidiaries worldwide that are most affected by fake names. Now, we're weeding out fake names so that marketers and sellers don't have to. Going forward, we plan to:

- **Create a lead data quality metric with more lead-related signals and other machine learning models that allow us to stack-rank our leads.** The goal is to give a list to our sellers and marketers that suggests which leads to call first and which to call next.
- **Make contact information visible to our sellers and marketers when they're talking on the phone with leads.** For example, if the phone number that someone gave in an online form is real, but the company name isn't, our seller can ask the lead to confirm the company name.

Choosing our technology and designing our approach

After we opted to use a machine learning model, we incorporated the following technologies into our solution:

- The programming language R and the Naive Bayes classifier algorithm for training and building our model are based, in part, on the approach that LinkedIn uses.
- Machine Learning Server with machine learning, R, and artificial intelligence (AI) capabilities help us build and operationalize our model.
- Our data quality service, which integrates with the machine learning models to determine if a name is fake.

While our approach has focused on fake company names, it is similar for fake person names. Figure 1 shows the architecture we built our solution upon.

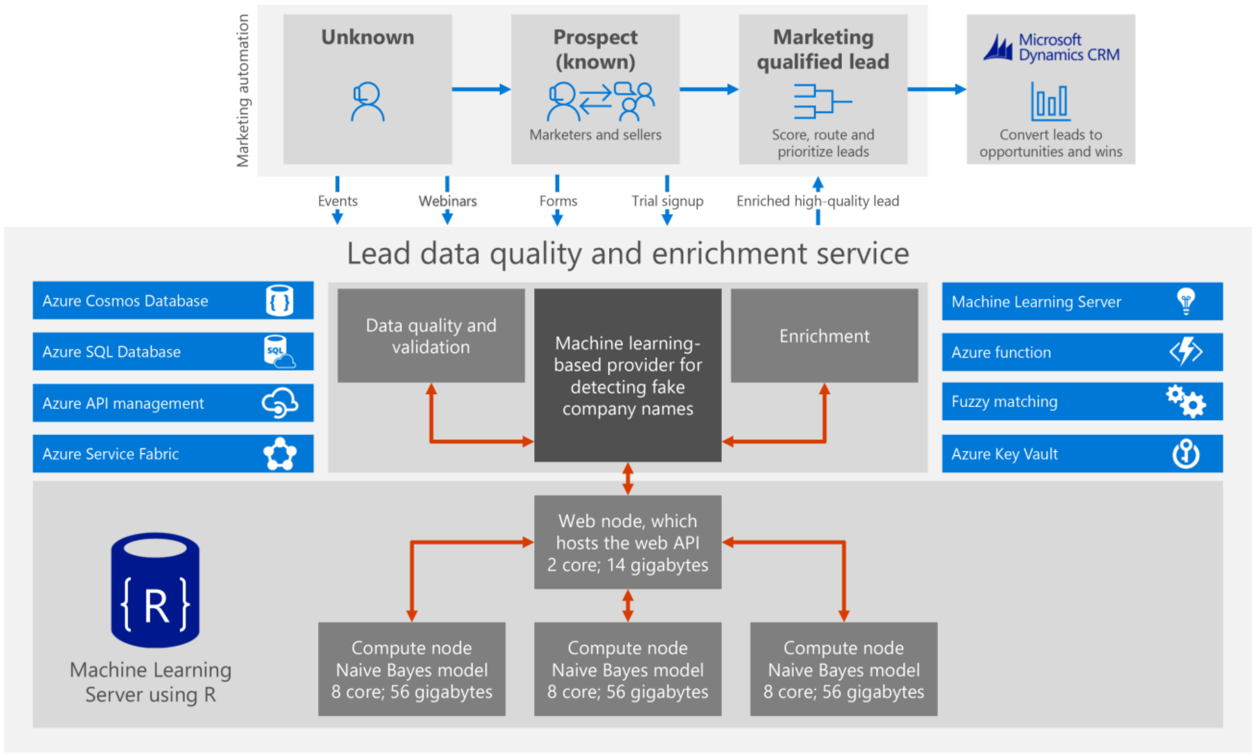
Marketing automation
Unknown
Prospect (known)
Marketers and sellers
Marketing qualified lead
Score, route and prioritize leads
Microsoft Dynamics CRM
Convert leads to opportunities and wins
Events
Webinars
Forms
Trial signup
Enriched high-quality lead
Lead data quality and enrichment service
Azure Cosmos Database
Azure SQL Database
Azure API management
Azure Service Fabric
Data quality and validation
Machine learning-based provider for detecting fake company names
Enrichment
Machine Learning Server
Azure function
Fuzzy matching
Azure Key Vault
Web node, which hosts the web API 2 core; 14 gigabytes
{R}
Machine Learning Server using R
Compute node Naive Bayes model 8 core; 56 gigabytes
Compute node Naive Bayes model 8 core; 56 gigabytes
Compute node Naive Bayes model 8 core; 56 gigabytes

Figure 1. Technical architecture for our data quality service

We designed our overall architecture and process to work as follows:

1. Marketing leads enter our data quality and enrichment service, where our team does fake-name detection, data matching, validation, and enrichment. We combine these data activities using a 590-megabyte model. Our training data consists of about 1.5 million real company names and about 208,312 fake (profanity and gibberish) company names. Before we train the model, we remove commonly used company suffixes such as Private, Ltd., and Inc.
2. We generate *n-grams*—combinations of contiguous letters—of three to seven characters, and calculate probabilities that each n-gram belongs to the real/fake name dataset in the model. For example, an n-gram that shows three sequenced letters of the name “Microsoft” would look like “Mic,” “icr” “cro” and so on. The training process computes how often the n-grams occur in real/fake company names and stores the computation in the model.
3. We have four virtual machines that run Machine Learning Server. One serves as a web node and three serve as compute nodes. We have more compute nodes so that we can scale to handle the volume of requests that we have. The architecture gives us the ability to scale up or down by adding/removing compute nodes as needed based on the volume of requests. The provider calls a web API hosted on the web node, with company name as input.
4. The web API calls the scoring function on the compute node. This scoring function generates n-grams from the input company name and calculates the frequencies of these n-grams in the real/fake training dataset.
5. To determine whether the input company name is real or fake, the predict function in R uses these calculated n-gram frequencies stored in the model, along with the Naive Bayes rule.

To summarize, the scoring function that’s used during prediction generates the n-grams. It uses the frequencies of each n-gram in the real/fake name dataset that’s stored in the model to compute the probability of the company name belonging to the real/fake name dataset. Then, it uses these computed probabilities to determine if the company name is fake.

Business, technical, and design considerations, based on what we learned

If your organization has similar business needs, consider that:

- **Ideally, the business problem should be solved within your organization itself rather than outsourcing it.** Your organization will have deeper historical knowledge of the business domain, which helps to design the most relevant solution.
- **Having good training and test data is crucial.** Most of the work we did was labeling our test data, analyzing how Naive Bayes performed compared to rxLogisticRegression and rxFastTrees algorithms, determining how accurate our model was, and updating our model where needed.
- **When you design a machine learning model, it's important to identify how to effectively label the raw data.** Unlabeled data has no information to explain or categorize it. We label the names as fake/real and apply the machine learning model. This model takes new, unlabeled data and predicts a likely label for it.
- **Even in machine learning, you risk having false positives and negatives, so you need to keep analyzing predictions and retraining the model.** Crowdsourcing is an effective way to analyze whether the predictions from the model are correct; otherwise, these can be time-consuming tasks. In our case, due to certain constraints we faced, we didn't use crowdsourcing, but plan to do so in the future.
- **Adding n-grams for profanity in the training dataset leads to significant false positives.** For training the model, we recommend using n-grams for gibberish data only. Use the entire word—rather than n-grams—for profanity. To reduce false negatives, you could use name entity recognition.

Operationalizing with Machine Learning Server vs. other Microsoft technologies

Some other technical and design considerations included deciding which Microsoft technologies to use for creating machine learning models. Microsoft offers great options such as Machine Learning Server, SQL Server 2017 Machine Learning Services (previously SQL Server 2016 R Services), and Azure Machine Learning Studio. Here are some tips to help you decide which to use for creating and operationalizing your model:

- If you don't depend on SQL Server for your model, Machine Learning Server is a great option. You can use the libraries in R and Python to build the model, and you can easily operationalize R and Python models. This option allows you to scale out as needed and lets you control the version of R packages that you want to use for modeling.
- If you have training data in SQL Server and want to build a model that's close to your training data, SQL Server 2017 Machine Learning Services works well—but there are dependencies on SQL Server and limits on model size.
- If your model is simple, you could build it in SQL Server as a stored procedure without using libraries. This option works well for simpler models that aren't hard to code. You can get good accuracy and use fewer resources, which saves money.
- If you're doing experiments and want quick learning, Azure Machine Learning Studio is a great choice. As your training dataset grows and you want to scale your models for high-volume requests, consider Machine Learning Server and SQL Server 2017 Machine Learning Services.

Challenges

Some of the roadblocks we've faced include:

- **Having good training data.** High-quality training data begins with a collection of company names that are clearly classified as real or fake—ideally, from companies around the world. We feed that information into our model for it to start learning the patterns of real or fake company names. It takes a while to build and refine this data, and it's an iterative process.

- **Identifying and manually labeling our training and test dataset.** We manually labeled thousands of records as real or fake, which takes a lot of time and effort. Instead, take advantage of crowdsourcing services if you can, so that you avoid manual labeling. With these services, you submit company names through a secure API and a human says if the company name is real or fake.
- **Deciding which product to use for operationalizing our model.** We tried different technologies, but found computing limitations and versioning dependencies between the R Naive Bayes package we used and what was available in Azure Machine Learning Studio at the time. We chose Machine Learning Server because it addressed those issues, had the computing power we needed, and helped us easily scale out our model.
- **Configuring load balance.** If our Machine Learning Server web node gets lots of requests, it randomly chooses which of the three compute nodes to send the request to. This can result in one node that's overutilized while another is underutilized. We like to use a round-robin approach, where all nodes are used equally to better distribute the load. This can be achieved by using an Azure load balancer in between the web and compute node.

Measurable benefits we've seen so far

The gains we've made thus far are just the beginning. So far, Machine Learning Server has helped us in the following ways:

- With the machine learning model, our system tags about 5 to 9 percent more fake records than the static model. This means the system prevented 5 to 9 percent more fake names from going to marketers and sellers. Over time, this represents a vast number of fake names that our sellers do not have to sort through. As a result, marketer and seller productivity is enhanced.
- We've captured more gibberish data and most profanities, with fewer fake positives and fake negatives. We have a high degree of accuracy, with an error rate of +/– 0.2 percent.
- Our time to respond to requests has improved. With 10,000 data classifications of real/fake in 16 minutes and 200,000 classifications in 3 hours, 13 minutes, we've ensured that our data quality service meets service level agreements for performance and response time. We plan to improve response time by slightly modifying the algorithm in Python.

Next steps

We're excited about how our digital transformation journey has already enabled us to innovate and be more efficient. We'll build on this momentum by learning more about business needs and delivering other machine learning solutions. Our roadmap includes:

- Ensuring that our machine learning model delivers value end-to-end. Machine learning is just one link in the chain that reaches all the way to sellers and marketers around the world. The whole chain needs to work well.
- Expanding our set of models and making business processes and lead quality more AI-driven vs. rule-driven.
- Operationalizing other machine learning models, so that we get a holistic view of a lead.
- Addressing issues created from sites that create fake registrations.

By improving data quality at scale, we're enabling marketers and sellers to focus on customers and to sell our products, services, and subscriptions more efficiently.

For more information

Microsoft IT Showcase

microsoft.com/itshowcase