**Automatic summarization**

**Automatic summarization** is the process of shortening a set of data computationally, to create a subset (a [summary](https://en.wikipedia.org/wiki/Abstract_(summary))) that represents the most important or relevant information within the original content. [Artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) [algorithms](https://en.wikipedia.org/wiki/Algorithm) are commonly developed and employed to achieve this, specialized for different types of data.

[Text](https://en.wikipedia.org/wiki/Plain_text) summarization is usually implemented by [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) methods, designed to locate the most informative sentences in a given document.[[1]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-Torres2014-1) On the other hand, visual content can be summarized using [computer vision](https://en.wikipedia.org/wiki/Computer_vision) algorithms. [Image](https://en.wikipedia.org/wiki/Image) summarization is the subject of ongoing research; existing approaches typically attempt to display the most representative images from a given image collection, or generate a video that only includes the most important content from the entire collection.[[2]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-3)[[4]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-4) Video summarization algorithms identify and extract from the original video content the most important frames (*key-frames*), and/or the most important video segments (*key-shots*), normally in a temporally ordered fashion.[[5]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-PalPetrosino2012-5)[[6]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-Elhamifar2012-6)[[7]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-Mademlis2016-7)[[8]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-Mademlis2018-8) Video summaries simply retain a carefully selected subset of the original video frames and, therefore, are not identical to the output of [video synopsis](https://en.wikipedia.org/wiki/Video_synopsis) algorithms, where *new* video frames are being synthesized based on the original video content.

Commercial products[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=1)]

In 2022 [Google Docs](https://en.wikipedia.org/wiki/Google_Docs) released an automatic summarization feature.[[9]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-9)

Approaches[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=2)]

There are two general approaches to automatic summarization: [extraction](https://en.wikipedia.org/wiki/Information_extraction) and [abstraction](https://en.wikipedia.org/wiki/Abstract_(summary)).

**Extraction-based summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=3)]

Here, content is extracted from the original data, but the extracted content is not modified in any way. Examples of extracted content include key-phrases that can be used to "tag" or index a text document, or key sentences (including headings) that collectively comprise an abstract, and representative images or video segments, as stated above. For text, extraction is analogous to the process of skimming, where the summary (if available), headings and subheadings, figures, the first and last paragraphs of a section, and optionally the first and last sentences in a paragraph are read before one chooses to read the entire document in detail.[[10]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-10) Other examples of extraction that include key sequences of text in terms of clinical relevance (including patient/problem, intervention, and outcome).[[11]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-Afzal_et_al-11)

**Abstractive-based summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=4)]

Abstractive summarization methods generate new text that did not exist in the original text.[[12]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-12) This has been applied mainly for text. Abstractive methods build an internal semantic representation of the original content (often called a language model), and then use this representation to create a summary that is closer to what a human might express. Abstraction may transform the extracted content by [paraphrasing](https://en.wikipedia.org/wiki/Automated_paraphrasing) sections of the source document, to condense a text more strongly than extraction. Such transformation, however, is computationally much more challenging than extraction, involving both [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) and often a deep understanding of the domain of the original text in cases where the original document relates to a special field of knowledge. "Paraphrasing" is even more difficult to apply to images and videos, which is why most summarization systems are extractive.

**Aided summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=5)]

Approaches aimed at higher summarization quality rely on combined software and human effort. In Machine Aided Human Summarization, extractive techniques highlight candidate passages for inclusion (to which the human adds or removes text). In Human Aided Machine Summarization, a human post-processes software output, in the same way that one edits the output of automatic translation by Google Translate.

Applications and systems for summarization[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=6)]

There are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is *generic summarization*, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.). The second is *query relevant summarization*, sometimes called *query-based summarization*, which summarizes objects specific to a query. Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.

An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a [cluster](https://en.wikipedia.org/wiki/Cluster_analysis) of articles on the same topic). This problem is called [multi-document summarization](https://en.wikipedia.org/wiki/Multi-document_summarization). A related application is summarizing news articles. Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.

Image collection summarization is another application example of automatic summarization. It consists in selecting a representative set of images from a larger set of images.[[13]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-13) A summary in this context is useful to show the most representative images of results in an [image collection exploration](https://en.wikipedia.org/wiki/Image_collection_exploration) system. Video summarization is a related domain, where the system automatically creates a trailer of a long video. This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions. Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured.

At a very high level, summarization algorithms try to find subsets of objects (like set of sentences, or a set of images), which cover information of the entire set. This is also called the *core-set*. These algorithms model notions like diversity, coverage, information and representativeness of the summary. Query based summarization techniques, additionally model for relevance of the summary with the query. Some techniques and algorithms which naturally model summarization problems are TextRank and PageRank, [Submodular set function](https://en.wikipedia.org/wiki/Submodular_set_function), [Determinantal point process](https://en.wikipedia.org/wiki/Determinantal_point_process), maximal marginal relevance (MMR) etc.

**Keyphrase extraction**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=7)]

The task is the following. You are given a piece of text, such as a journal article, and you must produce a list of keywords or key[phrase]s that capture the primary topics discussed in the text.[[14]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-14) In the case of [research articles](https://en.wikipedia.org/wiki/Research_article), many authors provide manually assigned keywords, but most text lacks pre-existing keyphrases. For example, news articles rarely have keyphrases attached, but it would be useful to be able to automatically do so for a number of applications discussed below. Consider the example text from a news article:

"The Army Corps of Engineers, rushing to meet President Bush's promise to protect New Orleans by the start of the 2006 hurricane season, installed defective flood-control pumps last year despite warnings from its own expert that the equipment would fail during a storm, according to documents obtained by The Associated Press".

A keyphrase extractor might select "Army Corps of Engineers", "President Bush", "New Orleans", and "defective flood-control pumps" as keyphrases. These are pulled directly from the text. In contrast, an abstractive keyphrase system would somehow internalize the content and generate keyphrases that do not appear in the text, but more closely resemble what a human might produce, such as "political negligence" or "inadequate protection from floods". Abstraction requires a deep [understanding of the text](https://en.wikipedia.org/wiki/Natural-language_understanding), which makes it difficult for a computer system. Keyphrases have many applications. They can enable document browsing by providing a short summary, improve [information retrieval](https://en.wikipedia.org/wiki/Information_retrieval) (if documents have keyphrases assigned, a user could search by keyphrase to produce more reliable hits than a [full-text search](https://en.wikipedia.org/wiki/Full-text_search)), and be employed in generating index entries for a large text corpus.

Depending on the different literature and the definition of key terms, words or phrases, [keyword extraction](https://en.wikipedia.org/wiki/Keyword_extraction) is a highly related theme.

**Supervised learning approaches**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=8)]

Beginning with the work of Turney,[[15]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-15) many researchers have approached keyphrase extraction as a [supervised machine learning](https://en.wikipedia.org/wiki/Supervised_machine_learning) problem. Given a document, we construct an example for each [unigram](https://en.wikipedia.org/wiki/Unigram), [bigram](https://en.wikipedia.org/wiki/Bigram), and trigram found in the text (though other text units are also possible, as discussed below). We then compute various features describing each example (e.g., does the phrase begin with an upper-case letter?). We assume there are known keyphrases available for a set of training documents. Using the known keyphrases, we can assign positive or negative labels to the examples. Then we learn a classifier that can discriminate between positive and negative examples as a function of the features. Some classifiers make a [binary classification](https://en.wikipedia.org/wiki/Binary_classification) for a test example, while others assign a probability of being a keyphrase. For instance, in the above text, we might learn a rule that says phrases with initial capital letters are likely to be keyphrases. After training a learner, we can select keyphrases for test documents in the following manner. We apply the same example-generation strategy to the test documents, then run each example through the learner. We can determine the keyphrases by looking at binary classification decisions or probabilities returned from our learned model. If probabilities are given, a threshold is used to select the keyphrases. Keyphrase extractors are generally evaluated using [precision and recall](https://en.wikipedia.org/wiki/Precision_and_recall). Precision measures how many of the proposed keyphrases are actually correct. Recall measures how many of the true keyphrases your system proposed. The two measures can be combined in an F-score, which is the harmonic mean of the two (*F* = 2*PR*/(*P* + *R*) ). Matches between the proposed keyphrases and the known keyphrases can be checked after stemming or applying some other text normalization.

Designing a supervised keyphrase extraction system involves deciding on several choices (some of these apply to unsupervised, too). The first choice is exactly how to generate examples. Turney and others have used all possible unigrams, bigrams, and trigrams without intervening punctuation and after removing stopwords. Hulth showed that you can get some improvement by selecting examples to be sequences of tokens that match certain patterns of part-of-speech tags. Ideally, the mechanism for generating examples produces all the known labeled keyphrases as candidates, though this is often not the case. For example, if we use only unigrams, bigrams, and trigrams, then we will never be able to extract a known keyphrase containing four words. Thus, recall may suffer. However, generating too many examples can also lead to low precision.

We also need to create features that describe the examples and are informative enough to allow a learning algorithm to discriminate keyphrases from non- keyphrases. Typically features involve various term frequencies (how many times a phrase appears in the current text or in a larger corpus), the length of the example, relative position of the first occurrence, various boolean syntactic features (e.g., contains all caps), etc. The Turney paper used about 12 such features. Hulth uses a reduced set of features, which were found most successful in the KEA (Keyphrase Extraction Algorithm) work derived from Turney's seminal paper.

In the end, the system will need to return a list of keyphrases for a test document, so we need to have a way to limit the number. Ensemble methods (i.e., using votes from several classifiers) have been used to produce numeric scores that can be thresholded to provide a user-provided number of keyphrases. This is the technique used by Turney with C4.5 decision trees. Hulth used a single binary classifier so the learning algorithm implicitly determines the appropriate number.

Once examples and features are created, we need a way to learn to predict keyphrases. Virtually any supervised learning algorithm could be used, such as decision trees, [Naive Bayes](https://en.wikipedia.org/wiki/Naive_Bayes), and rule induction. In the case of Turney's GenEx algorithm, a [genetic algorithm](https://en.wikipedia.org/wiki/Genetic_algorithm) is used to learn parameters for a domain-specific keyphrase extraction algorithm. The extractor follows a series of heuristics to identify keyphrases. The genetic algorithm optimizes parameters for these heuristics with respect to performance on training documents with known key phrases.

**Unsupervised approach: TextRank**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=9)]

Another keyphrase extraction algorithm is TextRank. While supervised methods have some nice properties, like being able to produce interpretable rules for what features characterize a keyphrase, they also require a large amount of [training data](https://en.wikipedia.org/wiki/Training_set). Many documents with known keyphrases are needed. Furthermore, training on a specific domain tends to customize the extraction process to that domain, so the resulting classifier is not necessarily portable, as some of Turney's results demonstrate. Unsupervised keyphrase extraction removes the need for training data. It approaches the problem from a different angle. Instead of trying to learn explicit features that characterize keyphrases, the TextRank algorithm[[16]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-16) exploits the structure of the text itself to determine keyphrases that appear "central" to the text in the same way that [PageRank](https://en.wikipedia.org/wiki/PageRank) selects important Web pages. Recall this is based on the notion of "prestige" or "recommendation" from [social networks](https://en.wikipedia.org/wiki/Social_network). In this way, TextRank does not rely on any previous training data at all, but rather can be run on any arbitrary piece of text, and it can produce output simply based on the text's intrinsic properties. Thus the algorithm is easily portable to new domains and languages.

TextRank is a general purpose [graph](https://en.wikipedia.org/wiki/Graph_(abstract_data_type))-based ranking algorithm for [NLP](https://en.wikipedia.org/wiki/Natural_language_processing). Essentially, it runs PageRank on a graph specially designed for a particular NLP task. For keyphrase extraction, it builds a graph using some set of text units as vertices. Edges are based on some measure of semantic or [lexical](https://en.wikipedia.org/wiki/Lexical_(semiotics)) [similarity](https://en.wikipedia.org/wiki/Semantic_similarity) between the text unit vertices. Unlike PageRank, the edges are typically undirected and can be weighted to reflect a degree of similarity. Once the graph is constructed, it is used to form a stochastic matrix, combined with a damping factor (as in the "random surfer model"), and the ranking over vertices is obtained by finding the eigenvector corresponding to [eigenvalue](https://en.wikipedia.org/wiki/Eigenvalue) 1 (i.e., the [stationary distribution](https://en.wikipedia.org/wiki/Stationary_distribution) of the [random walk](https://en.wikipedia.org/wiki/Random_walk) on the graph).

The vertices should correspond to what we want to rank. Potentially, we could do something similar to the supervised methods and create a vertex for each unigram, bigram, trigram, etc. However, to keep the graph small, the authors decide to rank individual unigrams in a first step, and then include a second step that merges highly ranked adjacent unigrams to form multi-word phrases. This has a nice side effect of allowing us to produce keyphrases of arbitrary length. For example, if we rank unigrams and find that "advanced", "natural", "language", and "processing" all get high ranks, then we would look at the original text and see that these words appear consecutively and create a final keyphrase using all four together. Note that the unigrams placed in the graph can be filtered by part of speech. The authors found that adjectives and nouns were the best to include. Thus, some linguistic knowledge comes into play in this step.

Edges are created based on word [co-occurrence](https://en.wikipedia.org/wiki/Co-occurrence) in this application of TextRank. Two vertices are connected by an edge if the [unigrams](https://en.wikipedia.org/wiki/Unigram) appear within a window of size N in the original text. N is typically around 2–10. Thus, "natural" and "language" might be linked in a text about NLP. "Natural" and "processing" would also be linked because they would both appear in the same string of N words. These edges build on the notion of "text [cohesion](https://en.wikipedia.org/wiki/Cohesion_(linguistics))" and the idea that words that appear near each other are likely related in a meaningful way and "recommend" each other to the reader.

Since this method simply ranks the individual vertices, we need a way to threshold or produce a limited number of keyphrases. The technique chosen is to set a count T to be a user-specified fraction of the total number of vertices in the graph. Then the top T vertices/unigrams are selected based on their stationary probabilities. A post- processing step is then applied to merge adjacent instances of these T unigrams. As a result, potentially more or less than T final keyphrases will be produced, but the number should be roughly proportional to the length of the original text.

It is not initially clear why applying PageRank to a co-occurrence graph would produce useful keyphrases. One way to think about it is the following. A word that appears multiple times throughout a text may have many different co-occurring neighbors. For example, in a text about machine learning, the unigram "learning" might co-occur with "machine", "supervised", "un-supervised", and "semi-supervised" in four different sentences. Thus, the "learning" vertex would be a central "hub" that connects to these other modifying words. Running PageRank/TextRank on the graph is likely to rank "learning" highly. Similarly, if the text contains the phrase "supervised classification", then there would be an edge between "supervised" and "classification". If "classification" appears several other places and thus has many neighbors, its importance would contribute to the importance of "supervised". If it ends up with a high rank, it will be selected as one of the top T unigrams, along with "learning" and probably "classification". In the final post-processing step, we would then end up with keyphrases "supervised learning" and "supervised classification".

In short, the co-occurrence graph will contain densely connected regions for terms that appear often and in different contexts. A random walk on this graph will have a stationary distribution that assigns large probabilities to the terms in the centers of the clusters. This is similar to densely connected Web pages getting ranked highly by PageRank. This approach has also been used in document summarization, considered below.

**Document summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=10)]

Like keyphrase extraction, document summarization aims to identify the essence of a text. The only real difference is that now we are dealing with larger text units—whole sentences instead of words and phrases.

**Supervised learning approaches**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=11)]

Supervised text summarization is very much like supervised keyphrase extraction. Basically, if you have a collection of documents and human-generated summaries for them, you can learn features of sentences that make them good candidates for inclusion in the summary. Features might include the position in the document (i.e., the first few sentences are probably important), the number of words in the sentence, etc. The main difficulty in supervised extractive summarization is that the known summaries must be manually created by extracting sentences so the sentences in an original training document can be labeled as "in summary" or "not in summary". This is not typically how people create summaries, so simply using journal abstracts or existing summaries is usually not sufficient. The sentences in these summaries do not necessarily match up with sentences in the original text, so it would be difficult to assign labels to examples for training. Note, however, that these natural summaries can still be used for evaluation purposes, since ROUGE-1 evaluation only considers unigrams.

**Maximum entropy-based summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=12)]

During the DUC 2001 and 2002 evaluation workshops, [TNO](https://en.wikipedia.org/wiki/Netherlands_Organisation_for_Applied_Scientific_Research) developed a sentence extraction system for multi-document summarization in the news domain. The system was based on a hybrid system using a [Naive Bayes classifier](https://en.wikipedia.org/wiki/Naive_Bayes_classifier) and statistical language models for modeling salience. Although the system exhibited good results, the researchers wanted to explore the effectiveness of a [maximum entropy](https://en.wikipedia.org/wiki/Maximum_entropy_classifier) (ME) classifier for the meeting summarization task, as ME is known to be robust against feature dependencies. Maximum entropy has also been applied successfully for summarization in the broadcast news domain.

**Adaptive summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=13)]

A promising approach is adaptive document/text summarization.[[17]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-17) It involves first recognizing the text genre and then applying summarization algorithms optimized for this genre. Such software has been created.[[18]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-18)

**TextRank and LexRank**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=14)]

The unsupervised approach to summarization is also quite similar in spirit to unsupervised keyphrase extraction and gets around the issue of costly training data. Some unsupervised summarization approaches are based on finding a "[centroid](https://en.wikipedia.org/wiki/Centroid)" sentence, which is the mean word vector of all the sentences in the document. Then the sentences can be ranked with regard to their similarity to this centroid sentence.

A more principled way to estimate sentence importance is using random walks and eigenvector centrality. LexRank[[19]](https://en.wikipedia.org/wiki/Automatic_summarization" \l "cite_note-19) is an algorithm essentially identical to TextRank, and both use this approach for document summarization. The two methods were developed by different groups at the same time, and LexRank simply focused on summarization, but could just as easily be used for keyphrase extraction or any other NLP ranking task.

In both LexRank and TextRank, a graph is constructed by creating a vertex for each sentence in the document.

The edges between sentences are based on some form of semantic similarity or content overlap. While LexRank uses [cosine similarity](https://en.wikipedia.org/wiki/Cosine_similarity) of [TF-IDF](https://en.wikipedia.org/wiki/TF-IDF) vectors, TextRank uses a very similar measure based on the number of words two sentences have in common ([normalized](https://en.wikipedia.org/wiki/Quantile_normalization) by the sentences' lengths). The LexRank paper explored using unweighted edges after applying a threshold to the cosine values, but also experimented with using edges with weights equal to the similarity score. TextRank uses continuous [similarity scores](https://en.wikipedia.org/wiki/Similarity_score) as weights.

In both algorithms, the sentences are ranked by applying PageRank to the resulting graph. A summary is formed by combining the top ranking sentences, using a threshold or length cutoff to limit the size of the summary.

It is worth noting that TextRank was applied to summarization exactly as described here, while LexRank was used as part of a larger summarization system ([MEAD](https://en.wikipedia.org/w/index.php?title=MEAD&action=edit&redlink=1)) that combines the LexRank score (stationary probability) with other features like sentence position and length using a [linear combination](https://en.wikipedia.org/wiki/Linear_combination) with either user-specified or automatically tuned weights. In this case, some training documents might be needed, though the TextRank results show the additional features are not absolutely necessary.

Unlike TextRank, LexRank has been applied to multi-document summarization.

**Multi-document summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=15)]

*Main article:*[*Multi-document summarization*](https://en.wikipedia.org/wiki/Multi-document_summarization)

**Multi-document summarization** is an automatic procedure aimed at extraction of information from multiple texts written about the same topic. Resulting summary report allows individual users, such as professional information consumers, to quickly familiarize themselves with information contained in a large cluster of documents. In such a way, multi-document summarization systems are complementing the [news aggregators](https://en.wikipedia.org/wiki/News_aggregators) performing the next step down the road of coping with [information overload](https://en.wikipedia.org/wiki/Information_overload). Multi-document summarization may also be done in response to a question.[[20]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-20)[[11]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-Afzal_et_al-11)

Multi-document summarization creates information reports that are both concise and comprehensive. With different opinions being put together and outlined, every topic is described from multiple perspectives within a single document. While the goal of a brief summary is to simplify information search and cut the time by pointing to the most relevant source documents, comprehensive multi-document summary should itself contain the required information, hence limiting the need for accessing original files to cases when refinement is required. Automatic summaries present information extracted from multiple sources algorithmically, without any editorial touch or subjective human intervention, thus making it completely unbiased. [[*dubious*](https://en.wikipedia.org/wiki/Wikipedia:Accuracy_dispute#Disputed_statement)*–*[*discuss*](https://en.wikipedia.org/wiki/Talk:Automatic_summarization#Dubious)]

**Diversity**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=16)]

Multi-document extractive summarization faces a problem of redundancy. Ideally, we want to extract sentences that are both "central" (i.e., contain the main ideas) and "diverse" (i.e., they differ from one another). For example, in a set of news articles about some event, each article is likely to have many similar sentences. To address this issue, LexRank applies a heuristic post-processing step that adds sentences in rank order, but discards sentences that are too similar to ones already in the summary. This method is called Cross-Sentence Information Subsumption (CSIS). These methods work based on the idea that sentences "recommend" other similar sentences to the reader. Thus, if one sentence is very similar to many others, it will likely be a sentence of great importance. Its importance also stems from the importance of the sentences "recommending" it. Thus, to get ranked highly and placed in a summary, a sentence must be similar to many sentences that are in turn also similar to many other sentences. This makes intuitive sense and allows the algorithms to be applied to an arbitrary new text. The methods are domain-independent and easily portable. One could imagine the features indicating important sentences in the news domain might vary considerably from the biomedical domain. However, the unsupervised "recommendation"-based approach applies to any domain.

A related method is Maximal Marginal Relevance (MMR),[[21]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-21) which uses a general-purpose graph-based ranking algorithm like Page/Lex/TextRank that handles both "centrality" and "diversity" in a unified mathematical framework based on [absorbing Markov chain](https://en.wikipedia.org/wiki/Absorbing_Markov_chain) random walks (a random walk where certain states end the walk). The algorithm is called GRASSHOPPER.[[22]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-22) In addition to explicitly promoting diversity during the ranking process, GRASSHOPPER incorporates a prior ranking (based on sentence position in the case of summarization).

The state of the art results for multi-document summarization are obtained using mixtures of submodular functions. These methods have achieved the state of the art results for Document Summarization Corpora, DUC 04 - 07.[[23]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-23) Similar results were achieved with the use of determinantal point processes (which are a special case of submodular functions) for DUC-04.[[24]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-24)

A new method for multi-lingual multi-document summarization that avoids redundancy generates ideograms to represent the meaning of each sentence in each document, then evaluates similarity by comparing ideogram shape and position. It does not use word frequency, training or preprocessing. It uses two user-supplied parameters: equivalence (when are two sentences to be considered equivalent?) and relevance (how long is the desired summary?).

**Submodular functions as generic tools for summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=17)]

The idea of a [submodular set function](https://en.wikipedia.org/wiki/Submodular_set_function) has recently emerged as a powerful modeling tool for various summarization problems. Submodular functions naturally model notions of *coverage*, *information*, *representation* and *diversity*. Moreover, several important [combinatorial optimization](https://en.wikipedia.org/wiki/Combinatorial_optimization) problems occur as special instances of submodular optimization. For example, the [set cover problem](https://en.wikipedia.org/wiki/Set_cover_problem) is a special case of submodular optimization, since the set cover function is submodular. The set cover function attempts to find a subset of objects which *cover* a given set of concepts. For example, in document summarization, one would like the summary to cover all important and relevant concepts in the document. This is an instance of set cover. Similarly, the [facility location problem](https://en.wikipedia.org/wiki/Optimal_facility_location) is a special case of submodular functions. The Facility Location function also naturally models coverage and diversity. Another example of a submodular optimization problem is using a [determinantal point process](https://en.wikipedia.org/wiki/Determinantal_point_process) to model diversity. Similarly, the Maximum-Marginal-Relevance procedure can also be seen as an instance of submodular optimization. All these important models encouraging coverage, diversity and information are all submodular. Moreover, submodular functions can be efficiently combined, and the resulting function is still submodular. Hence, one could combine one submodular function which models diversity, another one which models coverage and use human supervision to learn a right model of a submodular function for the problem.

While submodular functions are fitting problems for summarization, they also admit very efficient algorithms for optimization. For example, a simple [greedy algorithm](https://en.wikipedia.org/wiki/Greedy_algorithm) admits a constant factor guarantee.[[25]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-25) Moreover, the greedy algorithm is extremely simple to implement and can scale to large datasets, which is very important for summarization problems.

Submodular functions have achieved state-of-the-art for almost all summarization problems. For example, work by Lin and Bilmes, 2012[[26]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-26) shows that submodular functions achieve the best results to date on DUC-04, DUC-05, DUC-06 and DUC-07 systems for document summarization. Similarly, work by Lin and Bilmes, 2011,[[27]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-27) shows that many existing systems for automatic summarization are instances of submodular functions. This was a breakthrough result establishing submodular functions as the right models for summarization problems.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)]

Submodular Functions have also been used for other summarization tasks. Tschiatschek et al., 2014 show[[28]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-28) that mixtures of submodular functions achieve state-of-the-art results for image collection summarization. Similarly, Bairi et al., 2015[[29]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-29) show the utility of submodular functions for summarizing multi-document topic hierarchies. Submodular Functions have also successfully been used for summarizing machine learning datasets.[[30]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-30)

**Applications**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=18)]

|  |  |
| --- | --- |
| [[icon]](https://en.wikipedia.org/wiki/File:Wiki_letter_w_cropped.svg) | This section **needs expansion**. You can help by [adding to it](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=). *(February 2017)* |

Specific applications of automatic summarization include:

* The [Reddit](https://en.wikipedia.org/wiki/Reddit) [bot](https://en.wikipedia.org/wiki/Internet_bot) "autotldr",[[31]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-31) created in 2011 summarizes news articles in the comment-section of reddit posts. It was found to be very useful by the reddit community which upvoted its summaries hundreds of thousands of times.[[32]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-32) The name is reference to [TL;DR](https://en.wikipedia.org/wiki/TL;DR) − [Internet slang](https://en.wikipedia.org/wiki/Internet_slang) for "too long; didn't read".[[33]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-33)[[34]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-34)
* [Adversarial stylometry](https://en.wikipedia.org/wiki/Adversarial_stylometry) may make use of summaries, if the detail lost is not major and the summary is sufficiently stylistically different to the input.[[35]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-FOOTNOTEPotthastHagenStein201611-12-35)

Evaluation[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=19)]

The most common way to evaluate the informativeness of automatic summaries is to compare them with human-made model summaries.

Evaluation can be intrinsic or extrinsic,[[36]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-36) and inter-textual or intra-textual.[[37]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-37)

**Intrinsic versus extrinsic**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=20)]

Intrinsic evaluation assesses the summaries directly, while extrinsic evaluation evaluates how the summarization system affects the completion of some other task. Intrinsic evaluations have assessed mainly the coherence and informativeness of summaries. Extrinsic evaluations, on the other hand, have tested the impact of summarization on tasks like relevance assessment, reading comprehension, etc.

**Inter-textual versus intra-textual**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=21)]

Intra-textual evaluation assess the output of a specific summarization system, while inter-textual evaluation focuses on contrastive analysis of outputs of several summarization systems.

Human judgement often varies greatly in what it considers a "good" summary, so creating an automatic evaluation process is particularly difficult. Manual evaluation can be used, but this is both time and labor-intensive, as it requires humans to read not only the summaries but also the source documents. Other issues are those concerning [coherence](https://en.wikipedia.org/wiki/Coherence_(linguistics)) and coverage.

The most common way to evaluate summaries is [ROUGE](https://en.wikipedia.org/wiki/ROUGE_(metric)) (Recall-Oriented Understudy for Gisting Evaluation). It is very common for summarization and translation systems in [NIST](https://en.wikipedia.org/wiki/NIST)'s Document Understanding Conferences.[[2]](https://web.archive.org/web/20060408135021/http:/haydn.isi.edu/ROUGE/) ROUGE is a recall-based measure of how well a summary covers the content of human-generated summaries known as references. It calculates [n-gram](https://en.wikipedia.org/wiki/N-gram) overlaps between automatically generated summaries and previously written human summaries. It is recall-based to encourage inclusion of all important topics in summaries. Recall can be computed with respect to unigram, bigram, trigram, or 4-gram matching. For example, ROUGE-1 is the fraction of unigrams that appear in both the reference summary and the automatic summary out of all unigrams in the reference summary. If there are multiple reference summaries, their scores are averaged. A high level of overlap should indicate a high degree of shared concepts between the two summaries.

ROUGE cannot determine if the result is coherent, that is if sentences flow together in a sensibly. High-order n-gram ROUGE measures help to some degree.

Another unsolved problem is [Anaphor resolution](https://en.wikipedia.org/wiki/Anaphora_(linguistics)). Similarly, for image summarization, Tschiatschek et al., developed a Visual-ROUGE score which judges the performance of algorithms for image summarization.[[38]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-38)

**Domain-specific versus domain-independent summarization**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=22)]

Domain-independent summarization techniques apply sets of general features to identify information-rich text segments. Recent research focuses on domain-specific summarization using knowledge specific to the text's domain, such as medical knowledge and ontologies for summarizing medical texts.[[39]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-39)

**Qualitative**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=23)]

The main drawback of the evaluation systems so far is that we need a reference summary (for some methods, more than one), to compare automatic summaries with models. This is a hard and expensive task. Much effort has to be made to create corpora of texts and their corresponding summaries. Furthermore, some methods require manual annotation of the summaries (e.g. SCU in the Pyramid Method). Moreover, they all perform a quantitative evaluation with regard to different similarity metrics.

History[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=24)]

The first publication in the area dates back to 1957 [[40]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-40) ([Hans Peter Luhn](https://en.wikipedia.org/wiki/Hans_Peter_Luhn)), starting with a statistical technique. Research increased significantly in 2015. [Term frequency–inverse document frequency](https://en.wikipedia.org/wiki/Term_frequency%E2%80%93inverse_document_frequency) had been used by 2016. Pattern-based summarization was the most powerful option for multi-document summarization found by 2016. In the following year it was surpassed by [latent semantic analysis](https://en.wikipedia.org/wiki/Latent_semantic_analysis) (LSA) combined with [non-negative matrix factorization](https://en.wikipedia.org/wiki/Non-negative_matrix_factorization) (NMF). Although they did not replace other approaches and are often combined with them, by 2019 machine learning methods dominated the extractive summarization of single documents, which was considered to be nearing maturity. By 2020, the field was still very active and research is shifting towards abstractive summation and real-time summarization.[[41]](https://en.wikipedia.org/wiki/Automatic_summarization#cite_note-41)

**Recent approaches**[[edit](https://en.wikipedia.org/w/index.php?title=Automatic_summarization&action=edit&section=25)]

Recently the rise of [transformer models](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)) replacing more traditional [RNN](https://en.wikipedia.org/wiki/Rnn_(software)) ([LSTM](https://en.wikipedia.org/wiki/LSTM)) have provided a flexibility in the mapping of text sequences to text sequences of a different type, which is well suited to automatic summarization. This includes models such