Semantic Search

Semantic search is a data searching technique that focuses on understanding the contextual meaning and intent behind a user's search query, rather than only matching keywords. Instead of merely looking for literal matches between search queries and indexed content, it aims to deliver more relevant search results by considering various factors, including the relationships between words, the searcher's location, any previous searches, and the context of the search.

Traditional search engines typically focus on matching keywords within a search query to corresponding keywords in indexed web pages. In contrast, semantic search aims to comprehend the deeper meaning and intent behind a user's search, much like a human would. By understanding the meaning and context of words, phrases, and entities within a search query, semantic search strives to deliver highly relevant search results that satisfy the user's information needs.

Imagine searching for "best laptops for graphic design students." A traditional search engine might focus solely on matching those keywords to web pages. Whereas, a semantic search engine would try to understand that you are looking for laptops with specific features like powerful graphics cards, ample RAM, and color-accurate displays. It would then return results that recommend laptops suitable for graphic design tasks.

How does semantic search work?

Semantic search engines employ various techniques from natural language processing (NLP), knowledge representation, and machine learning to understand the semantics of search queries and web content. Here's a breakdown of the process:

- **Query analysis**: The search engine analyzes the user's query to identify keywords, phrases, and entities. It also attempts to interpret the user's search intent by analyzing the relationships between these elements.
- **Knowledge graph integration**: Semantic search engines often leverage knowledge graphs, vast databases containing information about entities and their relationships. This information helps the engine understand the context of the search query.
- **Content analysis**: Similar to how a search engine analyzes queries, it also examines the content of web pages to determine their relevance to a particular search. This analysis goes beyond keyword matching and considers factors such as the overall topic, sentiment, and entities mentioned within the content.
- **Result return and retrieval**: Based on the analysis of the query and the content, the search engine could return web pages according to their relevance and semantic

similarity to the search query. It then retrieves and displays the most relevant results to the user.

Why is semantic search important?

Semantic search is important for several reasons:

- **Improved relevance**: By understanding the meaning behind a search query, especially complex or ambiguous ones, search engines can deliver more relevant results. This means users are more likely to find exactly what they're looking for on the first try.
- **Enhanced user experience**: When search results are highly relevant, users have a more satisfying experience. They can quickly find the information they need without wading through pages of irrelevant links.
- **Increased engagement**: Relevance is key to engagement. When users find what they are looking for, they are more likely to spend time interacting with the content since they more quickly find what they're looking for.

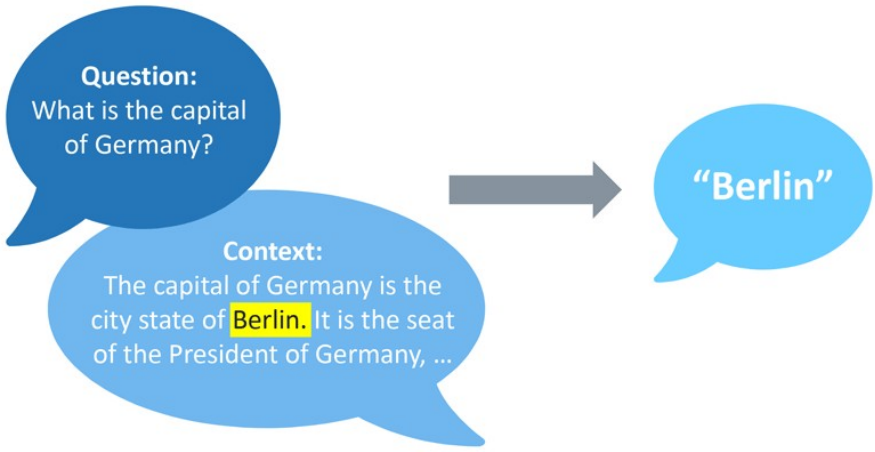
Examples of semantic search

Let's illustrate semantic search with a few examples:

1. **Understanding related terms**
   A search for " running shoes," for example, on a large e-commerce website, can illustrate how a semantic search engine operates. The engine understands that "running shoes" are related to terms like "sneakers," "athletic footwear," and "jogging shoes." It might also consider brands like Nike, Adidas, or Brooks, known for producing running shoes.
2. **Considering context**
   The search "trail maps" on a national park's website can demonstrate how location context impacts results. A semantic search engine, using the user's IP address or a previously provided location, might prioritize results for trail maps near their location. If the user is near the park's northern entrance, for example, the engine may prioritize maps for trails accessible from that point.
3. **Interpreting natural language**
   Semantic search excels at understanding natural language queries. For instance, a search like "what's the weather like in Paris next week?" on a search engine would be interpreted correctly, retrieving a weather forecast for Paris for the following week. The engine breaks down the query and understands the intent despite it being phrased conversationally.

Question Answering

Question Answering is a computer science discipline within the fields of information retrieval and natural language processing, which focuses on building systems that automatically answer questions posed by humans in a natural language. A computer understanding of natural language consists of the capability of a program system to translate sentences into an internal representation so that this system generates valid answers to questions asked by an user [1]. Valid answers mean answers relevant to the questions posed by the user. As the internal representation of natural language, sentences must adequately map semantics of this statement, the most natural approach is in the simulation of facts contained in the sentences using a description of real objects as well as actions and events connected with these objects. To form an answer it is necessary, in the first place, to execute the syntax and semantic analysis of a question. This article covers the introduction to Question Answering, types and challenges posed by the systems in real world.



A user who types in "What is the capital of Germany?" not only receives a web page with related information, but also the concrete answer "Berlin". This saves the user a lot of time in finding the answer, especially if he would otherwise have to trawl through particularly long documents.

How does a natural language question answering system work?

Developing question answering systems has been a hot topic in IT for quite some time. Earlier, there were attempts to set up complex rules to enable a system to understand a user's naturally worded question and to provide an answer.

The natural language question answering systems of today often take an **extractive** approach, and consist of a **retriever** and a **reader**. The answers to the questions are not stored in large databases, rather the system attempts to find and extract an appropriate answer to a user's

question from a mass of texts. Firstly, documents relevant to the user's question are loaded from a document store. Then, the reader attempts to extract the answer to the user's question.

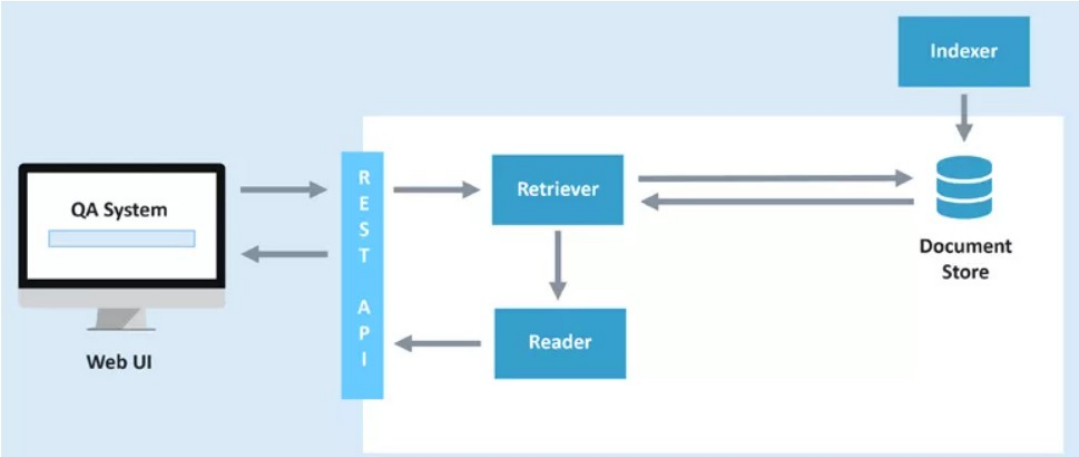


Figure 2: The roles of indexer, document store, retriever and reader in a natural language question answering system

What is a document store?

The document store is responsible for providing relevant documents. There are different ways of doing this. Often, a simple or reverse index is used. One well-known, commonly used such reverse index is Apache Lucene, which is used as an index in Elasticsearch. It's a fast, efficient way to call up documents.

What is a retriever?

A retriever is responsible for retrieving relevant documents for the user's question. First of all, it attempts to **extract the relevant terms in the question**. It then uses these to retrieve relevant documents.

In order to turn a user's question into the type of query a retriever can process, various Natural Language Processing (NLP) techniques are used. These include:

1. **Removing punctuation**
   Full stops, commas and other punctuation are superfluous in retrieving relevant documents. These are therefore removed from the user's question.
2. **Removing stopwords**
   Stopwords are commonly occurring words which don't have a significant impact on content. Examples include articles such as 'the', 'a', 'an'. These words are therefore filtered out.

3. **Tagging entities**
   Entities, such as products or names, are usually very relevant to the query. These are therefore incorporated into the query.

4. **Stemming**
   Words can appear in different forms or conjugations (walk, walked, walking, etc.). As they may well appear in different forms within in a document, such words are reduced to a base form before being incorporated into the query.

These steps help create a query, which is then made to the document store. The retriever grabs the most relevant documents and passes these on to the reader to extract an answer to the user's question.

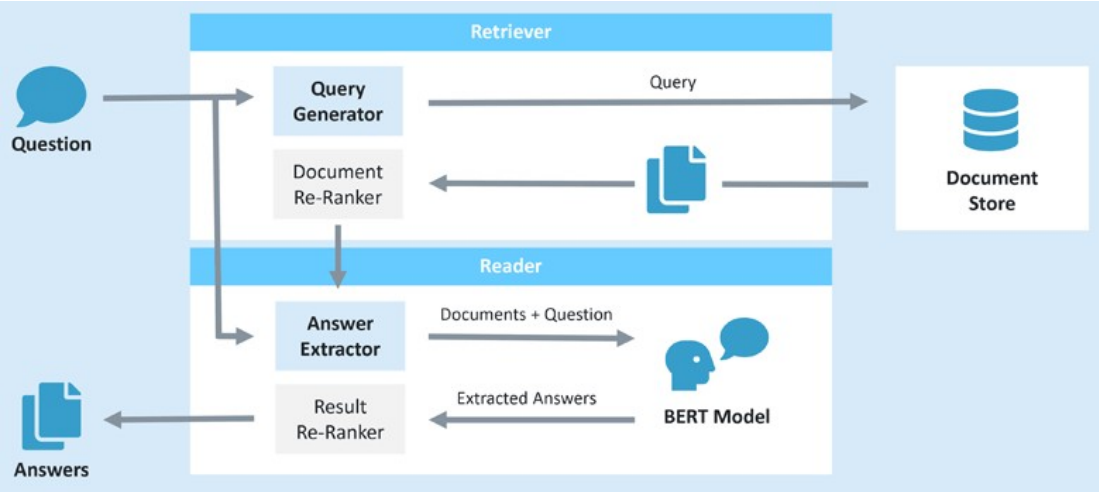


Figure 3: The BERT model in a natural learning question answering system

What is a reader in a question answering system?

A reader is responsible for extracting an answer from the documents it receives. Using a suitable language model, it tries to understand both the question and the documents and to extract the most appropriate answer from the texts.

Types of Question Answering

There are three major modern paradigms of question answering:

a) **IR-based Factoid Question Answering** goal is to answer a user's question by finding short text segments on the Web or some other collection of documents. In the question-processing phase a number of pieces of information from the question are extracted. The answer type specifies the kind of entity the answer consists of (person, location, time, etc.). The query specifies the keywords that should be used for the IR system to use in searching for documents.

b) **Knowledge-based question answering** is the idea of answering a natural language question by mapping it to a query over a structured database. The logical form of the question is thus either in the form of a query or can easily be converted into one. The database can be a full relational database, or simpler structured databases like sets of RDF triples. Systems for mapping from a text string to any logical form are called semantic parsers. Semantic parsers for question answering usually map either to some version of predicate calculus or a query language like SQL or SPARQL.

c) **Using multiple information sources: IBM's Watson** system from IBM that won the Jeopardy! challenge in 2011 is an example of a system that relies on a wide variety of resources to answer questions. The first stage is question processing. The DeepQA system runs parsing, named entity tagging, and relation extraction on the question. Then, like the text-based systems, the DeepQA system extracts the focus, the answer type (also called the lexical answer type or LAT), and performs question classification and question sectioning. Next DeepQA extracts the question focus. Finally the question is classified by type as definition question, multiple-choice, puzzle or fill-in-the-blank. Next is the candidate answer generation stage according to the question type, where the processed question is combined with external documents and other knowledge sources to suggest many candidate answers. These candidate answers can either be extracted from text documents or from structured knowledge bases. Then it is passed through the candidate answer scoring stage, which uses many sources of evidence to score the candidates. One of the most important is the lexical answer type. In the final answer merging and scoring step, it first merges the candidate answers that are equivalent. The merging and ranking is actually run iteratively; first the candidates are ranked by the classifier, giving a rough first value for each candidate answer, then that value is used to decide which of the variants of a name to select as the merged answer, then the merged answers are re-ranked.

Challenges in Question Answering

The main challenges posed by a Question Answering System are described below:

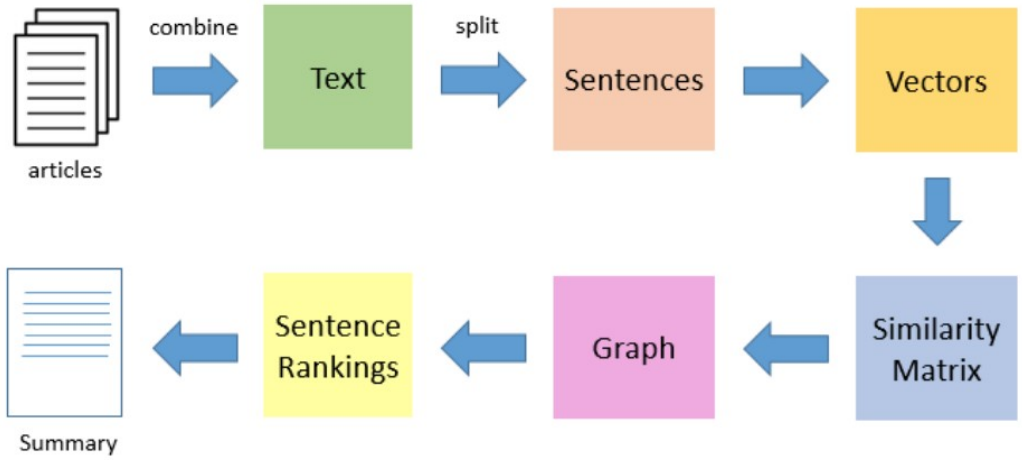
1. **Lexical Gap** : In a natural language, the same meaning can be expressed in different ways. Because a question can usually only be answered if every referred concept is identified, bridging this gap significantly increases the proportion of questions that can be answered by a system.
2. **Ambiguity** : It is the phenomenon of the same phrase having different meanings; this can be structural and syntactic (like "flying planes") or lexical and semantic (like "bank"). The same string accidentally refers to different concepts (as in money bank vs. river bank) and polysemy, where the same string refers to different but related concepts (as in bank as a company vs. bank as a building).

3. **Multilingualism** : Knowledge on the Web is expressed in various languages. While RDF resources can be described in multiple languages at once using language tags, there is not a single language that is always used in Web documents. Additionally, users have different native languages. A QA system is expected to recognize a language and get the results on the go!

Text summarization using NLP

Text summarization is the process of generating short, fluent, and most importantly accurate summary of a respectively longer text document. The main idea behind automatic text summarization is to be able to find a short subset of the most essential information from the entire set and present it in a human-readable format. As online textual data grows, automatic text summarization methods have the potential to be very helpful because more useful information can be read in a short time.

In this approach we build algorithms or programs which will reduce the text size and create a summary of our text data. This is called automatic text summarization in machine learning. Text summarization is the process of creating shorter text without removing the semantic structure of text.

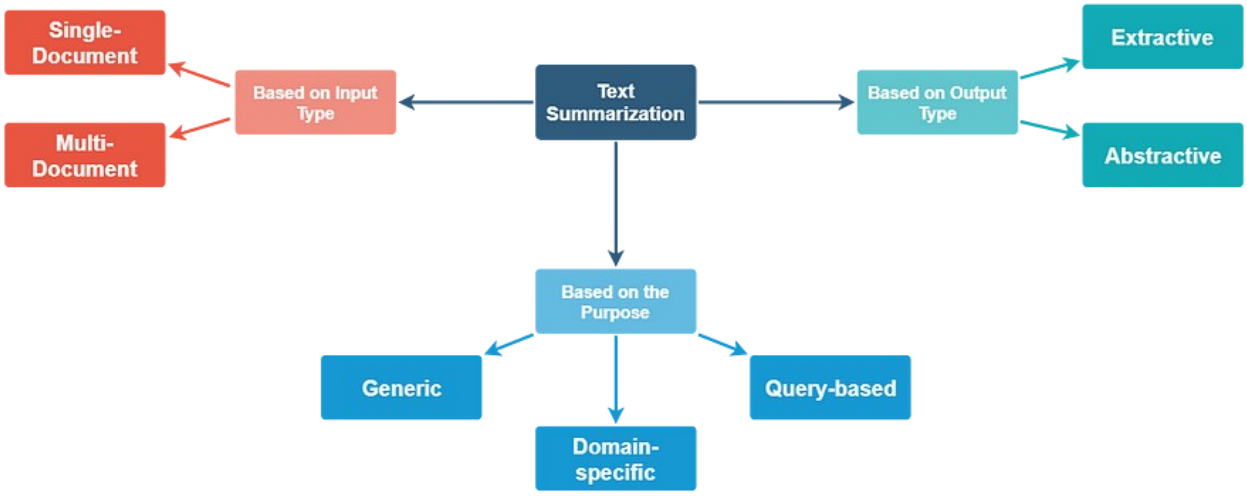


Why automatic text summarization?

1. Summaries reduce reading time.
2. When researching documents, summaries make the selection process easier.
3. Automatic summarization improves the effectiveness of indexing.
4. Automatic summarization algorithms are less biased than human summarization.

5. Personalized summaries are useful in question-answering systems as they provide personalized information.
6. Using automatic or semi-automatic summarization systems enables commercial abstract services to increase the number of text documents they are able to process.

Type of summarization:



Based on input type:

1. Single Document, where the input length is short. Many of the early summarization systems dealt with single-document summarization.
2. Multi-Document, where the input can be arbitrarily long.

Based on the purpose:

1. Generic, where the model makes no assumptions about the domain or content of the text to be summarized and treats all inputs as homogeneous. The majority of the work that has been done revolves around generic summarization.
2. Domain-specific, where the model uses domain-specific knowledge to form a more accurate summary. For example, summarizing research papers of a specific domain, biomedical documents, etc.
3. Query-based, where the summary only contains information that answers natural language questions about the input text.

Based on output type:

1. Extractive, where important sentences are selected from the input text to form a summary. Most summarization approaches today are extractive in nature.

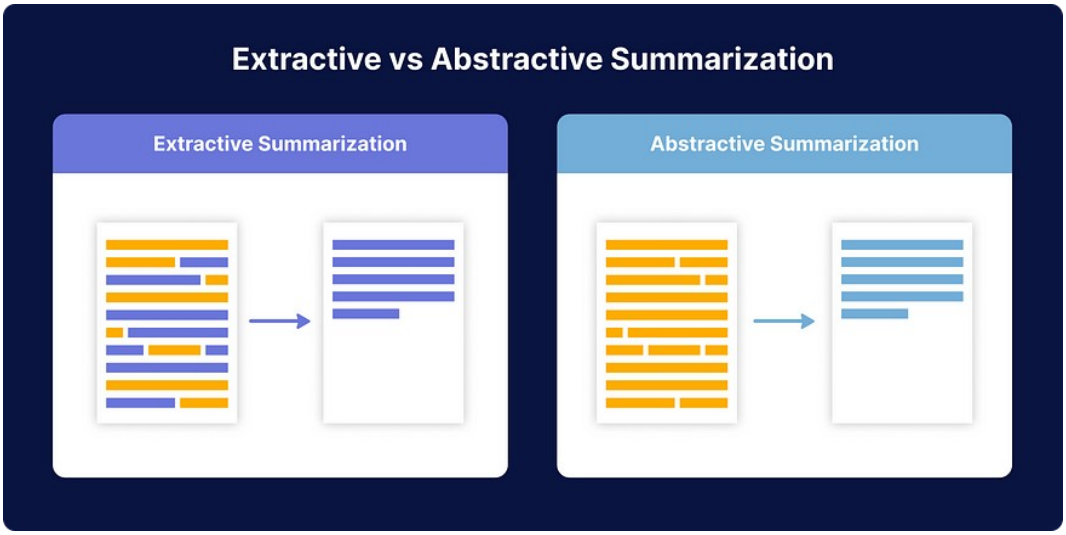
2. Abstractive, where the model forms its own phrases and sentences to offer a more coherent summary, like what a human would generate. This approach is definitely more appealing, but much more difficult than extractive summarization.

How to do text summarization

- Text cleaning
- Sentence tokenization
- Word tokenization
- Word-frequency table
- Summarization

Extractive Vs Abstractive Summarization

Extractive summarization creates a summary by extracting key sentences or phrases from the original text, while abstractive summarization generates a new summary in its own words, paraphrasing and rephrasing the original text. Essentially, extractive summarization is about identifying and extracting the most important parts of the original text, whereas abstractive summarization is about understanding the meaning and re-expressing it in a new, more concise way.



Extractive Summarization:

- **Process:** Identifies and selects the most important sentences or phrases from the original text.
- **Output:** A summary that is a subset of the original text.
- **Advantages:** Simpler to implement, less computationally intensive.

- **Disadvantages:** May not capture the nuances or full meaning of the original text.

Abstractive Summarization:

- **Process:** Understands the meaning of the original text and generates a new summary using its own words.
- **Output:** A new summary that is not a direct copy of the original text.
- **Advantages:** Can create more concise and informative summaries.
- **Disadvantages:** More complex to implement, requires advanced NLP techniques.

Key Differences:

Feature	Extractive Summarization	Abstractive Summarization
Process	Selects and extracts key parts	Generates new text based on understanding
Output	Subset of original text	New summary with original meaning
Complexity	Simpler	More complex
Computational Cost	Lower	Higher
Human-like Summaries	Less natural	More natural and human-like

Machine Translation

Machine translation technology enables the conversion of text or speech from one language to another using computer algorithms.

In fields such as marketing or technology, machine translation enables website localization, enabling businesses to reach wider clientele by translating their websites into multiple languages. Furthermore, it facilitates multilingual customer support, enabling efficient

communication between businesses and their international customers. Machine translation is used in language learning platforms to provide learners with translations in real time and improve their understanding of foreign languages. Additionally, these translation services have made it easier for people to communicate across language barriers.

How does machine translation work?

Machine translation works by using advanced algorithms and machine learning models to automatically translate text or speech from one language to another. Here's how it generally happens:

1. First, the input text or speech is prepared via filtering, cleaning and organizing.

2. Then, the machine translation system is trained using examples of texts in multiple languages and their respective translations.

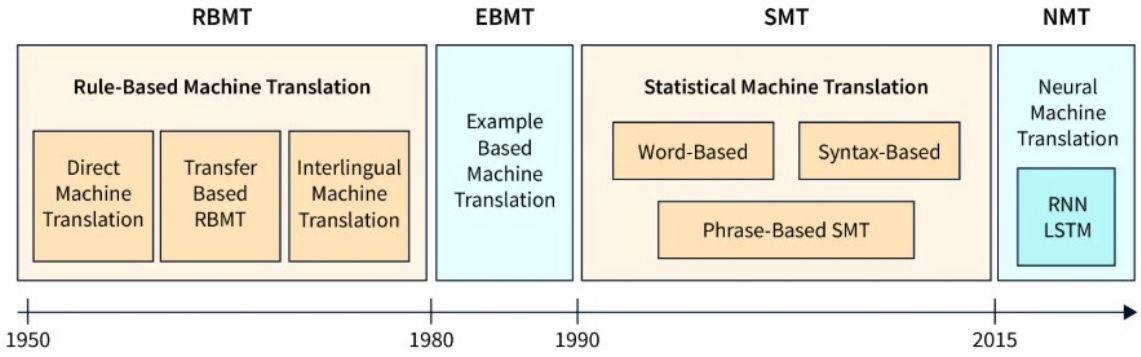
3. The system learns and analyzes examples to understand patterns and probabilities of how words or phrases are translated.

4. When a new text to translate is inputted, the system uses what it has learned to generate the translated version.

5. After generating the translation, some additional adjustments may be added to refine the results.

Different approaches to machine translation

Here are some common approaches machine translation uses to translate one text or language into another.



1. Rule-based machine translation (RBMT). In rule-based machine translation, linguistic rules and dictionaries are used to generate translations based on established language rules and structures. These rules define how words and phrases in the source language should be

transformed into the target language. RBMT requires human experts to create and maintain these rules, which can be time-consuming and challenging. It often performs better for languages with well-defined grammatical rules and less ambiguity and metaphors.

Example: A rule-based translation system might have a rule stating that the word "dog" in English should be translated to "perro" in Spanish.

2. Statistical machine translation (SMT). Statistical machine translation involves analyzing vast amounts of bilingual texts to identify patterns and probabilities for accurate translation. Instead of relying on linguistic rules, SMT uses statistical models to determine the most likely translations based on patterns observed in the training data. It aligns source and target language segments to learn translation patterns. SMT works well with larger training data and can handle diverse language pairs.

Example: In SMT, the system might learn that "cat" often appears in the same context as "gato" in parallel bilingual texts, leading to the translation of "cat" as "gato."

3. Syntax-based machine translation (SBMT). Syntax-based machine translation takes into account the syntactic structure of sentences to improve translation accuracy. It analyzes the grammatical structure of the source sentence and generates a corresponding structure in the target language. SBMT can capture more complex relationships between words and phrases, allowing for more accurate translations. However, it requires sophisticated parsing techniques and can be computationally expensive.

Example: SBMT learns the syntactic structure of a sentence and ensures that the subject and verb agreement is maintained in the translation for a more grammatically accurate output.

4. Neural machine translation (NMT). Neural machine translation utilizes deep learning models, particularly sequence-to-sequence models or transformer models, to learn translation patterns from training data. NMT learns to generate translations by processing the entire sentence, considering the context and dependencies between words. It has demonstrated significant improvements in translation quality and fluency. NMT can handle long-range dependencies and produce more natural-sounding translations.

Example: NMT takes an input sentence like "The cat is sleeping" and generates a translation like "El gato está durmiendo" in Spanish, capturing the context and idiomatic expression accurately.

5. Hybrid machine translation (HMT). Hybrid machine translation may incorporate rule-based, statistical and neural components to enhance translation quality. For example, a hybrid system might use rule-based methods for handling specific linguistic phenomena, statistical models for general translation patterns, and neural models for generating fluent and contextually aware translations.

Example: A hybrid system could use a rule-based approach for handling grammatical rules, statistical models for common phrases, and a neural model to generate fluent translations with improved context understanding.

6. Example-based machine translation (EBMT). Example-based machine translation relies on a database of previously translated sentences or phrases to generate translations. It searches for similar examples in the database and retrieves the most relevant translations. EBMT is useful when dealing with specific domains or highly repetitive texts but may struggle with unseen or creative language usage.

Example: If the sentence, "The cat is playing," has been previously translated as "El gato está jugando," EBMT can retrieve that translation as a reference to translate a new sentence, "The cat is eating."