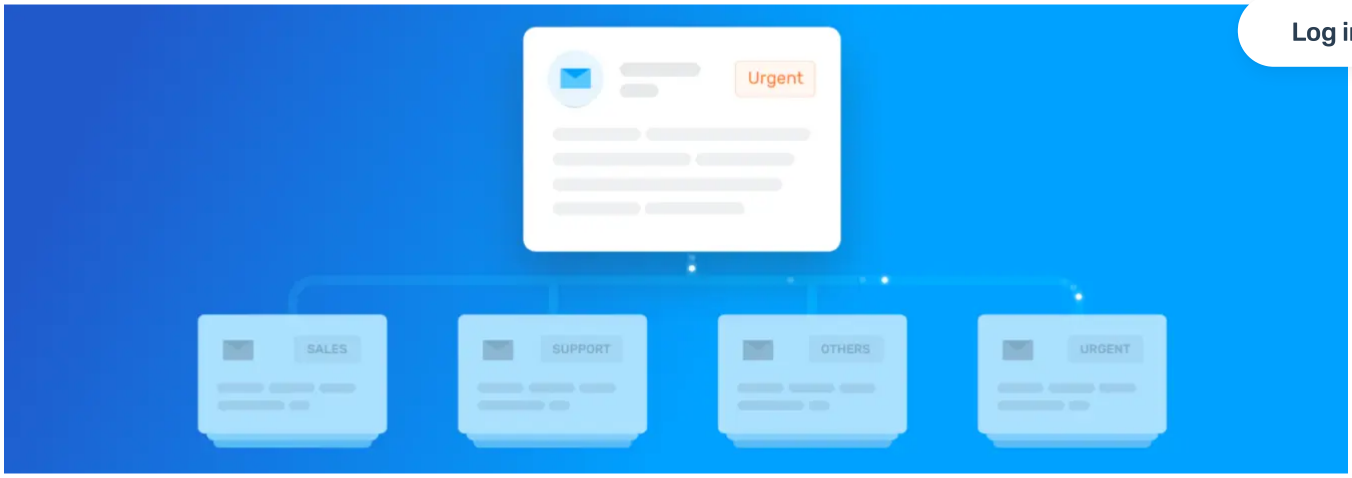
Topic Analysis



Topic analysis is a Natural Language Processing (https://monkeylearn.com/blog/definitive-guide-natural-language-processing/) (NLP) technique that allows us to automatically extract meaning from texts by identifying recurrent themes or topics.

Businesses deal with large volumes of unstructured text every day. Think about all the emails, support tickets, social media posts, customer feedback, reviews and other information that an organization sends and receives. The list is endless.

When it comes to analyzing huge amounts of text data, it’s too big a task to do manually. It’s also tedious, time-consuming, and therefore expensive, and manually sorting through large amounts of data is more likely to lead to mistakes and inconsistencies. Plus, it doesn’t scale well.

The good news is that businesses that handle large amounts of data can benefit greatly from topic analysis techniques.

Topic analysis models enable you to sift through large sets of data and identify the most frequent topics in a very simple, fast and scalable way.

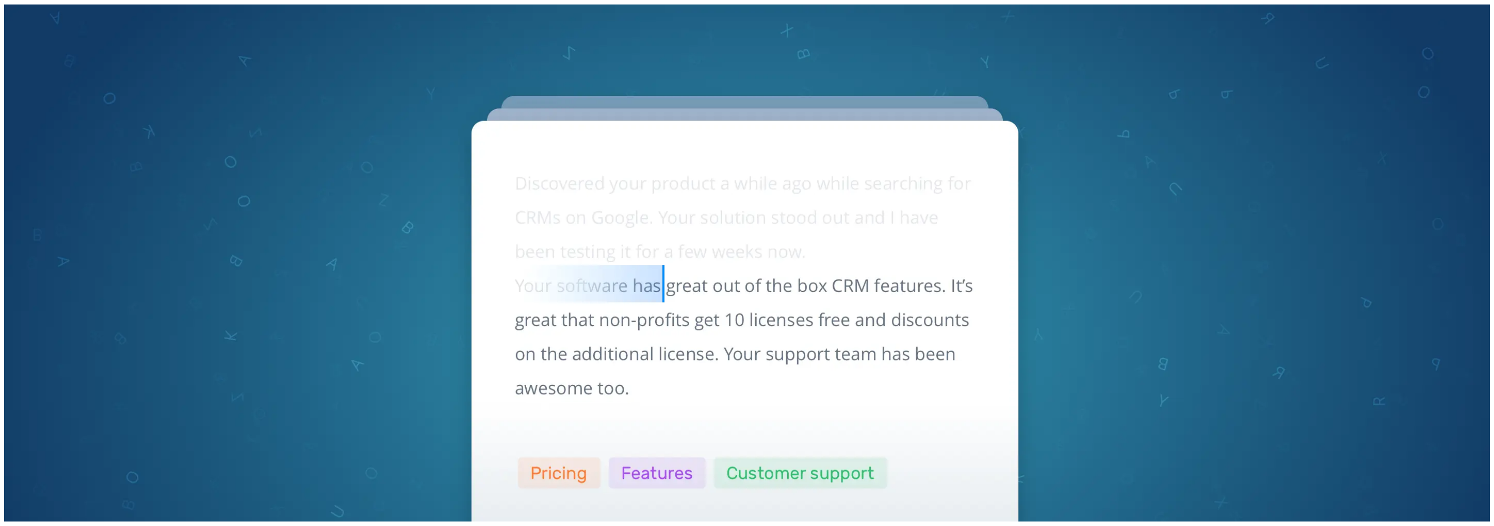
But what exactly is topic analysis? How does it work? What are its main applications? And finally, how can you start using it? This guide will shed some light on all these questions and provide nearly all you need to know about automated topic analysis.

Read this guide in your spare time, bookmark it for later, or jump to the sections that interest you the most:

3. **Use Cases and Applications**
    - Social Media Monitoring
    - Brand Monitoring
    - Customer Service
    - Voice of Customer
    - Business Intelligence
    - Sales and Marketing
    - Product Analytics
    - Knowledge Management
4. **Resources**
    - APIs
    - Papers
    - Courses and Lectures
    - Tutorials

Let's get started!

Introduction to Topic Analysis


Discovered your product a while ago while searching for CRMs on Google. Your solution stood out and I have been testing it for a few weeks now.
Your software has great out of the box CRM features. It's great that non-profits get 10 licenses free and discounts on the additional license. Your support team has been awesome too.

What is Topic Analysis?

Topic analysis (often referred to as *topic detection* or *topic modeling*) is a machine learning (https://monkeylearn.com/blog/gentle-guide-to-machine-learning/) technique that organizes and understands large collections of text data, by assigning tags or categories according to each individual text's topic or theme. In other words, they enable you to find patterns and unlock semantic structures behind each of the individual texts.

The two most common approaches for topic analysis with machine learning are **topic modeling** and **topic classification**.

Topic modeling is an unsupervised machine learning technique. This means it can infer patterns and cluster similar expressions without needing to define topic tags or train data beforehand. This type of algorithm can be applied quickly and easily, but there's a downside – they are rather *inaccurate*.

Text classification (https://monkeylearn.com/text-classification/), on the other hand, needs to know the topics of a text before starting the analysis, because you need to tag data in order to train a topic classifier. Although there's an extra step involved, topic classifiers pay off in the long run and they're much more precise than clustering techniques.

We'll look more closely at these two approaches in the section How It Works.

Examples of Topic Analysis?

Here are some examples to help you better understand the potential uses of automatic topic analysis:

- Topic labeling is used to identify the topic of a news headline. What is a news article talking about? Is it `Politics`, `Sport`, or `Economy`? For example:

"iPhone sales drop 20 percent in China as Huawei gains market share"

A topic model would infer the general topic of this headline is Economy by identifying words and expressions related to this topic (`sales` - `drop` - `percent` - `China` - `gains` - `market share`).

- Topic analysis is used to automatically understand which type of issue is being reported on any given Customer Support Ticket. Is this ticket about `Billing Issues`, `Account Issues` or `Shipping Issues`? For example:

"My order hasn't arrived yet" will be tagged as a Shipping Issue.

- Topic analysis can be used to analyze an open-ended question on a customer satisfaction survey, in order to find out which aspect the customer is referring to. For example:

Question: *"What is the one thing that we could do to improve your experience with [product]?"* Answer: *"Improve the shopping cart experience, it's super confusing."*

The topic of this answer is `UI/UX`.

Scope

Topic analysis can be applied at different levels of scope:

- **Document-level**: the topic model obtains the different topics within a complete text. For example, the topics of an email or an article.
- **Sentence-level**: the topic model obtains the topic of a single sentence. For example, the topic of a news article headline.
- **Sub-sentence level**: the topic model obtains the topic of sub-expressions within a sentence. For example, different topics within a single sentence of a product review.

When is Topic Analysis Used?

Topic tagging is particularly useful to analyze huge amounts of data in a fast and cost-effective way. Yes, you could do it manually but, let's face it, when there's too much information to be classified, the task often ends up being time-consuming, expensive and inaccurate.

At MonkeyLearn (https://monkeylearn.com/), we help companies use topic analysis to make their teams more efficient, automate business processes, get valuable insights from data, and save hours of manual data processing.

Imagine you need to analyze a large dataset of reviews to find out what people are saying about your product. You could combine topic labeling with sentiment analysis (https://monkeylearn.com/sentiment-analysis) to discover which aspects or features of your product are being discussed (topics) most often, and determine how people feel about them (are their statements positive, negative or neutral?). This technique is known as aspect-based sentiment analysis (https://monkeylearn.com/blog/aspect-based-sentiment-analysis/).

Besides brand monitoring, there are many other uses for topic analysis, such as social media monitoring, customer service, voice of customer (VoC), business intelligence, sales and marketing, SEO, product analytics and knowledge management.

We'll go into each of these uses in more detail in our "Applications" section, where we'll present real use cases and examples that will help you understand the full potential of topic analysis and the benefits of using it within your company.

Why is Topic Analysis Important?

Businesses generate and collect huge volumes of data every day. Analyzing and processing this data using automated topic analysis will help businesses make better decisions, optimize internal processes, identify trends and all sorts of other advantages that will make companies much more efficient and productive.

When it comes to sorting through massive amounts of data, machine learning models are crucial. Topic detection allows us to easily scan large documents and find out what customers are talking about.

Among the advantages of topic modeling are:

- **Scalability**

If you had to manually detect topics by sifting through a huge database, not only it would be very time-consuming, it would also be far too expensive. Automated topic analysis with machine learning makes it possible to scan as much data as you want, providing brand new opportunities to obtain meaningful insights.

- **Real-time analysis**

By combining topic tagging with other types of natural language processing techniques, like sentiment analysis, you can obtain a real-time picture of what your clients are saying about your product. And most importantly, you can use that information to make data-driven decisions.

- **Consistent Criteria**

Automated topic analysis is based on Natural Language Processing (https://monkeylearn.com/blog/definitive-guide-natural-language-processing/) (NLP) — a combination of statistics, computational linguistics, and computer science — so you can expect high-quality results and fewer errors that may occur with manual processing.

Wrap-up

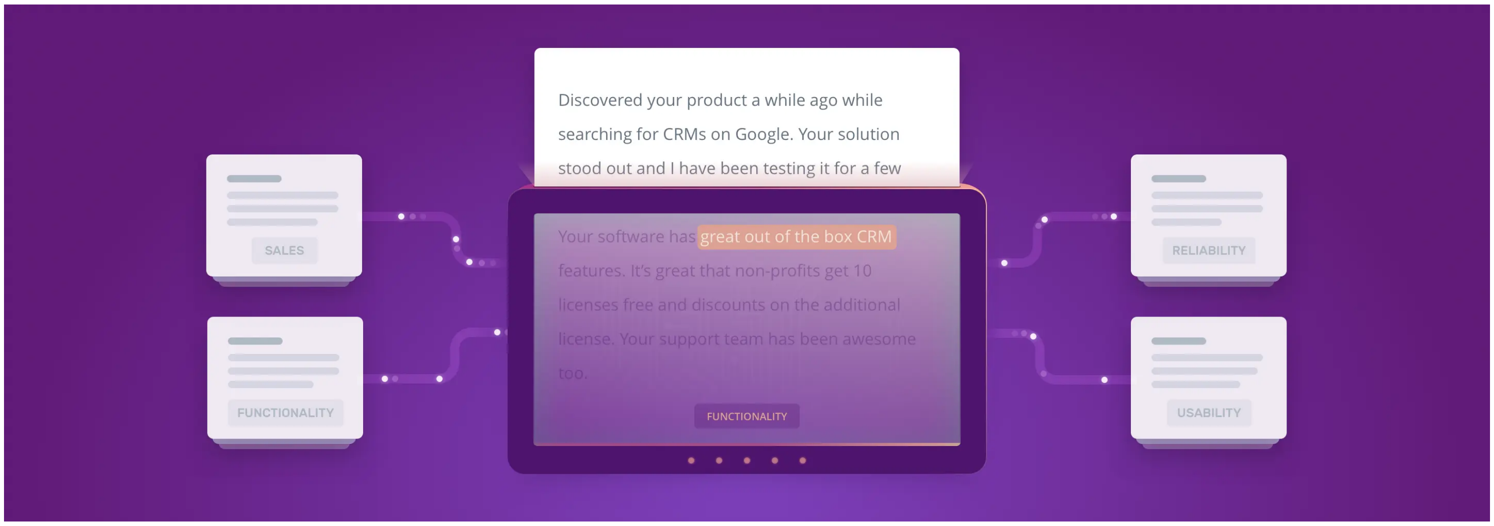
Topic analysis is a fast-growing approach that enables businesses to analyze large quantities of unstructured text data, and understand the main topics of each text.

They’re able to obtain relevant insights by analyzing data sources like reviews, NPS surveys, posts on social media, emails, and customer support tickets. Besides discovering the most frequently discussed topics about your product or service, you can also use a topic model to make new discoveries about your business.

The following section explores how topic models work, as well as the different algorithms that can be used.

Let’s move forward!

How does Topic Analysis work


Discovered your product a while ago while searching for CRMs on Google. Your solution stood out and I have been testing it for a few
Your software has great out of the box CRM features. It's great that non-profits get 10 licenses free and discounts on the additional license. Your support team has been awesome too.
FUNCTIONALITY
SALES
FUNCTIONALITY
RELIABILITY
USABILITY

Topic analysis models are able to detect topics within a text, simply by counting words and grouping similar word patterns.

Let's imagine you want to find out what customers are saying about various features of a new laptop.

Your topics of interest might be *Portability*, *Design*, and *Price*. Now, instead of going through rows and rows of customer feedback with a fine-tooth comb, in an attempt to separate feedback into topics of interest, you'll be able to run a topic analysis.

For *Price*, analysis models might detect patterns such as currency symbols followed by numbers, related words (affordability, expensive, cheap), synonyms (cost, price tag, worth) or phrases (worth every penny), and label the corresponding text accordingly.

Now, let's imagine you don't have a list of predetermined topics. Topic analysis models can also detect topics by counting word frequency and the distance between each other. Sounds simple, right? Well, it's a little more complicated than just counting and spotting patterns.

Topic Modeling vs Topic Classification

There are many approaches and techniques you can use to automatically analyze the topics of a set of documents, and the one you decide to use depends on the problem at hand. To understand the ins and outs of how topic analysis models work, we're going to focus on the two most common approaches.

If you just have a bunch of texts and want to figure out what topics these texts cover, what you're looking for is **topic modeling**.

Now, if you already know the possible topics for your texts and just want to label them automatically without having to read them one by one, you want **topic classification**.

These problems might sound similar, but they are entirely different beasts, solved using different algorithms

It's important to figure out what your needs are before blindly jumping into a solution that may not be what you're actually looking for.

Of course, you can always solve your problem manually. Sit down, read tens or hundreds of thousands of documents, understand them, make a list of topics, and label every one of them with your topics.

Sounds fun, right?

Right?

Enter machine learning (https://monkeylearn.com/blog/gentle-guide-to-machine-learning/). It can be used to automate tedious and time-consuming manual tasks. There are many machine learning algorithms that, given a set of documents and a couple of friendly nudges, are able to automatically infer the topics on the dataset, based on the content of the texts themselves.

The majority of these algorithms are unsupervised (https://en.wikipedia.org/wiki/Unsupervised_learning), which means that you feed them the texts and training parameters and they do the rest. Topic modeling runs on this kind of algorithm.

On the other hand, you have supervised (https://en.wikipedia.org/wiki/Supervised_learning) algorithms. Machines are fed examples of data labeled according to their topics so that they eventually learn how to tag text by topic, by themselves. These are the types of algorithms commonly used for topic classification.

Unsupervised machine learning algorithms are, in theory, less work-intensive than supervised algorithms, since they don't require human tagged data. They may, however, require quality data in large quantities. It could or could not be easy to come by, depending on your use case.

In this case, it may be advantageous to just run unsupervised algorithms and discover topics in the text, as part of an analysis process.

Having said this, topic modeling algorithms will not deliver neatly packaged topics with labels such as `Sports` and `Politics`. Rather, they will churn out collections of documents that the algorithm thinks are related, and specific terms that it used to infer these relations. It will be your job to figure out what these relations actually mean.

On the other hand, supervised machine learning algorithms require that you go through the legwork of explaining to the machine what it is that you want, via the tagged examples that you feed it. Therefore, the topic definition and tagging process are important steps that should not be taken lightly, since they make or break the real-life performance of the model.

The advantages of supervised algorithms win hands down, though. You can refine your criteria and define your topics, and if you're consistent in the labeling of your texts, you will be rewarded with a model that will classify new, unseen samples according to their topics, the same way you would.

Now, let's go further and understand how both topic modeling and topic classification actually work.

Topic Modeling

Topic modeling solves the following type of problem: you have a set of text documents (such as emails, survey responses, support tickets, product reviews, etc), and you want to find out the different topics that they cover and group them by those topics.

The way these algorithms work is by assuming that each document is composed of a mixture (https://www.statisticshowto.datasciencecentral.com/mixture-distribution/) of topics, and then trying to find out how strong a presence each topic has in a given document. This is done by grouping together the documents based on the words they contain, and noticing correlations between them.

Although similar, topic modeling shouldn't be confused with cluster analysis (https://en.wikipedia.org/wiki/Cluster_analysis).

To better understand the ideas behind topic modeling, we will cover the basics of two of the most popular algorithms: LSA and LDA.

Latent Semantic Analysis (LSA)

Latent Semantic Analysis is the ‘traditional' method for topic modeling. It is based on a principle called the distributional hypothesis (https://en.wikipedia.org/wiki/Distributional_semantics): words and expressions that occur in similar pieces of text will have similar meanings.

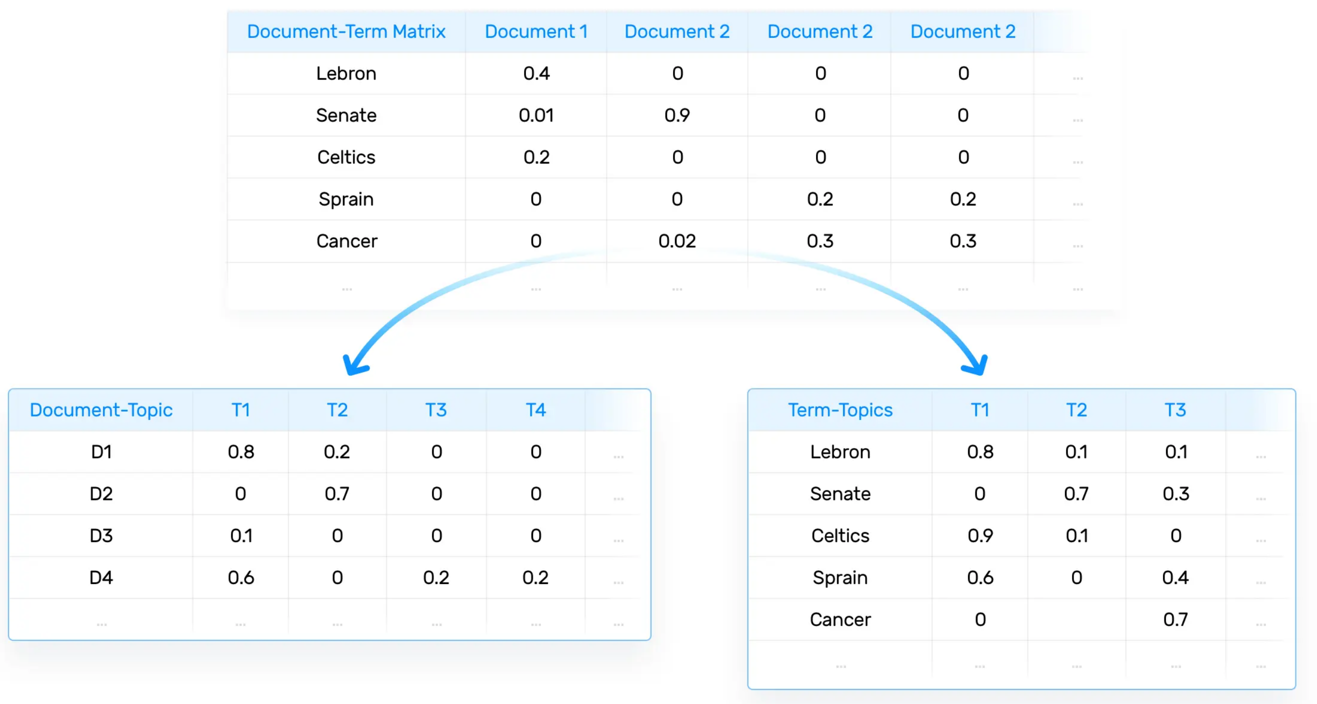
Like Naive Bayes (https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/), it is based on the word frequencies of the dataset. The general idea is that for every word in each document, you can count the frequency of that word and group together the documents that have high frequencies of the same words.

Now, to dive a little bit more into how this is actually done, let's first clarify what is meant by word frequency. The *frequency* of a word or term in a document is a number that indicates how often a word appears in a document. That's right, these algorithms ignore syntax and semantics such as word order, meaning, and grammar, and just treat every document as an unsorted bag of words.

The frequency can be calculated simply by counting — if the word `cat` appears 10 times in a document, then its frequency is 10. This approach proves to be a bit limited, so tf-idf (https://en.wikipedia.org/wiki/Tf%E2%80%93idf) is normally used. Tf-idf takes into account how common a word is overall (in all documents) vs how common it is in a specific document, so more common words are ranked higher since they are considered a better “representation” of a document even if they are not the most numerous.

After doing the word frequency calculation, we are left with a matrix that has a row for every word and a column for every document. Each cell is the calculated frequency for that word in that document. This is the *document-term matrix*; it relates documents to terms.

Hidden inside it is what we want: a *document-topic matrix* and a *term-topic matrix*, which relate documents to topics and terms to topics. These matrices are the ones that show information about the topics of the texts.



Document-Term Matrix	Document 1	Document 2	Document 2	Document 2	...
Lebron	0.4	0	0	0	...
Senate	0.01	0.9	0	0	...
Celtics	0.2	0	0	0	...
Sprain	0	0	0.2	0.2	...
Cancer	0	0.02	0.3	0.3	...
...	...	...	...	...	...

Document-Topic	T1	T2	T3	T4	...
D1	0.8	0.2	0	0	...
D2	0	0.7	0	0	...
D3	0.1	0	0	0	...
D4	0.6	0	0.2	0.2	...
...	...	...	...	...	...

Term-Topics	T1	T2	T3	...
Lebron	0.8	0.1	0.1	...
Senate	0	0.7	0.3	...
Celtics	0.9	0.1	0	...
Sprain	0.6	0	0.4	...
Cancer	0		0.7	...
...	...	...	...	...

The way these matrices are generated is by decomposing the document-term matrix into three matrices using a technique called *truncated SVD*. Firstly, singular-value decomposition (https://en.wikipedia.org/wiki/Singular-value_decomposition) (SVD) is a linear algebra algorithm for factorizing a matrix into the product of three matrices *U * S * V*.The important part is that the middle matrix *S* is a diagonal matrix (https://en.wikipedia.org/wiki/Diagonal_matrix) of the singular values (https://en.wikipedia.org/wiki/Singular_value) of the original matrix. For LSA, every one of the singular values represents a potential topic.

Truncated SVD selects the largest `t` singular values and keeps the first `t` columns of *U* and the first `t` rows of *V*, reducing the dimensionality of the original decomposition. `t` will be the number of topics that the algorithm finds, so it's a hyperparameter that will need tuning. The idea is that the most important topics are selected, and *U* is the document-topic matrix and *V* is the term-topic matrix.

The vectors that make up these matrices represent documents expressed with topics and terms expressed with topics; they can be measured with techniques such as cosine similarity (https://en.wikipedia.org/wiki/Cosine_similarity) to evaluate.

Latent Dirichlet Allocation (LDA)

Here is where things start getting a bit technical. Understanding LDA fully involves some advanced mathematical probability topics. However, the basic idea behind it is more easily understood.

Imagine a fixed set of topics. We define each topic as represented by an (unknown) set of words. These are the topics that our documents cover, but we don't know what they are yet. LDA tries to map all the (known) documents to the (unknown) topics in a way such that the words in each document are mostly captured by those topics.

The fundamental assumption here is the same used for LSA: documents with the same topic will use similar words. It's also assumed as well that every document is composed by a *mixture* of topics, and every word has a *probability* of belonging to a certain topic.

LDA assumes documents are generated the following way: pick a mixture of topics (say, 20% topic A, 80% topic B, and 0% topic C) and then pick words that belong to those topics. The words are picked at random according to how likely they are to appear in a certain document.

Of course, in real life documents aren't written this way, that would be madness. Documents are written by humans and have characteristics that make them readable, such as word order, grammar, etc. However, it can be argued that just by looking at the words of a document, you can detect the topic, even if the actual message of the document doesn't come through.

This is what LDA does. It sees a document and assumes that it was generated as described above. Then it works backward from the words that make up the document and tries to guess the mixture of topics that resulted in that particular arrangement of words.

The way this is achieved exceeds the scope of this article but if you'd like to learn more, a good starting point is the original LDA paper.

Something we should mention about the implementation is that it has two hyperparameters for training, usually called α (alpha) and β (beta). Knowing what these do is important for using libraries that implement the algorithm.

Alpha controls the similarity of documents. A low value will represent documents as a mixture of few topics, while a high value will output document representations of more topics -- making all the documents appear more similar to each other.

Beta is the same but for topics, so it controls topic similarity. A low value will represent topics as more distinct by making fewer, more unique words belong to each topic. A high value will have the opposite effect, resulting in topics containing more words in common.

Another important thing that has to be specified before training is the number of topics that the model will have. The algorithm cannot decide this by itself, it needs to be told how many topics to find. Then, the output for every document will be the mixture of topics that each particular document has. This output is just a vector; a list of numbers that means "for topic A, 0.2; for topic B, 0.7; ..." and so on. These vectors can be compared in different ways, and these comparisons are useful for understanding the corpus; to get an idea of its fundamental structures.

Topic Classification

Unlike Topic Modeling, in *Topic Classification* you already know what your topics are.

For example, you may have a set of customer support tickets you want to categorize, and you know the possible topics are `Software Issue` and `Billing Issue`. What you want to do is assign one of these topics to each of the tickets, usually to speed up and automate some human-dependent processes. For example, you could automatically assign support tickets, sorted by topic, to the correct person in the team without anybody having to sift through all the tickets

Unlike the algorithms for Topic Modeling, the machine learning algorithms used for Topic Classification are *supervised*. This means that you need to give them documents already labeled with topics, and the algorithms then learn how to label new, never-seen documents with these topics.

Now, how you predetermine topics for your documents is a different issue entirely. If you're looking to automate some already existing task, then you probably have a good idea about the topics of your texts. In other cases, you could use the previously discussed topic modeling methods as a way to better understand the content of your documents beforehand.

What ends up happening in real life scenarios is that the topics are uncovered as the model is built.

Since automated classification — either by rules or machine learning — always involves a first step of manually analyzing and tagging texts, you usually end up refining your topic set as you go. Before you can consider the model finished, your topics should be solid and your dataset consistent.

Next, we will cover the two main paths for automated topic classification: rule-based systems and machine learning systems, and also a little bit about its cute offspring, hybrid systems.

Rule-Based Systems

Before getting into machine learning algorithms, it's important to note that it's possible to build a topic classifier entirely by hand, without machine learning.

The way this works is by directly programming a set of hand-made rules, based on the content of the documents that a human *expert* actually read. The idea is that the rules represent the codified knowledge of the expert, and are able to discern between documents of different topics by looking directly at semantically relevant elements of a text, and at the metadata that a document may have. Each one of these rules consists of a *pattern* and a *prediction* (in this case, a predicted topic).

Back to support tickets, a way to solve this problem using rules would be to define lists of words, one for each topic (e.g. for `Software Issue` words like bug, program, crash, etc, and for `Billing Issue` words like price, charge, invoice, $, and so on). Now, when a new ticket comes in, you count the frequency of software-related words and billing-related words. Then, the topic with the highest frequency gets the new ticket assigned to it.

Rule-based systems such as this are *human comprehensible*; a person can sit down, read the rules, and understand how a model works. Over time it's possible to improve them as well, by refining existing rules and adding new ones.

However, there are some disadvantages. First, these systems require deep knowledge of the domain (remember that we used the word *expert*? It's not a coincidence). They also require a lot of work, because creating rules for a complex system can be quite difficult and requires a lot of analysis and testing to make sure it's working as intended. Lastly, rule-based systems are a pain to maintain and don't scale very well, because adding new rules will affect the performance of the rules that were already in place.

Machine Learning Systems

The world has moved on from rule-based systems, which are now being phased out in favor of more automated systems. This doesn't mean that rule-based models are worse, just that usually machine learning gives better results with much less effort involved.

In machine learning classification, examples of text and the expected categories (AKA *training data*) are used to train a classification model. This model learns from the training data to recognize patterns in order to make a classification into the categories you define.

As a first, training data has to be transformed into something a machine can understand, that is, *vectors* (i.e. lists of numbers which encode information). By using vectors, the model can extract relevant pieces of information (features) which will help it learn from the training data and make predictions. There are different methods to achieve this, but one of the most used is known as the *bag of words vectorization*. You can learn more about vectorization here (https://monkeylearn.com/blog/beginners-guide-text-vectorization/).

Once the training data is transformed into vectors, they are fed to an algorithm which uses them to produce a model that is able to classify the texts to come:

For making new predictions, the trained model transforms an incoming text into a vector, extract its relevant features, and makes a prediction:

The classification model can be improved by training it with more data, and changing the training parameters of the algorithm; these are known as *hyperparameters*.

The following are broad-stroke overviews of machine learning algorithms that can be used for topic classification. For a more in-depth explanation of each one, check out the linked articles.

Naive Bayes

Naive Bayes (https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/) is a family of simple algorithms that usually give great results for small amounts of data and limited computational resources. The most popular member of the family is probably Multinomial Naive Bayes (MNB), and it's one of the algorithms that we use here at MonkeyLearn.

In a nutshell, this algorithm correlates the probability of words appearing in a text with the probability of that text being about a certain topic.

If this sounds kinda similar to what LSA does, it's because it is. The main difference stems from what is done with the data afterwards: LSA looks for patterns in the existing dataset, while MNB uses the existing dataset to make predictions for new texts.

Support Vector Machines

Support Vector Machines (popularly known as SVM) is a tad more complex in its workings than Naive Bayes, but it's still based on a simple idea. It usually gives better results than MNB for topic classification problems but could be quite computationally intensive. However, it's possible to get training times similar to those of an MNB classifier with optimization by feature selection in addition to running an optimized linear kernel such as scikit-learn's LinearSVC (https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html).

The basic idea for SVM is, once all the texts are vectorized (https://monkeylearn.com/blog/beginners-guide-text-vectorization/) (so they are points in mathematical space), to find the best line (in higher dimensional space called a *hyperplane*) that separates these vectors into the desired topics. Then, when a new text comes in, vectorize it and take a look at which side of the line it ends up: that's the output topic.

Deep Learning

Deep learning (https://en.wikipedia.org/wiki/Deep_learning) is actually a catch-all term for a family of algorithms loosely inspired by the way human neurons work. Although the ideas behind Neural Networks are pretty old (as in, they originate in the 1950s), these algorithms have seen a great resurgence in recent years thanks to the decline of computing costs, the increase of computing power and the vast availability of data.

Text classification (https://monkeylearn.com/text-classification/) in general, and topic classification in particular, have greatly benefited from this resurgence and usually offer great results in exchange for some draconian computational requirements. It's not unusual for deep learning models to train for days, weeks, or even months.

For topic classification, the two main deep learning architectures used are Convolutional Neural Networks (https://en.wikipedia.org/wiki/Convolutional_neural_network) (CNN) and Recurrent Neural Networks (https://en.wikipedia.org/wiki/Recurrent_neural_network) (RNN). The differences are outside the scope of this article, but here's (https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f) a good comparison with some real-world benchmarks.

Now, something important to take into account is that deep learning algorithms require much more training data than traditional machine learning algorithms; in the realm of millions of tagged examples. However, it's important to note that traditional machine learning algorithms such as SVM and NB reach a certain threshold, after which adding more training data stops improving the accuracy. In contrast, deep learning classifiers continue to get better the more data they have:

This doesn't mean that the older algorithms are strictly worse; it depends on the task at hand. For instance, spam detection (https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering) was declared "solved" a couple of decades ago using just Naive Bayes and n-grams.

Other deep learning algorithms like Word2Vec (https://en.wikipedia.org/wiki/Word2vec) or GloVe (https://nlp.stanford.edu/projects/glove/) are also used; these are great for getting better vector representations for words when training with other, traditional machine learning algorithms.

Hybrid Systems

The idea behind Hybrid systems is to combine a base machine learning classifier with a rule-based system, that improves the results with fine-tuned rules. These rules can be used to correct topics that haven't been correctly modeled by the base classifier.

Metrics and evaluation

Training models is great and all, but unless you have a repeatable and consistent way to measure your results, you won't be able to know how good your model is and if your recent changes improved it.

In order to measure the performance of a model, you'll need to let it categorize texts that you already know which topic category they fall under, and see how it performed.

The basic metrics analyzed are:

- **Accuracy**: the percentage of texts that were predicted with the correct topic.
- **Precision**: the percentage of texts the model got right out of the total number of texts that it predicted for a given topic.
- **Recall**: the percentage of texts the model predicted for a given topic out of the total number of texts it should have predicted for that topic.
- **F1 Score**: the harmonic mean of precision and recall.

There's a big issue though. You shouldn't use your training data to measure performance, since the model has already seen these samples and it wouldn't be a fair test.

The *ideal* way to measure would be to take part of the manually tagged data, don't use it to train, and when the model is trained then use it to test. This dataset is called a *golden standard*. However, leaving part of your hard-earned data just lying around unused instead of powering the model isn't very optimal. You *could* add this testing data to the final model, but then you have the same problem: you don't know if it was actually better off without it.

The *ideal* way to measure would be to keep aside some of the manually tagged data. This dataset is called a *gold standard*. However, leaving part of your hard-earned data just lying around unused instead of powering the model isn't very optimal. You could add this testing data to the final model, but then you have the same problem: you don't know if it was actually better off without it.

Something that can be done instead is called cross-validation (https://en.wikipedia.org/wiki/Cross-validation_(statistics)). You split the training dataset randomly into equally sized sets (for example 4 sets with 25% of the data each). For each one of these sets, you train a classifier with all the data that's not in this set (75% of the data) and use this set as the gold standard to test.

Then, you build the final model by training with all the data, but the performance metrics you use for it are the average of the partial evaluations.

Wrap-up

We've looked at several systems and algorithms that can be used for each of the two approaches we've introduced above – topic modeling and topic classification. The approach you choose really depends on the problem you're trying to solve. If you just want to separate documents by an unknown set of topics, then a topic modeling algorithm can handle the job.

However, if you do have a set of known topics, we'd recommend using text classification, since it serves up more accurate results and, therefore, better insights for your business. While supervised models require more setup time for creating the training dataset, the advantages far outweigh those of unsupervised models. The difference could be the make or break your business, since the insights you gain from your results could help reshape and boost your business.

At MonkeyLearn, we use text classification with machine learning to detect topics. And, as luck would have it, we already have a model that you can test drive for free (https://app.monkeylearn.com/main/classifiers/cl_sGdE8hD9/).

So, now that we have an understanding of how topic analysis works, where to go from here? The following section will go into depth about the different use cases and applications of topic analysis and how it can help your business. We'll provide some examples to showcase how you can use topic analysis in a wide number of areas within your company, like social media monitoring, brand monitoring, customer support, voice of customer (VoC), business intelligence, sales marketing, product analytics, and knowledge management.

Use Cases & Applications

Topic analysis is helping businesses become more efficient by saving time on repetitive manual tasks, and providing an insightful approach to the text data they manage on a daily basis.

From sales and marketing to customer support and product teams, topic analysis offers endless possibilities across different industries and areas within a company. Let's say you want to uncover the main themes of conversations around your brand in social media, understand the priorities of hundreds of incoming support tickets, or identify brand promoters based on customer feedback. Topic analysis enables you to do all this (and more) in a simple, fast and cost-effective way.

On top of this, getting started is easy; you don't need to have coding skills or know too much about machine learning.

In this section, we'll explore the different applications of automated topic analysis and share some real use cases and examples that will help you understand how businesses are taking advantage of topic classification to analyze large sets of data.

Specifically, we will refer to how topic analysis can be used in the following areas of a company:

- Social Media Monitoring
- Brand Monitoring
- Customer Service
- Voice of Customer
- Business Intelligence
- Sales and Marketing
- Product Analytics
- Knowledge Management

Let's move forward with it!

Social Media Monitoring

Every day, people send 500 million tweets (https://blog.hootsuite.com/twitter-statistics/). Impressive, right? And that's just Twitter! In this immense volume of data generated on social media, there are mentions of products or services, stories of customer experiences (http://time.com/4894182/twitter-company-complaints/) and interactions between users and brands. For companies, following these conversations is vital to get real-time actionable insights from customers, address potential issues or anticipate a crisis. However, analyzing this data manually is not a feasible task.

Topic analysis allows you to automatically **add relevant context to the data obtained through social media**, in order to understand what people are actually saying about your brand. Imagine you work at HubSpot. You could use topic detection to analyze what users are posting about your brand across Twitter, Facebook, and Instagram and easily identify the most common topics in the conversation. Your customers may be alluding to functionalities, ease of use, or maybe referring to customer support issues. With topic analysis, you can get valuable insights such as:

- Understanding what people value the most about your product or service
- Identifying which areas of your product or service are raising more concerns
- Recognizing your pain-points, so that you can use them as opportunities for improvement

You can also use topic detection to keep an eye on your competition and track how trends in your field evolve over time.

If you want to add an extra dimension to your data analysis, the best way is to combine topic detection with sentiment analysis (https://monkeylearn.com/sentiment-analysis/), so you can get a sense of the *feelings* behind the interactions in social media. Aspect-based sentiment analysis (https://monkeylearn.com/blog/aspect-based-sentiment-analysis/) is a machine learning technique that allows you to associate specific sentiments (positive, negative, neutral) to different aspects (topics) of a product or service. In the case of HubSpot, not only would you know that most of your users are talking about your in-flight menu on Twitter, but you could also find out if they are referring to it in a negative or positive way, as well as the main keywords (https://app.monkeylearn.com/main/extractors/ex_YCya9nrn/) they are using for this topic.

Imagine the potential of this if you've just launched a marketing campaign. You would be able to track people's reactions in real time and get an accurate map of people's perceptions about the campaign.

Example: Trump vs Hillary, analyzing Twitter mentions during the US Elections (https://monkeylearn.com/blog/trump-vs-hillary-sentiment-analysis-twitter-mentions/)

At MonkeyLearn, we used machine learning to analyze millions of tweets posted by users during the 2016 US elections. First, we classified tweets by topic, whether they were talking about Donald Trump or Hillary Clinton. Then, we used sentiment analysis to classify tweets as `positive`, `negative` or `neutral`. This allowed us to do all sorts of analysis, like extracting the most relevant keywords (https://app.monkeylearn.com/main/extractors/ex_YCya9nrn/) for the negative tweets about Trump on a particular day.

This graph shows the progression of positive, neutral and negative tweets referring to Trump and Clinton over time. The majority of tweets about both candidates are negative.

Brand Monitoring

It’s not all about social media. Blogs, news outlets, review sites, and forums have proved to have a powerful influence on a brand’s reputation. In fact, nearly 90% of consumers read at least 10 online reviews before forming an opinion about a business (https://www.brightlocal.com/research/local-consumer-review-survey/), and almost 60% will only use a product or service if it has four or more stars. We’ve all been there, whether it’s booking a hotel for your next holiday or downloading a new app on your cell phone, checking the reviews is an inevitable step in our decision-making process.

Therefore, analyzing this kind of information is crucial for every business. Not only to keep track of your brand image (and take action in case of a crisis, or make improvements based on customer feedback), but also to monitor your competitors or detect the latest trends in your industry.

The first thing to do is to collect data about your product from different sources. Then, you can use topic analysis to get insights about your brand, by **detecting and tracking the different areas of your business (topics) that people are discussing the most**.

For a deeper understanding of your data, you could perform aspect-based sentiment analysis (https://monkeylearn.com/blog/aspect-based-sentiment-analysis/) in order to find out the polarity of the customers’ opinions (are their comments positive, negative, neutral?). And if you’d like to go even further, you could combine this with keyword extraction (https://app.monkeylearn.com/main/extractors/ex_YCya9nrn/). This will reveal the most relevant terms of conversation for each of the topics, allowing you to dig deeper into the reasons behind the opinions expressed by customers.

Example: Analyzing Slack reviews on Capterra (https://monkeylearn.com/blog/sentiment-analysis-using-r/)

Yes, you could analyze reviews manually. But honestly, you don't want to do it. It's time-consuming and it may be a hassle if you have large volumes of data. With MonkeyLearn, you can create personalized models with machine learning and train them to do the work automatically. To show you exactly how to do it, we used MonkeyLearn R package (https://github.com/ropensci/monkeylearn) to analyze thousands of Slack reviews on the product review site Capterra.

After scraping the data, we defined a series of topic classifiers that helped us detect what the text in the reviews was about. Some of them were about `pricing`, `UX`, `customer support`, `performance`, etc. Then, we used an opinion unit extractor (https://app.monkeylearn.com/main/extractors/ex_N4aFcea3/) to divide each review into small sentences, and created a sentiment analysis model to classify them as positive, negative or neutral:

The graphic above shows the sentiment for each of the aspects analyzed, weighted by the number of mentions. We see that the aspects users love most about Slack are `ease of use`, `integrations`, and `purpose`, while most of the complaints refer to `performance-quality-reliability`, `pricing`, and `notifications`.

Customer Service

It's not enough to have an amazing product at a competitive price. Being able to deliver a great customer experience can make all the difference and help you stand out from your competitors. According to a 2017 Microsoft report (http://info.microsoft.com/rs/157-GQE-382/images/EN-CNTNT-Report-DynService-2017-global-state-customer-service-en-au.pdf), 96% of the people say customer service has influenced their choice and loyalty to a brand. Plus, 56% stated they stopped using a product or service due to poor customer experience.

With lots of support tickets arriving every day at your helpdesk, a big part of the job in customer service consists of processing large amounts of text. First, you need to find out the subject of each ticket and tag them accordingly, and then triage tickets to the corresponding area that handles each type of issue.

Machine learning opens the door to automating this repetitive and time-consuming task, in order to **save valuable time for your customer support team**, and let them focus on what they can do best: helping customers and keeping them happy. Machine learning algorithms can be trained to sort large volumes of data (in this case, customer support tickets), recognize text patterns and figure out what the ticket is about.

Thanks to a combination of machine learning models (not only topic labeling, but also intent detection, urgency detection, sentiment analysis, and language detection) your customer support team can:

- Automatically tag customer support tickets (https://monkeylearn.com/blog/automate-your-zendesk-ticket-tagging-with-keyword-extraction/), which you can easily do with the MonkeyLearn Zendesk integration (https://monkeylearn.com/zendesk/)

- Automatically triage and route conversations (https://monkeylearn.com/blog/automatically-route-customer-support-tickets-to-the-most-appropriate-person/) to the most appropriate team
- Automatically detect the urgency (https://monkeylearn.com/blog/detecting-squeaky-wheel-urgency-in-customer-support-tickets/) of a support ticket and prioritize accordingly
- Get insights from customer support conversations (https://monkeylearn.com/blog/analyzing-customer-support-interactions-on-twitter-with-machine-learning/)

Voice of Customer (VoC)

Customer feedback is a valuable source of information that provides insights on the experiences, level of satisfaction and expectations of customers in relation to your brand. This allows you to take direct action, identify promoters and detractors, and make improvements based on feedback.

Net Promoter Score (NPS) surveys and customer surveys are two of the most common ways of measuring customer feedback. Gathering all the information is the simple part of the process, but then comes the analysis. When faced with a large number of reviews, categorizing all the responses manually can be a daunting task. Besides taking a lot of time, it can be hard to follow consistent criteria for tagging. Fortunately, **topic analysis enables teams to automatically process all the data generated by surveys**.

Analyzing NPS responses

NPS surveys consist of two different questions designed to evaluate customer satisfaction. The first one is the standardized question: *“How likely are you to recommend this app to a friend?”*. The customer has to give a score from 0 to 10 and, depending on the answer, the customer will be classified as promoter (9 or 10), passive (7, 8) or detractor (6 or below).

The second part is an open-ended question that asks the customer to provide some feedback to support their previous answer. In this answer lies the most significant data. Imagine that a customer scored six on the NPS survey and then added: *"The product is really great, I love the UX as its really easy to use. The bad thing about the service is the pricing, it's quite expensive"*. That's the answer that can actually provide you with better insights.

As mentioned before, the problem comes when you need to process all those open-ended answers manually. So, how can we make this easier by using machine learning? Let's take a look at this real use case:

Retently (https://www.retently.com/) used MonkeyLearn to analyze NPS responses (https://monkeylearn.com/blog/how-retently-automated-customer-feedback-analysis-using-monkeylearn/). They created a topic classifier (http://help.monkeylearn.com/text-classification/custom-classifiers/how-to-build-a-custom-classifier) and trained it to tag each response with different topics like `Product UX`, `Customer Support` and `Ease of Use`. Then, they grouped the Promoters, Passives, and Detractors to determine the most prevalent tags in each group's feedback. The result looked like this:

Combining topic analysis with sentiment analysis (https://monkeylearn.com/sentiment-analysis) and keyword extraction is a powerful approach that enables you to see beyond the NPS score and really understand how your customers feel about your product, and what aspects they appreciate or criticize.

Analyzing Customer Surveys

Perhaps your company is using a different customer survey system. Whether it's emails or online surveys, if you have lots of open-ended questions to tag you may want to use machine learning! Forget about time-consuming manual tasks and get results in a fast and simple way.

Business Intelligence

This is the era of data. Collecting and analyzing data from different sources (the so-called 'Business Intelligence') creates unique opportunities for businesses. By taking advantage of insightful and actionable information, companies are able to improve their decision-making processes, stand out from their competitors, identify trends and spot problems before they escalate.

When it comes to market research and competitive analysis, artificial intelligence (AI) can come to the rescue! You can use topic analysis to sift through product reviews of your brand and compare them with those that mention your competition.

By identifying the recurrent themes or topics in a set of data, you can **obtain valuable insights about what's important to your customers**. Once you've done that, you can also run an aspect-based sentiment analysis to add an extra layer of analysis and gain a deeper understanding about how customers feel about each of those topics.

At MonkeyLearn, we analyzed millions of reviews from TripAdvisor to understand how people feel about hotels in different cities of the world (https://monkeylearn.com/blog/machine-learning-hotel-reviews-insights/). To do that, we used three different machine learning models:

- A topic classifier: to identify if a particular review was referring to food, internet, location, cleanliness, comfort, and facilities, etc

- A sentiment analysis classifier: to get interesting insights on what people love and hate
- A keyword extractor: to collect precise information about the most relevant words mentioned in the reviews

By combining the results of these analyses, we obtained the following exciting insights:

- London hotels have the worst reviews
- London is dirtier than New York and has the worst food overall
- Internet is always an issue
- Hotels with three stars provide the best 'Value for Money'
- Bangkok has a cockroach problem
- Croissants are a huge disappointment for hotels in Paris

Sales and Marketing

Lead qualification (https://blog.hubspot.com/sales/ultimate-guide-to-sales-qualification) is one of the top challenges for sales teams. Only the leads that fit your ideal buyer persona can be qualified as good prospects for your product or service, and identifying them often requires tons of tedious research and manual tasks. What if you could use topic analysis to automate some parts of this task, making it easier and more effective?

Xeneta (https://medium.com/xeneta/boosting-sales-with-machine-learning-fbcf2e618be3), a company that provides sea freight market intelligence, is doing exactly that. Machine learning is helping them to predict the quality of their leads based on company descriptions. Basically, they trained an algorithm to do a job that used to take them hours of manual processing.

Productivity influencer Ari Meisel (https://blog.goodaudience.com/how-to-teach-a-machine-learning-algorithm-to-identify-your-customers-8e5ba38e01bd) is also using machine learning to identify potential customers. He trained a classifier model with MonkeyLearn that analyzes e-mails that are enriched with publicly available data and is able to predict if a contact is a good fit for any of the services he offers.

Another exciting use case of machine learning in sales teams is Drift (https://monkeylearn.com/customers/drift/). On its mission to connect salespeople with the best leads, the company is using MonkeyLearn to automatically filter outbound email responses and manage unsubscribe requests. Thanks to natural language processing, they avoid sending unwanted emails by allowing recipients to opt out based on how they reply. That way, they save their sales team valuable time.

Product Analytics

Product managers are responsible for finding ways to improve their product or make it better. One of their main challenges is to look both at the details and the ‘bigger picture’. When it comes to a customer’s journey for example, product managers should be able to effectively anticipate a customer’s needs or take action based on customer feedback.

Combined with sentiment analysis, topic analysis can be used by product managers to **analyze customer interactions and automatically detect areas for improvement**.

Let’s say you are analyzing data from customer support interactions and you see a spike in the number of people asking how to use a new feature on a SaaS. This may indicate that the explanation about how to use the feature is unclear, and therefore the product team should work on improving the UI/UX – or any documentation about that feature.

Knowledge Management

Organizations generate a huge amount of data every day. In this context, knowledge management (https://www.smartsheet.com/knowledge-management-101) aims to provide the means to capture, store, retrieve and share that data when needed. Topic detection has enormous potential when it comes to analyzing large data sets and extracting the most relevant information out of them.

This could transform industries like healthcare, where tons of complex data is produced every second — and it's expecting to see an explosive growth (https://healthitanalytics.com/news/big-data-to-see-explosive-growth-challenging-healthcare-organizations) in the next few years — but is hard to access it at the right time. Topic analysis makes it possible to classify data based on disease, symptoms, treatments among other aspects. That way, healthcare organizations could access relevant information when needed, and even find patterns and other relevant insights.

Wrap-up

Machine learning is transforming businesses across all industries. We explored the applications of topic analysis in the different areas of a company and presented real use cases and examples of how it's being used to identify the main topics, themes or aspects in a given set of data.

Automatization saves teams valuable time and allows them to focus on the things they do best, without needing to worry about tedious, time-consuming and repetitive tasks.

Analyzing text using machine learning is enabling companies to get insights from data in a way that wasn't possible before. By combining different AI-based techniques it is possible to obtain in-depth knowledge about customers' feelings, and how they perceive and relate to a product or service, just by analyzing the data from online conversations, customer feedback or support tickets. This leads companies to make data-driven decisions, be aware of a potential crisis, and find opportunities for improvement. Also, it provides businesses with valuable information about their competitors, as well as trends in their industry.

By now, you're probably thinking about all the exciting ways topic analysis can help your business. But before you get ahead of yourselves, it's time to get your feet wet! The following section includes different resources to help you get started with topic analysis. You'll find information about relevant tutorials, courses, papers, open-source libraries, and SaaS APIs. Besides, we'll be here to help you get started with MonkeyLearn, by providing a step-by-step tutorial to build a classifier for topic analysis.

Resources

You’re probably eager to get started with topic analysis, but you may not know where to begin. The good news is that there are many useful tools and resources, which are simple to follow and can help you get started. We’ll walk you through some of them in this section.

First, we'll share some topic analysis APIs, including open-source libraries and SaaS APIs. We'll also recommend papers and online courses, so you can learn more about topic detection and hone your skills. Finally, we'll provide you with some tutorials that will help you get hands-on with topic analysis.

Thanks to topic analysis you can simplify many of your team's daily tasks, and save precious time to work on the things that really matter. If you don't fancy building a topic model from scratch, you should take look at what you can do with MonkeyLearn.

Topic Analysis APIs

Implementing the algorithms we discussed before is a great exercise to understand how they work. However, when it's time to actually use them on real-world data, it's usually best to stick to previously implemented solutions.

Open source libraries

If you are going to code yourself, there are a plethora of open source libraries available in many programming languages to do topic analysis. Whether you're using topic modeling or topic classification, here are some useful pointers.

Topic analysis in Python

Python has grown in recent years to become one of the most important languages of the data science community. Easy to use, powerful, and with a great supportive community behind it, Python is ideal for getting started with machine learning and topic analysis.

Gensim (https://radimrehurek.com/gensim/) is the first stop for anything related to topic modeling in Python. It has support for performing both LSA and LDA among other topic modeling algorithms, as well as implementations of the most popular text vectorization algorithms.

NLTK (https://www.nltk.org/) is a library for everything NLP (Natural Language Processing) related. Its main purpose is to process text: cleaning it, splitting paragraphs into sentences, splitting up words in those sentences, and so on. Since it's for working with text in general, it's useful for both topic modeling and classification. This library is not the quickest of its kind (that title probably goes to spaCy (https://spacy.io/), but it's powerful, easy to use, and comes with plenty of tidy corpora to use for training.

Scikit-learn (https://scikit-learn.org/) is a simple library for everything related to machine learning in Python. For topic classification it's a great place to start: from cleaning datasets and training models, to measuring results, scikit-learn provides tools to do it all. And, since it's built on top of NumPy (https://www.numpy.org/) and SciPy (https://www.scipy.org/), it's pretty fast as well.

Topic analysis in R

R is a language that is popular with the statistics crowd right now. Since there are a lot of statistics involved in topic analysis, it's only natural to use R to solve stat-based problems.

For topic modeling, try the topicmodels (https://cran.r-project.org/web/packages/topicmodels/index.html) package. It's part of the tidytext family, which uses tidy data principles to make text analysis tasks easier and more effective. Its usage is nicely described here (https://cran.r-project.org/web/packages/tidytext/vignettes/topic_modeling.html).

Caret (https://topepo.github.io/caret/index.html) is a great package for doing machine learning in R. This package offers a simple API for trying out different algorithms and includes other features for topic classification such as pre-processing, feature selection, and model tuning.

mlR (https://mlr.mlr-org.com/) (short for *machine learning in R*) is an alternative R package also for training and evaluating machine learning models.

SaaS APIs

For a variety of reasons, you may decide that coding your solution on your own is not for you. Maybe you don't want the hassle of an internal machine learning project on your hands, or your issue isn't sensitive enough to justify hiring somebody to take care of a custom solution. Or, you just don't have the resources to develop and maintain in-house machine learning models.

That's alright.

There are people out there who now provide those machine learning solutions as services. These services are usually simpler to use than rolling out a custom implementation with an open source library on your own. It doesn't require hiring in-house machine learning specialists to deal create machine learning algorithms, or administer hardware to run them.

Instead, you just configure what you want from these services, usually from a graphic user interface, and integrate with your existing codebase and data. The only development required is integrating simple APIs with SDKs for popular programming languages, which shouldn't be much of an issue compared with custom implementations.

Some of these solutions are (and you can try them out for free):

- MonkeyLearn
- Amazon Comprehend
- IBM Watson
- Google Cloud NLP
- Aylien
- MeaningCloud
- BigML

Check them out if you are interested in doing topic analysis but don't have the time, experience or resources to do it yourself.

Papers About Topic Modeling and Topic Classification

There's a lot of existing literature on these topics, waiting to be read.

If you would like to learn more about the finer details of how topic analysis works the following papers are a good starting point:

- An Introduction to Latent Semantic Analysis (http://lsa.colorado.edu/papers/dp1.LSAintro.pdf) (Landauer, Foltz and Laham, D., 1998)
- Indexing by Latent Semantic Analysis (http://www.psychology.uwo.ca/faculty/harshman/latentsa.pdf) (Deerwester et al., 1990)
- Latent Dirichlet Allocation (http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf) (Blei, Ng and Jordan, 2003)
- An empirical study of the naive Bayes classifier (https://www.cc.gatech.edu/~isbell/reading/papers/Rish.pdf) (Rish, 2001)

- Text categorization with Support Vector Machines: Learning with many relevant features (https://link.springer.com/content/pdf/10.1007%2FBFb0026683.pdf) (Joachims, 1998)

Courses and Lectures

There are online courses for all students at different stages of their topic analysis journey.

For an explanation of topic modeling in general, check out this lecture (https://www.coursera.org/lecture/python-text-mining/topic-modeling-KiiBl) from the University of Michigan's text mining course in Coursera. Here, you can also find this lecture (https://www.coursera.org/lecture/python-text-mining/text-classification-H05Dd) covering text classification. Plus, this course (https://www.udemy.com/natural-language-processing-with-python-and-nltk/) at Udemy covers NLP in general and several aspects of topic modeling as well.

If you are looking for lectures on some of the algorithms covered in this piece, check out:

- Naive Bayes (https://www.youtube.com/watch?v=z5UQyCESW64), explained by Andrew Ng.
- Support Vector Machines (https://www.youtube.com/watch?v=N1vOgolbjSc), a visual explanation with sample code.
- Deep Learning (https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi) explained simply in four parts. This series is amazing for getting a sense of the idea behind deep learning.

- Latent Dirichlet Allocation (https://www.youtube.com/watch?v=3mHy4OSyRf0) explained in a simple and understandable way. For a more in-depth dive, try this lecture (https://www.youtube.com/watch?v=FkckgwMHP2s) by David Blei, author of the seminal LDA paper.

Now, if what you're interested in is a pro-level course in machine learning, Stanford cs229 (http://cs229.stanford.edu/) is a must. It's an advanced course for computer science students, so it's rated M for Math (which is great if that's what you're into). All the material is available online for free, so you can just hop in and check it out at your own pace.

Tutorials

By now, you probably have some solid knowledge about topic analysis. But, like with everything, there’s no better way to learn than by practicing. With this in mind, here are some useful tutorials and resources for you to get started with topic analysis and build your first model.

This section is split into two different parts. First, we’ll provide step-by-step tutorials to build topic analysis models using Python (with Gensim and NLTK). We’ll also share some useful tutorials to build a model with R.

Then, we’ll show you how to build a classifier for topic analysis using MonkeyLearn.

Tutorials Using Open Source Libraries

Topic Classification in Python

For this, we will cover a simple example of creating a text classifier using NLTK (https://www.nltk.org/) and scikit-learn (https://scikit-learn.org/).

```
pip install nltk
pip install sklearn
```

Download a CSV with sample data for this classifier here (https://monkeylearn.com/blog/wp-content/uploads/2019/04/reviews.csv).

```
import csv
reviews = [row for row in csv.reader(open('reviews.csv'))]
```

This is a list of lists, representing the rows and columns of the CSV file. Every row in the CSV has three columns: the text, the sentiment for that text (we won't use that one in this tutorial) and the topic:

```
[
 ['Text', 'Sentiment', 'Topic'],
 ['Room safe did not work.', 'negative', 'Facilities'],
 ['Mattress very comfortable.', 'positive', 'Comfort'],
 ['No bathroom in room', 'negative', 'Facilities'],
 ...
]
```

First, let's do a bit of processing and cleaning of the data. When doing NLP, text cleanup and processing is a very important first step. Good models cannot be trained with dirty data.

For this purpose, we define a function to do the processing using NLTK. An important feature of this library is that it comes with several corpora ready to use. Since they can get pretty big, they aren't included with the library; they have to be downloaded with the `download` function (https://www.nltk.org/data.html).

```
import re
import nltk

# We need this dataset in order to use the tokenizer
nltk.download('punkt')
from nltk.tokenize import word_tokenize

# Also download the list of stopwords to filter out
nltk.download('stopwords')
from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

def process_text(text):
    # Make all the strings lowercase and remove non alphabetic characters
    text = re.sub('[^A-Za-z]', ' ', text.lower())

    # Tokenize the text; this is, separate every sentence into a list of words
    # Since the text is already split into sentences you don't have to call sent_t
okenize
    tokenized_text = word_tokenize(text)

    # Remove the stopwords and stem each word to its root
    clean_text = [
        stemmer.stem(word) for word in tokenized_text
        if word not in stopwords.words('english')
    ]

    # Remember, this final output is a list of words
    return clean_text
```

Now that we have that function ready, process the data:

```
# Remove the first row, since it only has the labels
reviews = reviews[1:]

texts = [row[0] for row in reviews]
topics = [row[2] for row in reviews]

# Process the texts to so they are ready for training
# But transform the list of words back to string format to feed it to sklearn
texts = [" ".join(process_text(text)) for text in texts]
```

The sentences turned into list of stemmed words without any connectors, which is what we need to feed the algorithm. `texts` looks like this now:

```
['room extrem small practic bed',
 'room safe work',
 'mattress comfort',
 'uncomfort thin mattress plastic cover rustl everi time move',
 'bathroom room',
 ...
]
```

With the cleanup out of the way, we are ready to start training. First, the texts must be vectorized, this is, transformed into numbers that we can feed to the machine learning algorithm.

We do this using scikit-learn.

```
from sklearn.feature_extraction.text import CountVectorizer
matrix = CountVectorizer(max_features=1000)
vectors = matrix.fit_transform(texts).toarray()
```

Now, we separate our training data and our test data, in order to obtain performance metrics.

```
from sklearn.model_selection import train_test_split
vectors_train, vectors_test, topics_train, topics_test = train_test_split(vectors,
topics)
```

Finally, we train a Naive Bayes classifier with the training set and test the model using the testing set.

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(vectors_train, topics_train)

# Predict with the testing set
topics_pred = classifier.predict(vectors_test)

# ...and measure the accuracy of the results
from sklearn.metrics import classification_report
print(classification_report(topics_test, topics_pred))
```

This outputs something like this:

```
             precision    recall  f1-score   support

Cleanliness       0.43      0.43      0.43         7
    Comfort       0.52      0.57      0.54        23
 Facilities       0.55      0.50      0.52        22

avg / total       0.52      0.52      0.52        52
```

It's not a stellar performance, but considering the size of the dataset it's not bad. If you don't know what *precision*, *recall*, and *f1-score* are, they're explained in the Metrics and Evaluation section. *Support* for a category is simply how many samples there were in that category.

From here, the model can be tweaked and tested again in order to get better results. A good starting point is the parameters of the `CountVectorizer`.

Of course, this is a very simple example, but it illustrates all the steps required for building a classifier: obtain the data, clean it, process it, train a model, and iterate.

Using this same process you can also train a classifier for sentiment analysis (https://monkeylearn.com/sentiment-analysis/), with the sentiment tags included in the dataset that we didn't use in this tutorial.

Topic Modeling in Python

For topic modeling we will use Gensim (https://radimrehurek.com/gensim/).

We'll be building on the preprocessing done on the previous tutorial, so we just need to worry about getting Gensim up and running:

```
pip install gensim
```

We pick up halfway through the classifier tutorial. We leave our text as a list of words, since Gensim accepts that as input. Then, we create a Gensim dictionary from the data using the *bag of words* model:

```
from gensim import corpora
texts = [process_text(text) for text in texts]
dictionary = corpora.Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
```

After that, we're ready to go. It's important to note that here we're just using the review texts, and not the topics that come with the dataset. Using this dictionary, we train an LDA model, instructing Gensim to find three topics in the data:

```
from gensim import models
model = models.ldamodel.LdaModel(corpus, num_topics=3, id2word=dictionary, passes=
15)

topics = model.print_topics(num_words=3)
for topic in topics:
    print(topic)
```

And that's it! The code will print out the mixture of the most representative words for three topics:

```
(0, '0.034*"room" + 0.021*"bathroom" + 0.013*"night"')
(1, '0.056*"bed" + 0.043*"room" + 0.024*"comfort"')
(2, '0.059*"room" + 0.033*"clean" + 0.023*"shower"')
```

Interestingly, the algorithm identified words that look a lot like keywords for our original `Facilities`, `Comfort` and `Cleanliness` topics.

Since this is a toy example with few texts (and we know their topic) it isn't very useful, but this example illustrates the basics of how to do topic modeling using Gensim.

Topic Modeling in R

If you want to do topic modeling in R, we urge you to try out the *Tidy Topic Modeling* tutorial (https://cran.r-project.org/web/packages/tidytext/vignettes/topic_modeling.html) for the topicmodels package (https://cran.r-project.org/package=topicmodels). It's straightforward and explains the basics for doing topic modeling using R.

Creating a Topic Classifier with MonkeyLearn

With MonkeyLearn (https://monkeylearn.com/), you can build, train and use classifiers for topic detection in a very simple way. This platform is great for you if you don't want to invest many hours learning about machine learning, or you don't have any coding skills.

To get started, you just have to sign up for free (https://app.monkeylearn.com/accounts/register/) and follow these steps:

1. Create a new classifier

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

2. Select how you want to classify your data

In this case, choose “Topic Classification”. This will allow you to create a fully customized model that classifies texts based on topic, aspect or relevance.

3. Import your training data

Select and upload the data that you will use to train your model. Keep in mind that classifiers learn and get smarter from examples.

You can import training data from different sources, from CSV or Excel files, to third-party apps such as Twitter, Gmail, Zendesk, or RSS feeds.

4. Define the tags for your classifier

The next step is to define the tags that you will use to train your topic classifier. You will need to define at least two tags to get started. You can always add more tags later.

Once you've trained your classifier, it will start using the tags you defined to automatically categorize your data.

Some tips on defining tags:

- For best results, machine learning models need to start simple. Therefore, it's recommended that you start with less than 10 tags to begin training your model. You can add more complexity to your model later.

- Familiarize yourself with your data to get an idea of the tags you should add to your model.
- Start with broad tags and only include those for which you have enough examples.
- Avoid using overlapping or ambiguous tags that may affect the accuracy of your topic classifier.

5. Tag data to train your classifier

Start training your topic classifier by choosing tags for each example:

After tagging some text samples, the classifier will start making predictions when faced with similar text patterns. If you are not satisfied with the results, or you find that your model is not accurate enough, you'll have to tag more examples to continue training your model.

The more data you tag, the smarter your model will be.

6. Test the classifier

After training your classifier, you can test it to see how it works by clicking on "Run" > "Demo". Here, you can write your own text and see how your model classifies the new data:

If you're unhappy with the results, keep training your model by going back to "Build".

MonkeyLearn also provides tools that can help you understand how well your model is working. The “Stats” section allows to you take a sneak peek at the statistics (http://help.monkeylearn.com/text-classification/custom-classifiers/understanding-classifier-statistics) for each tag, like precision and recall, as well as a keyword cloud (http://help.monkeylearn.com/text-classification/custom-classifiers/using-keywords-to-increase-classifier-performance) with the most common terms and expressions for each tag. In this section, you will also find some clues to improve the accuracy of your model. You can add more training examples if needed, or you can go through the ‘false positives’ or ‘false negatives’ and retag them. This way, your model can learn from tags that were labeled incorrectly.

7. Integrate the topic classifier

You’ve trained your model so that it can make accurate predictions on the different topics of a given text. Now it’s time to use it to classify new data! There are three different ways to do this with MonkeyLearn:

Batch processing: to classify text in a batch (http://help.monkeylearn.com/analyzing-text-and-processing/batch-processing-of-text-analysis), go to “Run” > “Batch” and upload a CSV or Excel file. After uploading the file, the classifier will analyze the data and send you a new file with the same data plus the predictions.

API: you can use MonkeyLearn API (https://monkeylearn.com/api/v3/) to classify new data programmatically. Integrations: work with the tools you use every day, which are used to automatically import new text data into your classifier for an automated analysis. Integrations, such as Google Sheets, Zapier, Rapidminer, and Zendesk, can be used without having to type a single line of code:

Parting words

Topic analysis makes it possible to detect different topics within a large set of text data in a fast and simple way. By using this incredibly powerful tool, you can automate many processes in different areas of your business, from customer service to social media monitoring. Thanks to topic analysis, you can accomplish complex tasks more effectively. Also, you can obtain valuable insights from data that will ultimately lead to better business decisions.

We’ve come a long way since first learning about topic analysis, how it works, what are some of its applications, and the different resources and tutorials you can follow to get started with topic analysis. Now, you’re all set to take that next step and start working with topic detection!

As we mentioned above, creating a customized topic classifier with machine learning can be very simple. Sign up for free (https://app.monkeylearn.com/accounts/register/) to MonkeyLearn and start building your first model. Got any questions or need help? Contact us (https://monkeylearn.com/contact/) and we’ll be glad to help you get started with topic analysis.

Start using Topic Analysis today!

Automate business processes and save hours of manual data processing.

SIGN UP FREE

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

Schedule a Demo

(/)

RESOURCES

Pricing (/pricing)
Help (http://help.monkeylearn.com)
API Reference (/api)
Blog (/blog)

GUIDES

Sentiment Analysis (/sentiment-analysis)
Text Classification (/text-classification/)
Text Analysis (/text-analysis/)
Topic Analysis (/topic-analysis/)

COMPANY

About (/about)
Careers (we're hiring!) (/jc
Twitter (https://twitter.com/monl
Github (https://github.com/monl

Keyword
Extraction
(/keyword-
extraction/)
Natural
Language
Processing
(/natural-
language-
processing)

(https://tw