# Text Classification

## A Comprehensive Guide to Classifying Text with Machine Learning

Text classification is the process of assigning tags or categories to text according to its content. It's one of the fundamental tasks in Natural Language Processing (https://monkeylearn.com/blog/definitive-guide-natural-language-processing/) (NLP) with broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection.

Unstructured data in the form of text is everywhere: emails, chats, web pages, social media, support tickets, survey responses, and more. Text can be an extremely rich source of information, but extracting insights from it can be hard

and time-consuming due to its unstructured nature. Businesses are turning to text classification for structuring text in a fast and cost-efficient way to enhance decision-making and automate processes.

But, what is text classification? How does text classification work? What are the algorithms used for classifying text? What are the most common business applications?

We'll try to answer those questions in this guide. Some of the topics you'll find on this guide include the following:

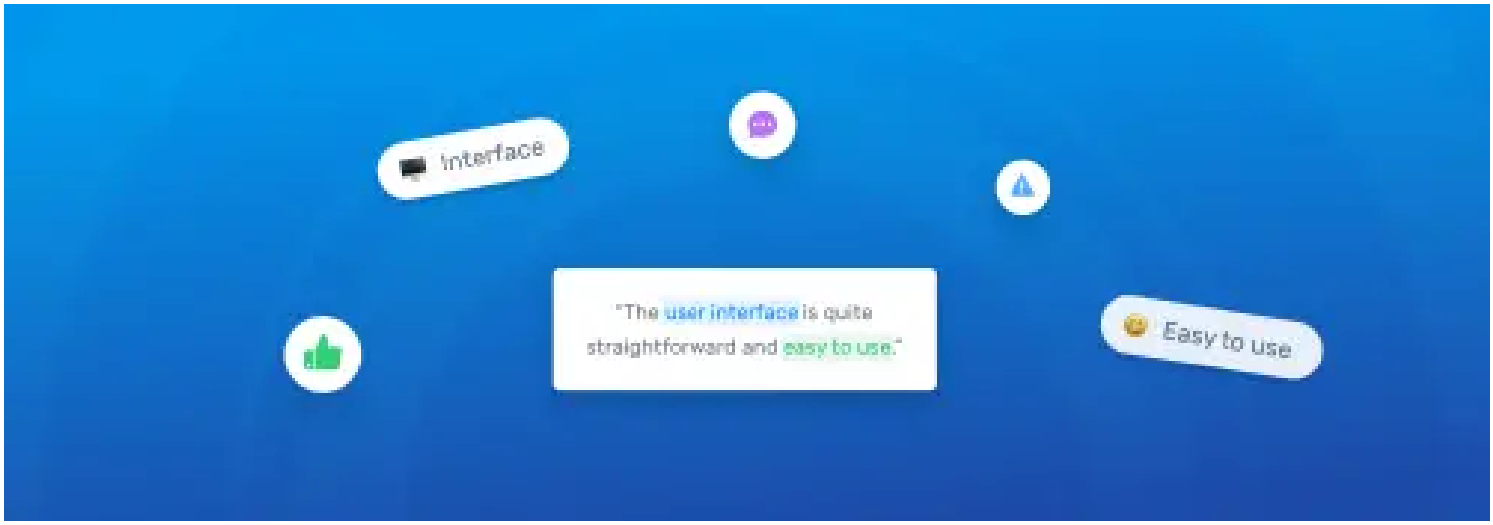
1. **What is Text Classification?**
2. **How Does Text Classification Work?**
   - Rule-based systems
   - Machine Learning based systems
   - Hybrid systems
   - Metrics and evaluation
   - Why is Text Classification Important?
3. **Text Classification Applications**
   - Examples
   - Use cases
     - Social media monitoring
     - Brand monitoring
     - Customer service
     - Voice of customer
4. **Resources**
   - Datasets
   - Tools

- Open source libraries
- SaaS APIs
- Tutorial

What is Text Classification? How does it work? Applications Resources

Let's dive in!

# What is Text Classification? (https://monkeylearn.com/what-is-text-classification)

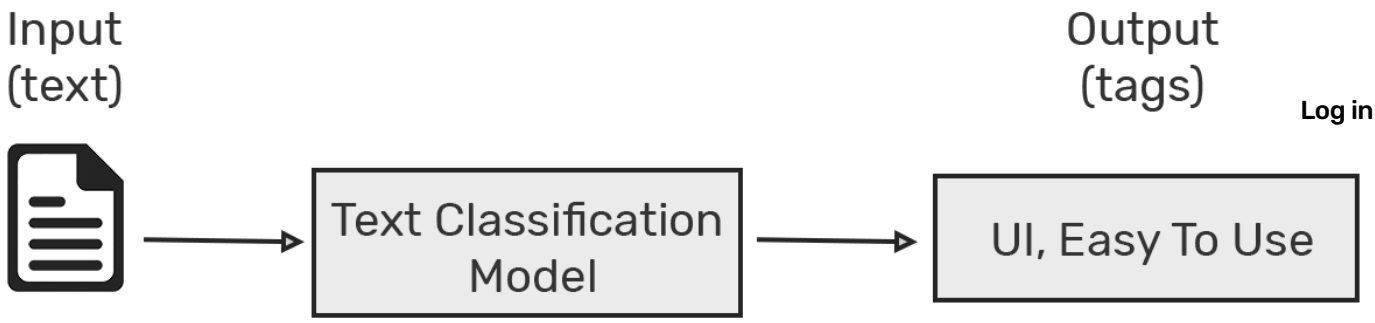



Text classification (a.k.a. *text categorization* or *text tagging*) is the task of assigning a set of predefined categories to free-text. Text classifiers can be used to organize, structure, and categorize pretty much anything. For example, new articles can be organized by topics, support tickets can be organized by urgency, chat conversations can be organized by language, brand mentions can be organized by sentiment, and so on.

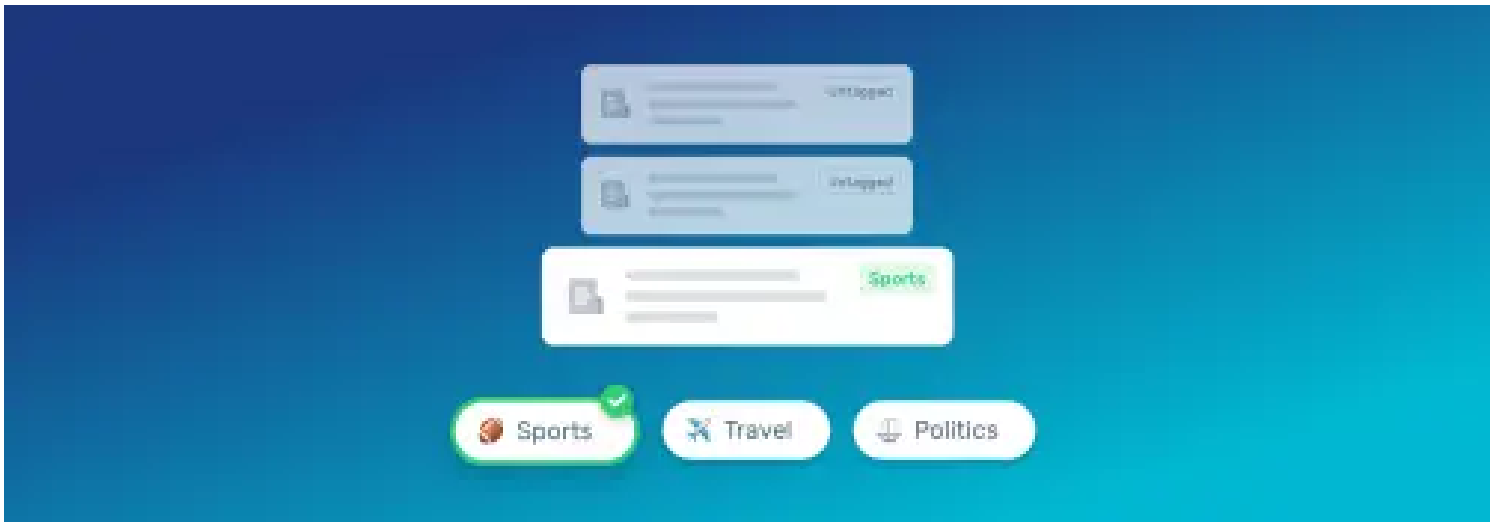
As an example, take a look at the following text below:

*"The user interface is quite straightforward and easy to use."*

A classifier can take this text as an input, analyze its content, and then and automatically assign relevant tags, such as *UI* and *Easy To Use* that represent this text:




# How Does Text Classification Work?




Text classification can be done in two different ways: manual and automatic classification. In the former, a human annotator interprets the content of text and categorizes it accordingly. This method usually can provide quality results but it's time-consuming and expensive. The latter applies machine learning (https://monkeylearn.com/blog/gentle-guide-to-machine-learning/), natural language processing, and other techniques to automatically classify text in a faster and more cost-effective way.

There are many approaches to automatic text classification, which can be grouped into three different types of systems:

- Rule-based systems
- Machine Learning based systems
- Hybrid systems

## Rule-based Systems

Rule-based approaches classify text into organized groups by using a set of handcrafted linguistic rules. These rules instruct the system to use semantically relevant elements of a text to identify relevant categories based on its content. Each rule consists of an antecedent or pattern and a predicted category.

Say that you want to classify news articles into 2 groups, namely, *Sports* and *Politics*. First, you'll need to define two lists of words that characterize each group (e.g. words related to sports such as *football*, *basketball*, *LeBron James*, etc., and words related to politics such as *Donald Trump*, *Hillary Clinton*, *Putin*, etc.). Next, when you want to classify a new incoming text, you'll need to count the number of sport-related words that appear in the text and do the same for politics-related words. If the number of sport-related word appearances is greater than the number of politics-related word count, then the text is classified as *sports* and vice versa.

For example, this rule-based system will classify the headline *"When is LeBron James' first game with the Lakers?"* as *Sports* because it counted 1 sport-related term (*Lebron James*) and it didn't count any politics-related terms.

Rule-based systems are human comprehensible and can be improved over time. But this approach has some disadvantages. For starters, these systems require deep knowledge of the domain. They are also time-consuming, since generating

rules for a complex system can be quite challenging and usually requires a lot of analysis and testing. Rule-based systems are also difficult to maintain and don't scale well given that adding new rules can affect the results of the pre-existing rules.

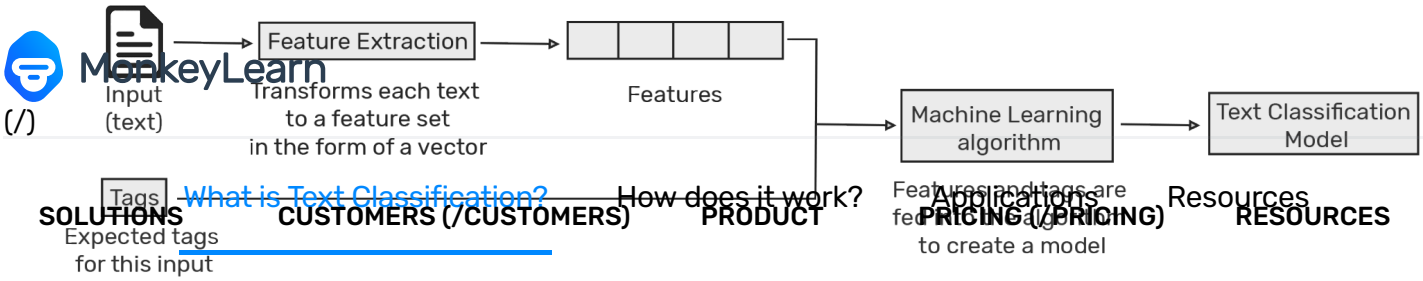
## Machine Learning Based Systems

Instead of relying on manually crafted rules, text classification with machine learning learns to make classifications based on past observations. By using pre-labeled examples as training data, a machine learning algorithm can learn the different associations between pieces of text and that a particular output (i.e. tags) is expected for a particular input (i.e. text).

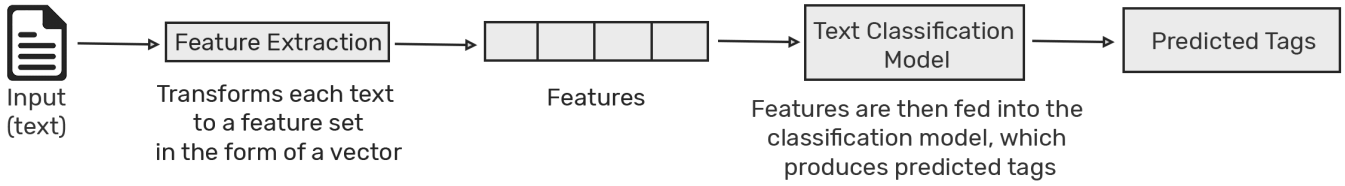
The first step towards training a classifier with machine learning is feature extraction: a method is used to transform each text into a numerical representation (https://monkeylearn.com/blog/beginners-guide-text-vectorization/) in the form of a vector. One of the most frequently used approaches is bag of words (https://machinelearningmastery.com/gentle-introduction-bag-words-model/), where a vector represents the frequency of a word in a predefined dictionary of words.

For example, if we have defined our dictionary to have the following words {This, is, the, not, awesome, bad, basketball}, and we wanted to vectorize the text *"This is awesome"*, we would have the following vector representation of that text: (1, 1, 0, 0, 1, 0, 0).

Then, the machine learning algorithm is fed with training data that consists of pairs of feature sets (vectors for each text example) and tags (e.g. *sports*, *politics*) to produce a classification model:




Once it's trained with enough training samples, the machine learning model can begin to make accurate predictions. The same feature extractor is used to transform unseen text to feature sets which can be fed into the classification model to get predictions on tags (e.g. *sports*, *politics*):




Text classification with machine learning is usually much more accurate than human-crafted rule systems, especially on complex classification tasks. Also, classifiers with machine learning are easier to maintain and you can always tag new examples to learn new tasks.

## Text Classification Algorithms

Some of the most popular machine learning algorithms for creating text classification models include the naive bayes family of algorithms, support vector machines, and deep learning.

### Naive Bayes

Naive Bayes (https://monkeylearn.com/text-classification-naive-bayes/) is a family of statistical algorithms we can make use of when doing text classification. One of the members of that family is Multinomial Naive Bayes (MNB). One of its main

advantages is that you can get really good results when data available is not much (~ a couple of thousand tagged samples) and computational resources are scarce.

All you need to know is that Naive Bayes is based on Bayes' Theorem, which helps us compute the conditional probabilities of occurrence of two events based on the probabilities of occurrence of each individual event. This means that any vector that represents a text will have to contain information about the probabilities of appearance of the words of the text within the texts of a given category so that the algorithm can compute the likelihood of that text's belonging to the category.

**Support Vector Machines**

Support Vector Machines (https://monkeylearn.com/text-classification-support-vector-machines-svm/) (SVM) is just one out of many algorithms we can choose from when doing text classification. Like naive bayes, SVM doesn't need much training data to start providing accurate results. Although it needs more computational resources than Naive Bayes, SVM can achieve more accurate results.

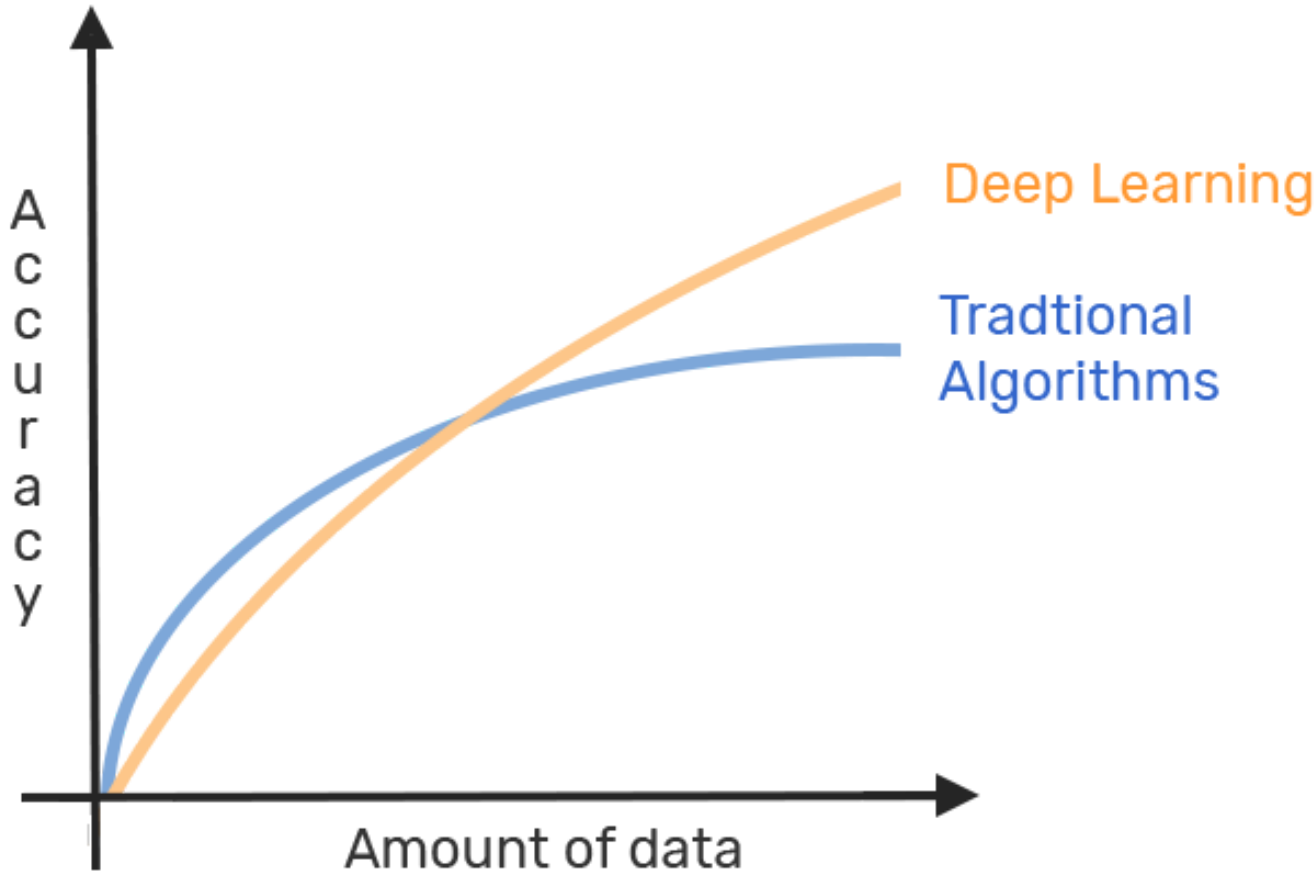
In short, SVM takes care of drawing a "line" or hyperplane that divides a space into two subspaces: one subspace that contains vectors that belong to a group and another subspace that contains vectors that do not belong to that group. Those vectors are representations of your training texts and a group is a tag you have tagged your texts with.

**Deep Learning**

Deep learning (https://medium.com/dair-ai/deep-learning-for-nlp-an-overview-of-recent-trends-d0d8f40a776d) is a set of algorithms and techniques inspired by how the human brain works. Text classification has benefited from the recent

resurgence of deep learning architectures due to their potential to reach high accuracy with less need of engineered features. The two main deep learning architectures used in text classification are Convolutional Neural Networks (https://machinelearningmastery.com/crash-course-convolutional-neural-networks/) (CNN) and Recurrent Neural Networks (https://machinelearningmastery.com/crash-course-recurrent-neural-networks-deep-learning/) (RNN).

On the one hand, deep learning algorithms require much more training data than traditional machine learning algorithms, i.e. at least millions of tagged examples. On the other hand, traditional machine learning algorithms such as SVM and NB reach a certain threshold where adding more training data doesn't improve their accuracy. In contrast, deep learning classifiers continue to get better the more data you feed them with:



Deep learning algorithms such as Word2Vec (https://code.google.com/archive/p/word2vec/) or GloVe (https://nlp.stanford.edu/projects/glove/) are also used in order to obtain better vector representations for words and improve the accuracy of classifiers trained with traditional machine learning algorithms.

## Hybrid Systems

Hybrids systems combine a base classifier trained with machine learning and a rule-based system, which is used to further improve the results. These hybrid systems can be easily fine-tuned by adding specific rules for those conflicting tags that haven't been correctly modeled by the base classifier.

## Metrics and Evaluation

Cross-validation (https://en.wikipedia.org/wiki/Cross-validation_(statistics)) is a common method to evaluate the performance of a text classifier. It consists in splitting the training dataset randomly into equal-length sets of examples (e.g. 4 sets with 25% of the data). For each set, a text classifier is trained with the remaining samples (e.g. 75% of the samples). Next, the classifiers make predictions on their respective sets and the results are compared against the human-annotated tags. This allows finding when a prediction was right (true positives and true negatives) and when it made a mistake (false positives, false negatives).

With these results, you can build performance metrics that are useful for a quick assessment on how well a classifier works:

- **Accuracy:** the percentage of texts that were predicted with the correct tag.
- **Precision:** the percentage of examples the classifier got right out of the total number of examples that it predicted for a given tag.

- **Recall:** the percentage of examples the classifier predicted for a given tag out of the total number of examples it should have predicted for that given tag.
- **F1 Score:** the harmonic mean of precision and recall.

## Why is Text Classification Important?

According to IBM (https://www.ibm.com/blogs/watson/2016/05/biggest-data-challenges-might-not-even-know/), it is estimated that around 80% of all information is unstructured, with text being one of the most common types of unstructured data. Because of the messy nature of text, analyzing, understanding, organizing, and sorting through text data is hard and time-consuming so most companies fail to extract value from that.

This is where text classification with machine learning steps in. By using text classifiers, companies can structure business information such as email, legal documents, web pages, chat conversations, and social media messages in a fast and cost-effective way. This allows companies to save time when analyzing text data, help inform business decisions, and automate business processes.

Some of the reasons why companies are leveraging text classification with machine learning are the following:

- Scalability

Manually analyzing and organizing text takes time. It's a slow process where a human needs to read each text and decide how to structure it. Machine learning changes this and enables to easily analyze millions of texts at a fraction of a cost.

- Real-time analysis

There are critical situations that companies need to identify as soon as possible and take immediate action (e.g. PR crises on social media). Text classifiers with machine learning can make accurate precisions in real-time that enable companies to identify critical information instantly and take action right away.

- Consistent criteria

Human annotators make mistakes when classifying text data due to distractions, fatigue, and boredom. Other errors are generated due to inconsistent criteria. In contrast, machine learning applies the same lens and criteria to all of the data, thus allowing humans to reduce errors with centralized text classification models.

# Text Classification Applications

# Examples (https://monkeylearn.com/text-classification-examples)

Text classification can be used in a broad range of contexts such as classifying short texts (https://monkeylearn.com/short-text-classification/) (e.g. as tweets, headlines or tweets) or organizing much larger documents

(https://monkeylearn.com/document-classification-vs-sentence-classification/) (e.g. customer reviews, media articles or legal contracts). Some of the most well-known examples of text classification include sentiment analysis, topic labeling, language detection, and intent detection.

## Sentiment Analysis

Probably the most common example of text classification is sentiment analysis (https://monkeylearn.com/sentiment-analysis/): the automated process of determining whether a text is positive, negative, or neutral. Companies are using sentiment classifiers on a wide range of applications (https://monkeylearn.com/sentiment-analysis/#sentiment-analysis-use-cases-and-applications), such as product analytics, brand monitoring, customer support, market research, workforce analytics, and much more.

This is a pre-trained classifier (https://app.monkeylearn.com/main/classifiers/cl_pi3C7JiL/) using MonkeyLearn for classifying text in English according to their sentiment. Feel free to experiment and try different expressions to see the classifier makes the predictions:

### Test with your own text

This is a great tool!

Classify Text

### Results

| TAG | CONFIDENCE |
| --- | --- |
| Positive | 99.8% |

If you see an odd result, don't worry, it's probably because it hasn't been trained (yet) with similar expressions. As an alternative, you can build a custom classifier (http://help.monkeylearn.com/text-classification/custom-classifiers/how-to-build-a-custom-classifier) for sentiment analysis and get more appropriate results for your data and criteria.

## Topic Labeling

Another common example of text classification is topic labeling, that is, understanding what a given text is talking about. It's often used for structuring and organizing data such as organizing customer feedback by its topic or organizing news articles according to their subject.

The following is a pre-trained model for classifying NPS responses for SaaS products (https://app.monkeylearn.com/main/classifiers/cl_sGdE8hD9/) according to their topic. It can tag customer feedback in categories such as *Customer Support*, *Ease of Use*, *Features*, and *Pricing*:

### Test with your own text

I don't completely understand the pricing. But it's a trivial issue compared to value received.

Classify Text

### Results

| TAG | CONFIDENCE |
| --- | --- |
| Pricing | 85.1% |

# Language Detection

What is Text Classification? How does it work? Applications Resources

Language detection is another great example of text classification, that is the process of classifying incoming text according to its language. These text classifiers are often used for routing purposes (e.g. route support tickets according to their language to the appropriate team).

The following is a classifier trained for detecting 49 different languages (https://app.monkeylearn.com/main/classifiers/cl_Vay9jh28/) in text:

## Test with your own text

The scientific method is a body of techniques for investigating phenomena, acquiring new knowledge, or correcting and integrating previous knowledge.

Classify Text

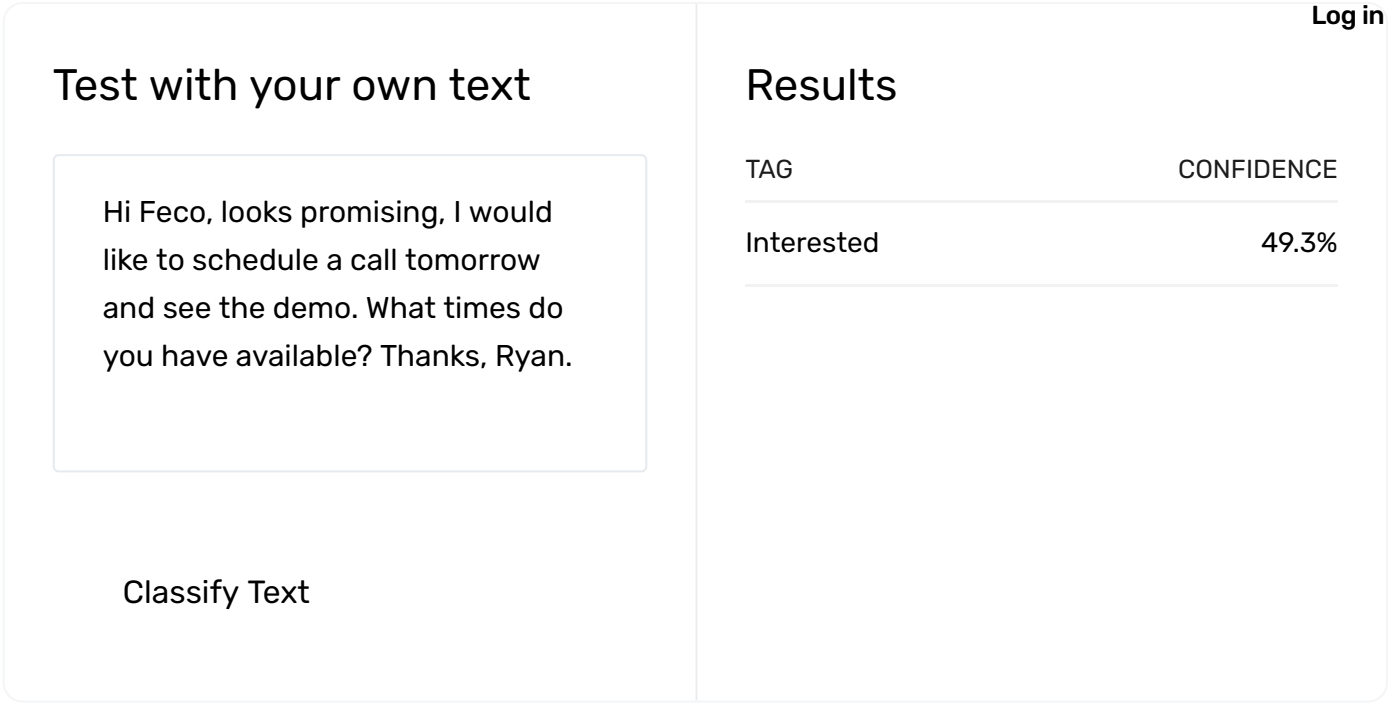
## Results

| TAG | CONFIDENCE |
| --- | --- |
| English-en | 100% |

## Intent Detection

Companies are also using text classifiers for automatically detecting the intent from customer conversations, often used for generating product analytics or automating business purposes.

For example, the following classifier was trained for detecting the intent from replies (https://app.monkeylearn.com/main/classifiers/cl_v9GTn7zi/) in outbound sales emails. It can classifier answers in tags such as *Interested*, *Not Interested*, *Unsubscribe*, *Wrong Person*, *Email Bounce*, and *Autoresponder*:




## Use Cases

Text classification can be applied to a wide range of tasks. In some cases, classifiers work behind the scenes to empower product features we inadvertently interact with on a daily basis (such as spam filtering on emails clients). In some other cases, classifiers are used by marketers, product managers, engineers, and salespeople to automate business processes and save hours of manual data processing.

When we get asked about the things we can do with text classification, the answer is never short. In this section, we'll cover a few typical applications of this technology including all of the following:

- Social media monitoring.
- Brand monitoring.
- Customer service.
- Voice of customer.

## Social Media Monitoring

According to Hootsuite (https://www.slideshare.net/hootsuite/hootsuite-survey-highlights-importance-of-social-media-across-the-customer-journey/1), nearly half of Americans have interacted with companies or institutions on at least one of their social media networks. All of these interactions represent a *lot* of actionable insights for businesses; just on Twitter alone, users send 500 million tweets (http://www.internetlivestats.com/twitter-statistics/) each day.

*What are people complaining about when they mention a particular brand? What are they praising? How have they reacted to a specific message or campaign?*

The answers to these questions can be found within the sea of data available on social media, but without the help of computers, making sense of all this data manually would have to be deemed impossible. Fortunately, machine learning makes it possible to analyze social media data in a scalable and cost-effective way. You can leverage aspect-based sentiment analysis (https://monkeylearn.com/blog/aspect-based-sentiment-analysis/) over a period of time to understand what people are talking about on social media, how they are doing so and track trends over time. You can also use text classification for getting actionable insights such as the following:

- Detecting potential PR crises about to burst;
- Keeping an eye on the competition and detecting sales opportunities when a customer is complaining about their product or service in social media;

- Detecting people who complain about downtime or bugs on social media in order to alert the product team;
- Identify social media interactions that are seeking help and route them to the support team.

Example: A machine learning analysis of the Brexit result (https://monkeylearn.com/blog/machine-learning-analysis-brexit-result/)

When the UK voted to leave the European Union, people were in shock and flooded social media with their opinions on the surprising result. Since it was such a polarizing event, we thought it would be interesting to analyze the conversation on Twitter, so we collected more than 450,000 tweets with the *#Brexit* hashtag and used sentiment analysis (https://monkeylearn.com/sentiment-analysis/) and keyword extraction (https://app.monkeylearn.com/main/extractors/ex_y7BPYzNG/) to get insights from these tweets.

The data confirmed that people's opinion was extremely divided, with slightly more people talking negatively about the results

On the one hand, people tweeting with a positive sentiment were proclaiming that the result was a ‘good thing’, celebrating the ‘independence of the UK’:

On the other hand, people tweeting with a negative sentiment were expressing how sad and disappointed they were at the results:

The results also showed that people were trashing UK politicians such as David Cameron (48% more negative tweets than positive) and especially pro-Brexit politician Nigel Farage (272% more negative tweets than positive). Even Donald Trump was part of the Brexit conversation with a very polarized sentiment with 2808 positive tweets and 3208 negative tweets.

## Brand Monitoring

The online conversation around a brand and its competitors heavily influences consumers. Some blogs, forums, review sites, and influencers are becoming more important than traditional outlets. According to MineWhat, 81% of buyers conduct online research (http://minewhat.com/blog/motivate-shoppers-who-research-online-to-buy/) before making a purchase. Consumers care about what people are saying online about a brand; BrightLocal states that 85% of consumers trust online reviews (https://www.brightlocal.com/learn/local-consumer-review-survey/) as much as personal recommendations.

So, online conversation matters --and that's why you need to create and maintain a process that keeps a close watch on your brand mentions, extracts insights to help drive decisions, and allows you to take action when needed.

With text classification, you can categorize brand mentions all over the internet to find more about the following topics:

- **Features:** are people talking about a particular aspect of your product or service?
- **Wishes:** are your customers expressing particular desires?
- **Price:** is your brand perceived as good value for money? Is it considered cheap or expensive?
- **Use cases:** how do your customers use your product?
- **Competitors:** how does your brand compare to competitors? What are your strengths and weaknesses?

You can create a text classifier to help you identify these topics every time someone shares something about your brand. Moreover, you can combine these topic classifiers with sentiment analysis models to get a real-time thermometer about your online presence.

Example: Sentiment analysis of Slack reviews (https://monkeylearn.com/blog/sentiment-analysis-using-r/)

We scraped 4,000 Slack reviews from Capterra (https://www.capterra.com/) and used text classification to understand which aspects users *love* or *hate* about Slack. The results revealed that Slack is doing great with most reviews talking mostly positively about the company:

Furthermore, we also performed aspect-based sentiment analysis (https://monkeylearn.com/blog/aspect-based-sentiment-analysis/) on the reviews to understand which aspects people are praising or complaining about. The results showed that users love things like its ease of use, integrations, and file sharing system, but hate stuff like the search tool, the notifications system, the pricing, and the performance and reliability:

# Customer Service

Building a good customer experience is one of the foundations of a sustainable and growing company. According to Hubspot, people are 93% more likely to be repeat customers (https://research.hubspot.com/customer-acquisition-study?_ga=2.30579126.869780861.1541182844-402761219.1539702262) at companies with excellent customer service. The study also unveiled that 80% of respondents said they had stopped doing business with a company because of a poor customer experience.

Text classification can help support teams provide a stellar experience by automating tasks (https://monkeylearn.com/customer-service/) that are better left to computers, saving precious time that can be spent on more important things.

For instance, text classification is often used for automating ticket routing and triaging (https://monkeylearn.com/blog/automatically-route-customer-support-tickets-to-the-most-appropriate-person/). Imagine a global company that provides customer support in several languages; this involves the process of assigning tickets based on the ticket's language. To do this, a person is needed to manually assign the ticket to the correct team who can understand and reply to the customer in the right language. With text classification, instead of using humans you can use a language detection (https://app.monkeylearn.com/main/classifiers/cl_Vay9jh28/) classifier to do this task for you.

Text classification can also be used for routing support tickets to a teammate with specific product expertise. For instance, if a customer writes in asking about refunds, you can automatically assign the ticket to the teammate with permission to perform refunds. This will ensure the customer gets a quality response more

quickly. Without the need for triaging every single ticket, support teams can work more efficiently and reduce response times. You'll always know that tickets have been routed to the team that needs to answer them.

Support teams can also use text classification to automatically detect the urgency of a support ticket and prioritize accordingly. By using machine learning to set priorities, you can ensure your team is always working on the most urgent tickets, every time.

Companies are also leveraging text classification for getting insights from support conversations, thus improving their reporting and analytics.

Example: Analyzing customer support interactions on Twitter (https://monkeylearn.com/blog/analyzing-customer-support-interactions-on-twitter-with-machine-learning/)

There are different trends around how to deal with customers in social media. Some support teams try to appear hip and cool while others project a more professional appearance. But, which approach is better received by customers?

To find out, we experimented with keyword extraction (https://app.monkeylearn.com/main/extractors/ex_YCya9nrn/) and sentiment analysis (https://monkeylearn.com/sentiment-analysis/) to analyze 200,000+ customer interactions with Verizon, T-Mobile, AT&T, and Sprint on Twitter. First, we analyzed the most relevant keywords in all these tweets and found out that each carrier has its unique approach towards interacting with customers. For instance, T-Mobile has a friendlier and more personal approach, with every support representative signing each message with their name, while Verizon

tweets are very dry and professional. Then, we performed sentiment analysis on the data, and the results suggest that a friendlier take on social media elicits more positive responses:

## Voice of Customer

When companies leverage surveys such as Net Promoter Score (https://en.wikipedia.org/wiki/Net_Promoter) (NPS) to gather feedback from customers continuously, they start to drive their business decisions based on its results.

To be able to do this, the information gathered, which usually involves open-ended responses, must be processed. By manually annotating responses into different categories, product teams can identify valuable insights and trends over time. The problem is that this manual process is tedious and very time-consuming.

That's when text classification comes in. Instead of relying on humans to do this task, you can quickly process customer feedback with machine learning. Classification models can help you analyze survey results to discover patterns and insights like:

- What do people like about our product or service?
- What should we improve?
- What do we need to change?

By combining the quantitative results with this qualitative but structured analysis, product teams can make more informed decisions without having to put so much time or resources into reading every single open-ended response.

Example: How Retently Automated Customer Feedback Analysis Using MonkeyLearn (https://monkeylearn.com/blog/how-retently-automated-customer-feedback-analysis-using-monkeylearn/)

In their effort to obtain actionable insights for roadmap improvements, Retently (https://www.retently.com/) wanted to figure out what was driving their NPS score. But, manually sorting through all the feedback was quite a time-consuming task that they preferred to avoid. So, they turned to machine learning to automate this process and trained a classifier that was able to classify NPS open-ended responses into the following tags:

Excited about the results of the classifier, Retently decided to implement a new reporting system that can showcase customer priorities from their own custom words:

This new report system allowed Retently to discover actionable insights about their customers that now drives strategic decisions to provide a better user experience.

# Text Classification Resources

So, you want to start using text classification? Great idea! Machines are much faster at processing than humans are. You can begin to automate manual and repetitive tasks so that you can focus on more important and fulfilling activities.

But...how the heck do you get started? There's so much information out there about machine learning and natural language processing that it can be *overwhelming*. Where's the starting point?

Well, don't worry. Building your first text classifier can be simple and straightforward. You just need two things:

1. A dataset to provide examples for training the classifier.
2. A tool for generating and consuming the classifier.

Below, we'll expand on these concepts, provide several options, and we'll end up with a simple hands-on tutorial to create your first classifier. Sounds good? Awesome, let's dive in!

## Datasets

A text classifier is worthless without accurate training data to power it. Just like humans, machine learning algorithms can make predictions by learning from previous examples. By telling the algorithm that you expect a specific set of tags as output for a particular text, it can learn to recognize patterns in text, like the sentiment expressed by a tweet, or the topic mentioned in a customer review.

An accurate classifier depends entirely on getting the right training data, which means gathering examples that best represent the outcomes you want to predict. Say you want to predict the intent from chat conversations; you'll need to identify and gather chat conversations that represent the different intents you want to predict. If you train your model with another type of data, the classifier will provide poor results.

So, how do you get training data?

You can use *internal data* generated from the apps and tools that you use every day such as CRMs (e.g. Salesforce, Hubspot), chat apps (e.g. Slack, Drift, Intercom), help desk software (e.g. Zendesk, Freshdesk, Front), survey tools (e.g. SurveyMonkey, Typeform, Google Forms), and customer satisfaction tools (e.g. Promoter.io, Retently, Satismeter). These tools usually provide an option to export data in a CSV file that you could use for training your classifier.

Another option is using *external data* available on the web, either by using web scraping, APIs, or public datasets.

The following are some publicly available datasets that you can use for building your first text classifier and start experimenting right away.

Topic classification:

- Reuters news dataset (http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html): probably one the most widely used dataset for text classification, it contains 21,578 news articles from Reuters labeled with 135 categories according to their topic, such as Politics, Economics, Sports, and Business.
- 20 Newsgroups (http://qwone.com/~jason/20Newsgroups/): another popular dataset that consists of ~20,000 documents across 20 different topics.

Sentiment analysis:

- Amazon Product Reviews (http://jmcauley.ucsd.edu/data/amazon/): a well-known dataset that contains ~143 million reviews and star ratings (1 to 5 stars) spanning May 1996 - July 2014. You can get an alternative dataset for Amazon product reviews here.

- IMDB reviews (http://ai.stanford.edu/~amaas/data/sentiment/): a much smaller dataset with 25,000 movie reviews labeled as positive and negative from the Internet Movie Database (IMDB).
- Twitter Airline Sentiment (https://www.kaggle.com/crowdflower/twitter-airline-sentiment): this dataset contains around 15,000 tweets about airlines labeled as positive, neutral, and negative.

Other popular datasets:

- Spambase (https://archive.ics.uci.edu/ml/datasets/Spambase): a dataset with 4,601 emails labeled as spam and not spam.
- SMS Spam Collection (https://www.kaggle.com/uciml/sms-spam-collection-dataset): another dataset for spam detection that consists of 5,574 SMS messages tagged as spam or legitimate.
- Hate speech and offensive language (https://github.com/t-davidson/hate-speech-and-offensive-language): this dataset contains 24,802 labeled tweets organized into three categories: clean, hate speech, and offensive language.

## Tools

Alright. Now that you have training data, it's time to feed it to a machine learning algorithm and create a text classifier.

So, how do we do this?

Luckily, many resources can help you during the different phases of the process, i.e. transforming texts into vectors, training a machine learning algorithm, and using a model to make predictions. Broadly speaking, these tools can be classified into two different categories:

- Open Source libraries

# Open Source libraries for Text Classification

One of the reasons machine learning is becoming mainstream is because of the myriad of open source libraries available for developers interested in applying it. Although they still require machine learning knowledge for building and deploying models, these libraries offer a fair level of abstraction and simplification. Python, Java, and R all offer a wide selection of machine learning libraries that are actively developed and provide a diverse set of features, performance, and capabilities.

### Text Classification with Python

Python is often the programming language of choice for developers and data scientists who need to work in machine learning models. The simple syntax, its massive community, and the scientific-computing friendliness of its mathematical libraries are some of the reasons why Python is so prevalent in the field.

Scikit-learn (http://scikit-learn.org/) is one of the go-to libraries for general purpose machine learning. It supports many algorithms and provides simple and efficient features for working with text classification, regression, and clustering models. If you are a beginner in machine learning, scikit-learn is one of the most friendly libraries for getting started with text classification with lots of tutorials and step-by-step guides all over the web.

NLTK (https://www.nltk.org/) is a popular library focused on Natural Language Processing (https://monkeylearn.com/blog/definitive-guide-natural-language-processing/) (NLP) that has a big community behind it. Its super handy for text classification as it provides all kinds of useful tools for making a machine understand text such as splitting paragraphs into sentences, splitting up words, and recognizing the part of speech of those words.

A modern and newer NLP library is SpaCy (https://spacy.io/), a toolkit with a more minimal and straightforward approach than NLTK. For example, spaCy only implements a single stemmer (NLTK has 9 different options). SpaCy has also integrated word embeddings (https://monkeylearn.com/blog/word-embeddings-transform-text-numbers/) which can be useful to help boost accuracy in text classification.

Once you are ready to experiment with more complex algorithms, you should check out deep learning libraries like Keras, TensorFlow, and PyTorch. Keras (https://keras.io/) is probably the best starting point as its designed to simplify the creation of recurrent neural networks (RNNs) and convolutional neural networks (CNNs). TensorFlow (https://www.tensorflow.org/) is the most popular open source library for implementing deep learning algorithms. Developed by Google and used by companies such as Dropbox, eBay, and Intel, this library is optimized for setting up, training, and deploying artificial neural networks with massive datasets. Although it's harder to master than Keras, it's the undisputed leader in the deep learning space. A reliable alternative to TensorFlow is PyTorch (https://pytorch.org/), an extensive deep learning library primarily developed by Facebook and backed by Twitter, Nvidia, Salesforce, Stanford University, University of Oxford, and Uber.

**Text Classification with Java**

Another programming language that is broadly used for implementing machine learning models is Java. Like Python, it has a big community, an extensive ecosystem, and a great selection of open source libraries for machine learning and NLP.

CoreNLP (https://stanfordnlp.github.io/CoreNLP/) is the most popular framework for NLP in Java. Created by Stanford University, it provides a diverse set of tools for understanding human language such as a text parser, a part-of-speech (POS) tagger, a named entity recognizer (NER), a coreference resolution system, and information extraction tools.

Another popular toolkit for natural language tasks is OpenNLP (https://opennlp.apache.org/). Created by The Apache Software Foundation, it provides a bunch of linguistic analysis tools useful for text classification such as tokenization, sentence segmentation, part-of-speech tagging, chunking, and parsing.

Weka (https://www.cs.waikato.ac.nz/ml/weka/) is a machine learning library developed by the University of Waikato and contains many tools like classification, regression, clustering, and data visualization. It provides a graphical user interface for applying Weka's collection of algorithms directly to a dataset, and an API to call these algorithms from your own Java code.

**Text Classification with R**

The R language is an approachable programming language that is becoming increasingly popular among machine learning enthusiasts. Historically, it has been widely used among academics and statisticians for statistical analysis, graphics representation, and reporting. According to KDnuggets (https://www.kdnuggets.com/2017/08/python-overtakes-r-leader-analytics-data-science.html), it's currently the second most popular programming language for analytics, data science, and machine learning (while Python is #1).

R is an excellent choice for text classification tasks as it provides an extensive, coherent, and integrated collection of tools for data analysis.

Caret (http://topepo.github.io/caret/index.html) is a comprehensive package for building machine learning models in R. Short for “Classification and Regression Training”, it offers a simple interface for applying different algorithms and contains useful tools for text classification such as pre-processing, feature selection, and model tuning.

Mlr (https://mlr-org.github.io/mlr/) is another R package that provides a standardized interface for using classification and regression algorithms along with their corresponding evaluation and optimization methods.

## SaaS APIs for Text Classification

Open source tools are great, but they are mostly targeted at people with a background in machine learning. Also, they don’t provide an easy way to deploy and scale machine learning models, clean and curate data, tag training examples, do feature engineering, or bootstrap models.

You might be wondering, is there an easier way?

Well, if you want to avoid these hassles, a great alternative is to use a Software as a Service (SaaS) for text classification which usually solves most of the problems mentioned above. Another advantage is that they don’t require machine learning experience and even people who don’t know how to code can use and consume text classifiers. At the end of the day, leaving the heavy lifting to a SaaS can save you time, money, and resources when implementing a text classification system.

Some of the most remarkable SaaS solutions and APIs for text classification include:

- MonkeyLearn
- Google Cloud NLP

- IBM Watson
- Lexalytics
- MeaningCloud
- Amazon Comprehend
- Aylien

What is Text Classification? How does it work? Applications Resources

# Text Classification Tutorial

The best way to learn about text classification is to get your feet wet and build your first classifier. If you don't want to invest too much time learning about machine learning or deploying the required infrastructure, you can use MonkeyLearn (https://monkeylearn.com/), a platform that makes it super easy to build, train, and consume text classifiers. You can sign up for free (https://app.monkeylearn.com/accounts/register/) and build your own classifier following these four simple steps:

**1. Create a new text classifier:**

Go to the dashboard (https://app.monkeylearn.com/main/explore/), then click Create a Model (https://app.monkeylearn.com/main/module-create/wizard/choose-module-type/), and choose *Classifier*:

**2. Upload training data:**

Next, you'll need to upload the data that you want to use as examples for training your model. You can upload a CSV or Excel file or import your text data directly from a 3rd party app such as Twitter, Gmail, Zendesk, or RSS feeds:

### 3. Define the tags for your model:

The next step is to define the tags you want to use for your text classifier:

Once the classifier has been trained, incoming data will be automatically categorized into the tags you specify in this step. Try avoiding using tags that are overlapping or ambiguous as this can cause confusion and can make the classifier's accuracy worse.

**4. Tag data to train the classifier:**

Finally, you'll need to tag each example with the expected category to start training the machine learning model:



As you tag data, the classifier will learn to recognize similar patterns when presented with new text and make an accurate classification. Remember: the more data you tag, the more accurate the model will be.

## Testing the classifier

Once you've finished the creation wizard, you will be able to test the classifier in "Run" > "Demo" and see how the model classifies the texts you write:



MonkeyLearn provides some useful tools for understanding how well the model is working such as classifier stats (http://help.monkeylearn.com/text-classification/custom-classifiers/understanding-classifier-statistics) (e.g. accuracy, F1 score, precision, and recall) and a keyword cloud

(http://help.monkeylearn.com/text-classification/custom-classifiers/using-keywords-to-increase-classifier-performance) of n-grams for each category. There are multiple ways for improving the accuracy (http://help.monkeylearn.com/text-classification/custom-classifiers/improving-recall-and-precision) of your classifier, including tagging more training data, going through the false positives and false negatives and retag the incorrectly labeled examples, and cleaning your data to disassociate keywords with a specific tag.

## Integrating the classifier

Once the predictions are good enough, the model will be ready to categorize new unseen text. MonkeyLearn provides different ways to achieve this: batch processing, API, or integrations.

You can upload a CSV or Excel file (http://help.monkeylearn.com/analyzing-text-and-processing/batch-processing-of-text-analysis) to classify text in a batch in "Run" > “Batch”:

After uploading the file, the classifier will analyze the data and return a new file with the same data plus the predictions.

Alternatively, you can use MonkeyLearn API (https://monkeylearn.com/api/) to classify new data programmatically:

Another possibility is to use one of the available integrations (http://help.monkeylearn.com/integrations) to put the classifier to work and automatically categorize incoming text in your favorite apps with zero lines of code:

# Takeaway

Text classification can be your new secret weapon for building cutting-edge systems and organizing business information. Turning your text data into quantitative data is incredibly helpful to get actionable insights that can drive

business decisions. You can also automate manual and repetitive tasks and get more done.



Are you interested in creating your first text classifier? Register (/register) or signup to MonkeyLearn (https://monkeylearn.com/) for free and start experimenting right away. You can quickly create text classifiers with machine learning by using our easy-to-use UI (no coding required!) and put them to work by using our API (https://monkeylearn.com/api/v3/) or integrations (https://monkeylearn.com/integrations/).

Have questions? Reach out (/contact) and we'll help you get started with text classification.

# Start using Text Classification today!

Automate business processes and save hours
of manual data processing.

**SIGN UP FOR FREE**

(https://app.monkeylearn.com/accounts/register/)