


# Developing a Semantic Search Engine for Modern Tibetan

James Engels (University of Edinburgh)  
and  
Robert Barnett (SOAS University of London)\*

atural language processing (NLP) as a scientific and mathematical realm has undergone at least two generational shifts since the first earnest work on Tibetan NLP began in the late 1990s and early 2000s (e.g., Hackett 2000, Jiang 2003). The first generation of generic NLP tools was developed from theoretical foundations that were established as early as the 1950s – implementations in computers of (theoretically) crosslinguistically applicable formalisms that could be used to create relatively small-scale tokenisers and parsers. The practical realisation of the second and third generations of NLP development arrived soon after the publication of the first mathematical proposals on which they rely.<sup>1</sup>

---

\* The research for this paper was conducted as part of the Divergent Discourses project which received funding from the Deutsche Forschungsgemeinschaft (DFG) under project number 508232945 (<https://gepris.dfg.de/gepris/projekt/508232945?language=en>), and from the Arts and Humanities Research Council (AHRC) under project reference AH/X001504/1 (<https://gtr.ukri.org/projects?ref=AH%2FX001504%2F1>). For more information on Divergent Discourses, see <https://research.uni-leipzig.de/diverge/>. Coding for all tasks described in this paper was carried out by James Engels.

<sup>1</sup> Blei *et al.* (2003) provide a reasonable benchmark for the original mathematics of generation II, in which the authors proposed Latent Dirichlet Allocation (LDA), a method still used for topic modelling across small-scale applications. The transformer revolution characterises Generation III, starting with the landmark paper “Attention is All You Need” by Vaswani *et al.* (2017). The impact of “Attention” was immediate and overwhelming.

Despite consistent effort by a small number of dedicated researchers, the availability of tools from the most recent (post-transformer) generation of NLP advancement depended on parseable corpora at a scale that was not achievable in Tibetan, at least until roughly 2020. Recent work by Marieke Meelen and collaborators has contributed greatly to language-technological support for classical Tibetan, either in the form of preliminary parsing models and treebanks (Faggionato & Meelen 2019) or large parsed corpora (Meelen *et al.* 2021).

This article is concerned with the development, for the first time, of a semantic search engine for modern Tibetan. Semantic search contrasts with keyword search in its capacity to compare similarity of meaning across queries and results, independent of literal lexical overlap. To illustrate this capacity, we briefly describe how this is achieved, first by contrasting it with a prototypical keyword search method and then by discussing the major components of a semantic search system that is under development for general use.

Keyword searches were the standard for search systems for most of the life of the internet – indeed, when most people imagine a search system, they usually imagine a keyword search. To make keyword searches useable, the system must reward query results that contain “useful” words – that is, keywords that can be used to diagnose a more relevant result – and punish more common words by, essentially, ignoring them. To some extent, this is achieved through lists of stopwords – very common words, such as “is” or “the”, which are usually pre-excluded from searches. Thus, the query “average weight of an elephant” would be reduced to “average weight elephant” for maximum relevance, and this can be done by systematically removing all instances of “of” and “an.” But for more complex cases, where much less frequent terms mix with much more frequent terms, such as a query about a “new building site overseen by the Ministry of Finance”, it is the combination of relatively frequent terms that generates the most relevant results. Most keyword search systems typically guess at the relative importance of words in a query by checking for their frequency in the overall corpus. Even simpler

versions of keyword searches check whether strings in their searchable data contain the same strings as in the query.

Semantic search, on the other hand, allows the user to enter a long query (ideally two to three sentences or the equivalent) and to receive results based on the similarity of the content of the returned result, independent of keyword similarity. Semantic search systems inherently require a large corpus from which to build a semantic space or a prefabricated semantic space that can be secondarily imported and used to generate the search space for the smaller corpus. This contrasts with most keyword search systems. To illustrate, consider our previous “Ministry of Finance building site” example: a human reader with world-knowledge might be able to deduce that a different text about a “newly constructed tax office” in the same region is likely the same building, but with no overlapping keywords, that text would be invisible to a standard keyword search. A well-trained semantic search system will identify the underlying conceptual overlap between the two queries and will return results containing the second to a user querying the first.

For well-resourced languages with well-developed lower-level NLP tools, creating a facsimile of that language’s semantic space is computationally intensive but resource-trivial because the training data and/or the frameworks are already readily available and widely known. The most important component of such a process is a reliable tokeniser – a tool that can reliably identify and split words. In white-spaced languages like those found in Europe, this begins with the relatively trivial step of splitting text on white spaces and punctuation and compiling them into a list. Languages of East and Southeast Asia, on the other hand, tend to identify syllables using overt marking but do not differentiate in writing between, say, two consecutive monosyllabic words and a disyllabic word (for more on the tokenisation problem in Tibetan, see Meelen *et al.* 2021 and Kyogoku *et al.* 2025 in this issue).

What is to be done? Small language models rely on rule-based tokenizers. In languages without word spacing, this creates a significant challenge. These models must use long and complex lookup algorithms. The process works by checking syllables one by

one against a large hidden dictionary. If no match is found, the algorithm checks pairs of syllables for possible entries. This continues until a likely match is identified. The process then repeats with any remaining syllables until the entire text is processed.

Large language models (LLMs) like GPT work quite differently. They function more like human readers. This is achieved through their training process. They are trained on billions of texts and trillions of words in natural contexts. Through this exposure, the models develop an understanding of word distribution patterns. As a result, they develop something akin to natural language understanding. This allows them to parse text in a way that resembles how proficient human readers process language.

The ability of an LLM to tokenise is a secondary property of what might be called its “understanding space,” an abstract geometry of relationships between all its word-concepts. An Understanding Space can be imagined as a complex map showing how all words and concepts relate to each other. This map is created through training on massive amounts of text data. While it cannot be perfect (since that would require learning from every possible sentence), it gets remarkably close to human-like understanding, generated by training on billions of examples. The relationships between words, and indeed entire texts/utterances, is a literal projection onto the multidimensional space that the machine generates from its past training, and once the training is complete, that understanding space becomes fixed.

Once this understanding map is complete, it's used to create a second system, a second multidimensional space (let's call it the “Memory Space”), into which texts can be projected, i.e., where actual documents get stored for searching. When the semantic search database is first generated, texts are converted into mathematical representations or vectors by projecting the text content into the memory space using the understanding space.

Once the Understanding Space is set, new search results can be indefinitely added to the Memory Space using the same Understanding Space without the need to retrain anything. When the search tool actually conducts a search, the user's query is projected

temporarily into the memory space and compared to its most similar stored vectors corresponding to documents.

Training an understanding space from scratch for a low-resource language is no easy task, requiring gigabytes to terabytes of well-organised text. We found, however, that Cohere, a company founded by former Google scientists, had released access to understanding spaces for many low-resource languages to little fanfare. Its default embedding model includes native support for Tibetan, and its Multilingual Model 2.0 provides an out-of-the-box understanding space for Tibetan.<sup>2</sup> It also claims to provide support for, among other languages, Arabic-script Uyghur and (Cyrillic) Mongolian.

More recently, other large corporate entities have incorporated Tibetan into their recent wave of language technology offerings, with varying degrees of success. Google Translate introduced Tibetan in June of 2024, though proficient Tibetan readers are generally unsatisfied with its capabilities, impressionistically comparing it to Google Translate's powers in Spanish and French from its earliest days. OpenAI significantly improved its Tibetan proficiency between GPT-4 (2023) and GPT-4o (2024), generating passable translations from English to Tibetan and back again. In general, by late 2024, Tibetan language understanding and text generation from corporate AI services had progressed from being practically useless curiosities to reliable everyday tools for Tibetan text analysis. However, while these tools are useful for a single user with a single query at, say, the ChatGPT interface, their capacity for use at larger scales requires a level of programming knowledge that not all Tibetologists have, to say nothing of the high upfront costs for constant API requests.

Given this evolving context in the new capabilities available for machine translation and NLP of Tibetan, the rest of this paper is intended to achieve two goals. The first is to familiarise less technically proficient Tibetologists with the broad theories required to build such a system, in roughly chronological order of their appearance and

---

<sup>2</sup> Cohere's user policies are additionally attractive because the vectors that their model generates are the property of the user who generated them.

application in NLP systems. The second is to explain the structure of our semantic search system, which relies in differing amounts on theoretical elements from each of the sections.

## 1 *Vectorisation*

Our project, *Divergent Discourses*, includes two main channels for developing computational tools for use with modern Tibetan. One involves adapting an existing tool, the integrated Leipzig Corpus Miner (iLCM), to carry out complex forms of keyword searches in modern Tibetan using an approach based on a method called Latent Dirichlet Allocation (LDA, developed by Blei *et al.* 2003; see below) and an engine provided by the language modelling software package spaCy (see Kyogoku *et al.* 2025 in this issue). The second involved the building of two applications for use with modern Tibetan texts that use a vector-based approach: a topic modelling engine (automatic, corpus-scale identification of textual foci; see Schwartz & Barnett 2025 in this issue) and a semantic search engine.

For the first task, a good tokeniser is the single most important low-level NLP requirement: without it, the process faces a bottleneck. This is also the case with developing Named Entity Recognition (NER) for any language, or if one is creating a language model for a platform such as spaCy. However, vector-based topic modelling and semantic searching do not need tokenising or part-of-speech (POS) tagging to be carried out on their training data.

### 1.1 *Document Vectorisation*

In NLP, a vector is a numerical representation of a word, phrase, sentence, or document that captures its meaning in a way that computers can process. Vectorisation can be imagined as converting a language into a list of numbers that preserve the semantic relationships of its constituencies. One of the earliest methods of vectorisation (by the standards of AI development) is called “term

frequency-inverse document frequency" (TF-IDF), originally proposed by Karen Spärck Jones (1972). In this system, which is still frequently used, including by the iLCM, each document (in NLP, the term "document" is generally used for a paragraph) in a corpus is vectorised, as well as each word. A matrix is created in which the rows consist of the numerical strings ascribed to each document (paragraph) and columns hold the strings for each word in that document. This produces a frequency table, which can be used as a baseline or master table for comparison with matrices produced for other documents, as well as with the weight of terms reflecting their overall frequency in the entire corpus. This by default punishes very common content words and rewards more specific or unusual terms. This generates a matrix from which specific features of documents are easier to extract solely because their numerical values will be further from the mean. This is particularly useful for topic modelling.

Techniques like TF-IDF are usually termed document vectorisation because their dimensionality (size) is exactly the number of unique features across the entire corpus – i.e., the total number of unique non-stopword words. Because each document matrix consists only of the co-occurrences of entries in the lexicon, they are known as *sparse* matrices, where any column will only have one cell containing a value, and all others will be empty or 0. Suppose there are  $D$  unique words in a document and  $V$  unique words in the entire corpus. In that case, the size of the TF-IDF matrix will be 2-dimensional  $D \times V$ . TF-IDF matches words with a statistic that makes those words diagnostic for identifying a certain document or set of documents, by rewarding a document/page/search return object for the frequent use of uncommon words while punishing documents for frequently using common words in general. Straightforward TF-IDF is considered a fairly simple document-ranking model, but many (perhaps even a majority of) ad-hoc search systems on the internet today use TF-IDF or a refinement of it.

An ideal semantic vector space has been trained on every possible felicitous input from a given language. Of course, while the number of felicitous inputs is basically infinite, the point of diminishing returns is only reached in corpora (such as those in English, Spanish, Russian,

or Chinese) of enormous size. TF-IDF is only as good as the documents you use to build it – there is no real “training” process, just very large amounts of first-order matrix arithmetic. More advanced “word embedding” models that work with higher dimensions have to be trained on very large amounts of text in the target language, to develop some kind of internal sense of a distributional “vocabulary” into which they can insert the new documents. The necessary amount of data is not available in Tibetan, though we have found that word embeddings with a high-resource language baseline (like English in our spaCy model) can and do still perform better than strict language-internal matrix generation methods like TF-IDF.

When a text is transformed into a dense embedding vector, it not only interprets the text as a sequence of numbers but also positions that sequence in conceptual space. It is thus envisaged as existing in multiple (mathematical) dimensions. In the case of the Cohere multilingual model (one of the types of embeddings offered by Cohere), it is a space with 768 dimensions. By projecting the vectors in this multidimensional space, the computer can subsequently perform a range of algebraic operations (angle comparison → cosine similarity, dot product, etc) to measure similarity, compare them analogically, and so on. In simple language, because the computer can recognise that two numbers or numerical arrays are close to each other in a mathematical sense, it can infer that once those strings have been translated back into the words or tokens they represent, those words or tokens will be similar in meaning. This is the basic principle behind semantic searching.

Both topic modelling and semantic searches require some capacity for dynamic updates – meaning the capacity for the machine to improve its knowledge over time as new resources are added to the corpus. This is what a possible data flow (pipeline inputs) for the creation of a useable vector space for topic modelling and/or searching might be:

*Raw text → Tokenised Text → Selected Features →  
Embeddings // Understanding Space*



Each of the steps following “Tokenised Text” broadly relies on increasingly complex matrix algebra, but the critical input unit for any of those linear-algebraic operations is a “feature”, a term generally used to refer to any word in a text apart from the stopwords. Put more simply, generally speaking (though not always), a feature is always a word, but not every word is a feature. Those selected features will then be vectorised, although the specific shape or dimensionality of the vector array varies by method. Stopword removal is not necessary for LLM-type searches, but it is for traditional Boolean-keyword searches<sup>3</sup> and various common topic modelling methods like LDA.

### 1.2 *Word Embeddings*

Word embeddings were the technological successor to document vectorisation and were industry-standard from the mid-2010s (Word2Vec, for example, was released in 2013) until the large-scale development of transformers from 2017 onwards. Canonical methods of word embeddings at that time included Word2Vec and later GloVe (Global Vectors for Word Representation, developed by the StanfordNLP group). Word2Vec and GloVe are optional embedding methods included in boilerplate spaCy, and both embedding models are critically reliant on large training corpora for ideal performance. Word2Vec represents terms (words, tokens,  $n$ -grams) as “dense” vectors in a single space generated by a reading of the entire corpus at once to generate the distributional semantics of features, and the size of the embedding space is determined by the specific embedding method or formula. Dense matrices fill all the cells of their matrices with meaning-bearing values, unlike sparse matrices (such as TF-IDF), where each row and/or column corresponds to a single feature or unit of analysis. Overwhelmingly, most cells in a sparse matrix will thus

---

<sup>3</sup> A Boolean keyword search uses logical operators (AND, OR, NOT) to combine search terms and narrow or expand results. For example, “cats AND dogs” yields results containing both terms, while “cats OR dogs” finds results containing either term. NOT excludes terms, like “pets NOT snakes” to find pet content without snake-related results.

have value 0, because a sparse matrix element at position  $ij$  will only be non-zero when the row index of  $i$  is the same as the column index of  $j$ .

Unlike TF-IDF, in a dense embedding matrix,<sup>4</sup> the size of the final space is not dependent on the number of documents or unique words, but rather leverages a series of dimensionality reductions as needed in order to shrink or grow the size of the space to the user's preset specification. The previous generation of Tibetan NLP research (see, for example, Tao *et al.* 2020) invoked Word2Vec as its preferred embedding framework. Unfortunately, because the Word2Vec Tibetan model is closed and only its product (output) is publicly released, then we as end-users would have to retrain a Word2Vec model, which would also mean having to reinvent the wheel for tokenisation, and so forth. If one has a good tokeniser that can be integrated into a very simple pipeline, generating TF-IDF matrices is much easier than training a model to develop a new embedding space for a previously unknown language.

At the time of writing, only corporate entities like Cohere have developed embedding spaces for Tibetan that are practically useful and publicly available. Cohere's model architecture is proprietary and not publicly available but certainly leverages some kind of transformer model (the same technology underlying LLMs like OpenAI's GPT) to create better context-sensitive embeddings for input text. It should also be noted that, from version to version, Cohere's multilingual engine varies in its ability to handle data across languages: Cohere Multilingual Model 2.0 had a much better "natural" understanding of Tibetan than the current available model, 3.0, so we continue to use Model 2.0 for our embeddings. Being a corporate entity, Cohere does not publicly release information about their data sources or model training process; it is difficult, perhaps impossible to speculate why their newer model might have worse Tibetan understanding than the older one. Our communications with the Cohere engineering team suggests that even they are unsure of the details that led to a drop in performance on some languages with the newest version of their

---

<sup>4</sup> Note that dense embedding matrices have more dimensions.

multilingual model. Our experience with the Cohere embedding space has been positive, and we recommend its use to others interested in a variety of embedding-driven NLP tasks for Tibetan, especially searching and document classification.

## 2 *Topic Modelling and Semantic Searching*

Our project is mainly interested in using digital methods to identify “divergent discourses” – changing topics or narrative foci over time in a set of texts. We therefore need to use topic modelling – automatic, corpus-scale identification of textual foci – with Tibetan texts. Topic modelling and its uses are discussed in more detail in Schwartz & Barnett 2025 in this issue, so here we give a simplified description of its basic principles in order to show how it differs from semantic searching. Topic modelling identifies topics or themes in a text based on algebraic forms of detection – i.e., it creates mathematical abstractions of the co-occurring vocabularies of a text to detect narrative threads, whether or not the explicit name of a “topic” is mentioned. These themes are extracted based on the semantic similarity of content-bearing texts. As a result, the identified topic or theme may be labelled with a word that never occurs in the text it is attributed to. One approach to topic modelling is that used by the iLCM. It infers topics using LDA, a conditional probabilistic method. LDA assumes, firstly, that documents are collections of “topics,” which it understands as inherently mathematical abstractions corresponding roughly to narrative threads, and secondly, that topics are composed of words, and topics can be extracted from corpus-level rather than document-level word distributions, the former of which of course are too large to be visible to an end-user. It uses statistical (Bayesian) methods to calculate how diagnostic or indicative certain word combinations are of a new topic and then plots the distribution of those content words to identify probable topics. The user can adjust or refine that probability by predefining a few likely topics, if so desired. LDA requires that you specify ahead of time how many topics to search for. Most importantly, it is at its core “keyword” based, in

that all the “words” on which it works are listed ahead of time and in effect searched for. LDA can identify multiple topics per document and is based on word frequency, working in a somewhat similar way to building concordances to determine whether Shakespeare wrote a play.

For some purposes, a different approach to topic modelling will be more relevant, depending on what one is trying to accomplish. As we have seen, transformer models are the current standard for high-performance NLP applications. Transformers achieve improved contextual understanding over previous-generation NLP tools mainly by encoding individual tokens based on the encodings of other tokens in the same sentence, so that polysemous and homophonous distinctions will naturally emerge given enough training data (a “bow” for shooting arrows, bending at the waist, wrapping a gift, and the front of a ship would each be assigned unique representations), while the model also stores positional information about each token to learn and reproduce good natural language syntax.

For a task such as topic modelling, transformers, when combined with other methods, have proven effective when used with high-resource languages. Among these transformer-hybrid methods, BERT (bidirectional encoder representations from transformers), a language model developed for generating embeddings of texts (Devlin *et al.* 2019) was applied to Tibetan texts by scholars in Tibet in 2022.<sup>56</sup> At that time, this approach was primarily used only for document classification. BERTopic, a topic modeller based on the BERT architecture, is perhaps the best known (Grootendorst 2022). Ronald

---

<sup>5</sup> Although it has since been outpaced by stronger models, BERT remains the standard against which large language model architectures are measured.

<sup>6</sup> Two BERT models for Tibetan were released in 2022: TiBERT (Sun *et al.*, Minzu University) and Tibetan BERT (Zhang *et al.*, Tibet University). The two models mainly differ in size and computational complexity: TiBERT is an adaptation of the baseline BERT model,<sup>6</sup> and was designed with text classification in mind. Tibetan BERT is freely available online through HuggingFace, while TiBERT is available through the creators' own distribution, and includes neither the training data nor a detailed description of it. These represented an early revolution in transformer technologies for Tibetan NLP but have largely been outmoded by the more potent forces in corporate AI services.

Schwartz has since succeeded in developing BERTopic as a tool for topic modelling with modern Tibetan texts (Schwartz & Barnett 2025).

Topic modelling allows a user to conduct “outside-in” analysis of a corpus, such as looking at what topics tend to occur over time within a corpus or identifying what is represented in a particular article. Textual analysts, however, also need an “inside-out” method, where they know what kinds of things they want or expect to see but do not know where to find them. For this we need a search system that can match “meanings”, or informal (rather than modelled) topics broadly considered, independent of previously assembled keywords. This is the task we call semantic searching. Semantic searches do not exactly do topic modelling – they find documents (or normally paragraphs) that are similar to a given query text, but they do not give those documents a set of semantic labels. Instead, they keep them in their “natural” state. In a user-driven search application, a semantic search pipeline vectorises the text query input by the user and then compares its similarity with that of the vectorised corpus. It then ranks the documents in the corpus according to their similarity to the query. This leads to overall better search results than simply searching for a specific keyword, since semantic search will detect words that are lexically dissimilar but have semantically similar values or contexts, as with “Tibetologists” and “people who study Tibetan texts”. In the same way, if the word “military” is in the query, semantic search will also return documents about “army” and “navy”. The results of a semantic search are thus based on similarity of meaning rather than word frequency or resemblance.

### 3 *Building a Semantic Search Tool for Tibetan*

The possibility of developing semantic search for Tibetan, and the use of Cohere to create such a system, was first proposed by Ronald Schwartz in 2023 (Schwartz & Barnett 2025). Our communications with various members of the Tibetan studies community indicate that he was the first to realise that such a system could be developed for Tibetan and that the tools needed for this task were by then already

available. As a pilot project, he first developed a semantic search system for use with newspaper texts in Simplified Chinese. These were based on the Cohere multilingual model and formed the architectural basis for the Tibetan semantic search tool. The Schwartz prototype digested the Chinese-language versions of his test corpus, which consisted of several thousand *Tibet Daily* articles stored in HTML files, covering years from 2008–2023. Those HTML files were then converted into CSV files using a bespoke parser that separates the main text into chunk “paragraphs” of roughly 256 characters. These were then associated with their metadata. His tool also demonstrated the practicability of providing toggleable automatic translation in English for each Chinese chunk or paragraph.

Working closely with Schwartz, we then adapted the basic architecture of his Chinese-language semantic search engine for use with texts in modern Tibetan. We again used Cohere, since it is unique among AI services in providing built-in support for Tibetan. We needed to resolve two issues in particular: an accessible form of storage for our vectors, and a supply of data in Tibetan as our test corpus. To store our vectors (for each document there is a vector of 768 numbers, totalling ~278,000 vectors for our test corpus), we used a cloud storage service called Pinecone, which specialises in storing vectors and provides an API to access the vector database for end-users to create their own search engines. To test our semantic search engine, Schwartz provided four years of newspaper articles in Tibetan that he had previously collected between 2020 and 2024. The articles are in HTML form and are remarkably consistent in format and easily parseable but first had to be preprocessed for use by the search tool. This we did by writing a new bespoke parser to extract the text in paragraph-sized chunks (which can be detected by line breaks). The parser also extracts metadata from the HTML file, including the title of the article, the reported author, the date of publication split into its long date string plus its individual components, the publication source and other features, such as the issue number and the source file name.

Using these articles as our test corpus and Cohere to create numerical representations or embeddings, we were then able to build a semantic search system around the embeddings, storing the index in

Pinecone. This index includes not only the vectors but also the raw text and the metadata. New texts can be added dynamically to the corpus by vectorising them with the same method that generated the initial embeddings. New queries can be treated in the same way, except that queries are not saved to the corpus. When we add a new article, the system needs to calculate how similar it is to all existing articles. While this takes more time as our database grows, the processing time increases in a manageable way (linearly rather than exponentially). This is because adding a new article only requires calculating its relationships to existing articles - it does not affect how existing articles relate to each other.

A full schematic of our search system structure can be found in Figure 1. Note that circles indicate objects, squares indicate operations, and green boxes are bespoke python programs that necessarily interact with their connected processes. Coloured circles are interactable from the user's perspective; white circles are hidden from the user's direct access unless specifically declared.

The "raw text with metadata" corresponds in our case to the CSV containing the information we parsed from the HTML files, but the exact format of the data is not of great importance as long as it is machine-readable. The text column of this raw data is then chunked into smaller pieces if and only if the length of the object in the text column is greater than 256 syllables – that is, in effect, if the sum of the text + (!) *shad* + punctuation marks is greater than 256, since Cohere's tokeniser tends to interpret every Tibetan syllable as (computationally speaking) a token. The chunked text is then embedded (converted into vectors) and stored in the database using the Pinecone script, which encircles the embedding process and includes a toggle for inspecting the vectors when they are produced, in addition to sending them to the Pinecone database. When the vectors appear in the Pinecone database, they can also be manually inspected (mainly for the correct dimensionality; the array of numbers itself is inscrutable to humans). This process generates the vector database. Each paragraph in the CSV will have an associated ID in the vector database that corresponds to

its vector. The paragraph ID is associated to the CSV table, which is what allows the comparison operation to be searchable. To interpret the query, we follow an essentially identical process: the

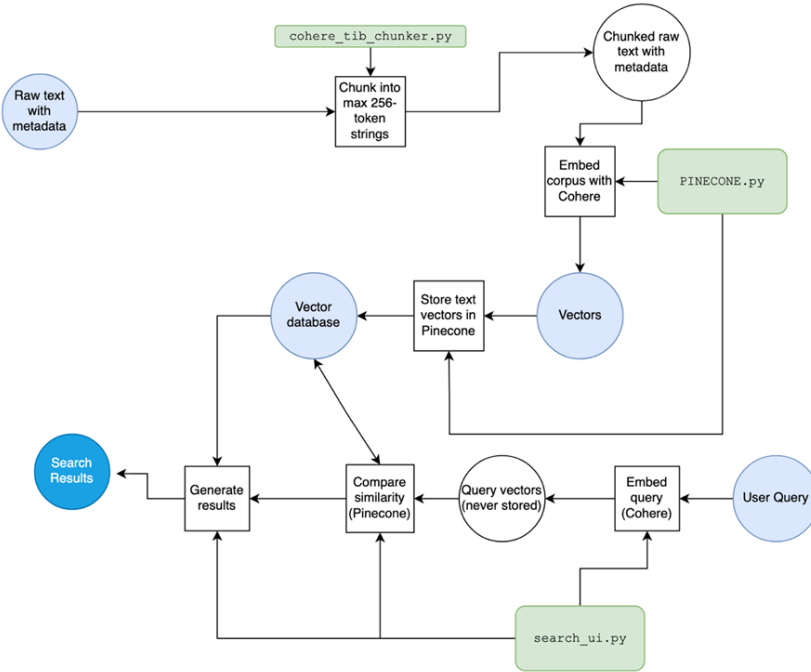


Figure 1 Search System Schematic

query is dynamically embedded in exactly the same understanding space as the other vectors (note that it is *essential* to use Cohere Multilingual Model 2.0 here, and not Multilingual 3.0). The embedded query is then measured against the embeddings of the paragraphs in the corpus using cosine similarity (not inner dot product), and the  $n$  most similar results are returned to the user. All parts of the query pipeline in our prototype are handled in an integrated system with the user interface. After the query results are returned, the query embeddings are forgotten by the system; the query embedding pipeline must be kept separate from the corpus in order not accidentally to contaminate the database with endless queries.

We also added several external functionalities to our search interface: a number of metadata fields (dates, source, title of article,



and so on) and a simple user interface that allows the user to rank results either by relevance (semantic match) or by metadata fields. We added a toggle allowing non-readers of Tibetan to access a translation of either the query or the results in English. For this, we integrated Bing Translate, which, at the time of writing, appears to be the most reliable provider of online translations from Tibetan to English. The public link to the search tool is <https://tibetcorpus.uni-leipzig.de/search/>.

A sample query and result from the semantic search engine is shown in Table 1. The query is taken from an online Tibetan-language news report about the Tingri earthquake in January 2025. The result that the search tool found in our test corpus to be most similar in terms of meaning was an article from a July 2020 issue of *Tibet Daily*. The query refers to the importance of early warnings regarding aftershocks and other risks following the earthquake, and the need to send relief forces to the site of the disaster rapidly. The result does not include the word earthquake or any similar term, but it describes a meeting four years earlier which had stressed the importance of almost exactly the same measures listed in the query. It is a paragraph (or <256 tokens) in length, so it includes additional information besides that included in the query, such as references to epidemic control. But it mirrors, in different words, all the main points in the query about the importance of early warnings and the rapid dispatch of relief forces in the event of what we can guess from context is an earthquake.

Table 1 Sample of semantic search and top result

<p>Query</p>	<p>ཡོམ་འཕྲོ་དང་རི་ཉིལ་བ་སོགས་ལ་ལྷ་ཞིབ་ཚད་ལེན་དང་སྤྱོད་བཟང་གཏོང་བར་ཤུགས་སྤྱོད་རྒྱག་པ། གནས་གཤིས་འགྱུར་སྤྱོད་ལ་དོ་སྣང་ཚུན་པོ་བྱེད་པ། ས་གཤིས་གཞོན་འཛེའི་མཛོད་མེད་རྒྱུན་དུ་ཡོད་པ་ཡོངས་ཁྲུབ་གྲུབ་ལ་བཤེར་བཅས་བྱས་ཏེ་ཞོར་ཐོན་གཞོན་འཛེའི་སྤྱོད་འགོག་ནན་པོ་བྱེད་དགོས། ཟུང་དང་། གློག་ལས་སོགས་རྩལ་གཞིའི་སྤྱི་གཞི་བཞུགས་པའི་གཏོང་སྤྱོད་པོ་གཤམ་རྩལ་ཤུགས་ཡོད་རྒྱུ་སྤུར་བཟོ་བྱེད་པ་དང་གོགས་སེལ་སྤོབས་ཤུགས་དང་གོགས་སེལ་དགོས་མཁོ་ཚན་རིག་དང་མཐུན་པའི་སྤྱོད་སྤྱོད་པའི་གཏོང་གཏོང་བྱེད་པ་དང་ཡོམ་འགོག་གོགས་སེལ་གྱི་རྒྱུ་ལའོར་དང་། དེའི་རྒྱུ་ལའོར་ཤར་སྤྱོད་ཐུབ་པའི་འགན་ལེན་བྱེད་དགོས།</p>
<p>Bing Translation</p>	<p>Strengthen monitoring and early warning of aftershocks and landslides, pay close attention to climate change, survey geological hazards, and strictly prevent the occurrence of secondary disasters. Where roads and other infrastructure facilities are damaged, disaster relief forces and disaster relief</p>

	<p>needs are to be scientifically dispatched to ensure the smooth flow of earthquake relief vehicles, materials, and personnel.</p>
<p>Result #1</p>	<p>Source name: Tibet Daily  Source date: 2020-07-20  Article title &lt; འབྲུབ་འགོག་གི་གནས་སེལ་ལས་དོན་ལ་ཞིབ་འཇུག་དང་བཀའ་ &gt;  ཚོགས་འདུའི་ཐོག་ནན་བཤད་བྱས་དོན། གནད་སློན་ཉེན་བརྗོད་གཏོང་བ་དང་དོགས་ཐོན་ནན་པོ་བྱེད་དགོས་ཤིང་། ཚར་པ་  འབབ་ཚུལ་དང་ཚུའི་གནས་ཚུལ་སོགས་གནས་གཤིས་ཀྱི་གྲངས་གཞི་དུས་ཐོག་དང་འཕྲུགས་མེད་པར་སྔར་བརྗོད་གཏོང་  བ་དང་ཞོར་ཐོན་གནོད་འཚེའི་སྔོན་བརྗོད་གཏོང་རྒྱུ་ལུགས་སྔོན་རྒྱག་དགོས་པ་དང་། ལྷག་པར་དུས་ཚའ་ཤས་སུ་དག་  ཚར་འབབ་པ་དང་། རྒྱ་མཚོའི་རྒྱུ་འཚུབ་དག་ལོ་ལྕང་བ། རྒྱ་ལོག་ཐོན་པ། འདས་རྒྱུ་ཚུན་ཐོན་པ་སོགས་སྔོན་དཔག་  དང་སྔོན་བརྗོད་གཏོང་བའི་རྒྱ་ཚོད་མཐོ་རུ་བཏང་ནས་སྔོན་བརྗོད་ཚུལ་མཐོ་བོ་སྤོང་དང་། དུད་ཚང་། མི་བཅས་སུ་བྱ་བ་པར་བྱེད་  དགོས། རྒྱ་གར་གལ་ཆེན་དང་འགོག་སྲུང་སྐྱིལ་ཆས་གལ་ཆེན་བསྐྱར་རྒྱུ་ལུགས་སྔོན་རྒྱག་པ་དང་། རྒྱ་བེད་ལས་གྲུ་  ཚན་རིག་དང་མཐུན་པའི་དང་བཀོད་གཏོང་བྱེད་པ། རྒྱ་གར་དང་རྒྱ་ལོག་གི་ཉེན་ལར་ཞིབ་བཤེར་བྱ་རྒྱུ་ལུགས་སྔོན་  རྒྱག་པ། ཉེན་ལ་ཤེས་འཕྲུལ་སྲུང་དུ་འགོག་སྲུང་བཅས་བྱས་ཏེ་མེད་གཞིའི་སྐྱིལ་བཀོད་གལ་ཆེན་གྱི་བེད་འཇགས་འགན་  ལེན་བྱེད་དགོས། རུས་ལུགས་ཡོད་ཀྱིས་ཉེན་སེལ་དོགས་སྐྱོབ་དང་བོགས་སེལ་བྱས་ནས་རིགས་འགག་གི་ཉེན་སེལ་  ལྟབ་སྐྱོབ་ཀྱི་སྐྱབས་ལུགས་གཅིག་ཟུར་བཀོད་གཏོང་དང་། ལྷ་ས་ནས་བཀོད་སྐྱིལ་འཕྲུལ་མར་སྐྱོད་བྱས་པ། རུས་ཆེའི་  དོགས་སྐྱོབ་བཅས་བྱས་ནས་བྱོང་གུན་སྐྱོད་ཚོགས་པོ་གཙོ་བོ་ཆེད་དུ་གཏོང་གང་ཐུབ་བྱེད་དགོས། རི་མས་ནད་སྔོན་འགོག་  ཚོད་འཛིན་དང་ཉེན་སྐྱོབ་བོགས་སེལ་གྱི་ལས་དོན་སྐྱི་འདྲེས་སེལ་རང་ལག་ལོ་བསྐྱབས་ཏེ་སྔོན་འགོག་དང་ཚོད་འཛིན་གྱི་  བྱེད་ཐབས་འགན་དོན་འཁྲོལ་ནན་པོ་བྱས་ནས་རི་མས་ནད་བསྐྱར་དུ་མི་ཐོན་པ་བྱེད་དགོས།</p>
<p>Bing Translation</p>	<p>The meeting emphasised that it is necessary to give accurate early warning, strict prevention, timely and accurate forecasting of meteorological data such as rain and water conditions, and strengthen early warning of secondary disasters. In particular, heavy rains, typhoons, floods, and mudslides in some areas have raised the level of early warning. Strengthen the construction of important embankments and protective facilities, and scientifically dispatch water conservancy projects. Strengthen the investigation of embankments and flood risks, detect dangerous and rapid protection, and ensure the safety of major infrastructure. We will make every effort to rescue and rescue, uniformly dispatch all kinds of rescue and disaster relief forces, and make early deployment, early deployment, and large-scale rescue to reduce all kinds of losses. Coordinate epidemic prevention and control, emergency rescue and disaster relief work, and strictly implement various prevention and control measures to prevent the recurrence of the epidemic.</p>

Because Cohere’s model is multilingual, the system returns good search results in many major world languages besides English, especially Spanish, French, and Chinese. It can also accept queries in any of the languages in its model. The quality of cross-lingual results

decreases as the language of the query moves further from major global languages – our impression of the quality of results from a Vietnamese query was not favourable. But we found that, in terms of similarity, entering a query in Chinese or English produces accurate results from the corpus in Tibetan. The multilingual functionality of the system, both in query parsing and translation of results, means the tool can be used by a general audience, is not limited to readers of Tibetan, and does not require any programming knowledge. Our user interface for the search system, which is hosted on a private university server, is available for use by the public and includes a few useful design elements: sliders for restricting searches in years and months, toggleable translation of both queries and results, and a toggleable option for machine-readable and human-readable results. Figure 2 is a screenshot of the homepage with the same query as in Table 1:

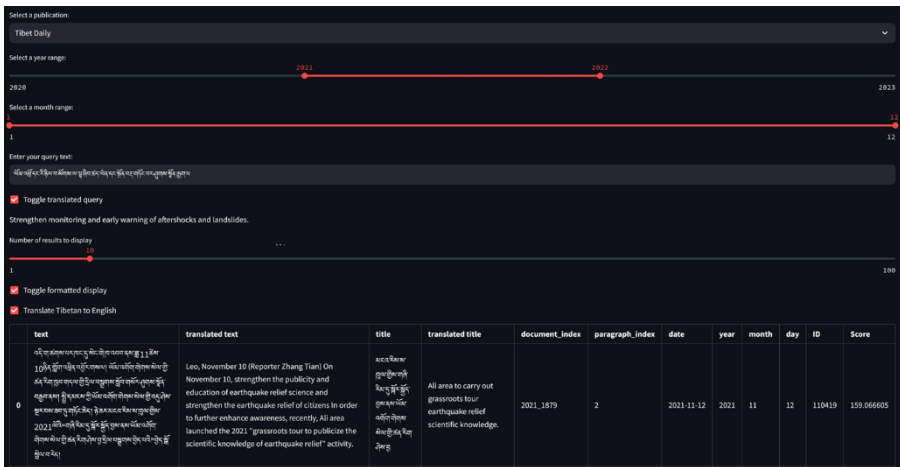


Figure 2 Semantic search homepage with query

We aim to add all digitised corpora from the Divergent Discourses project to the searchable system, which will provide the first ever wide-scope system for searching historical Tibetan newspapers – or any other set of Tibetan documents – using state-of-the-art NLP tools. Further, by integrating the spaCy tokeniser developed by Kyogoku *et al.* 2025 (in this issue), we can append context-sensitive keyword search functions to the semantic search function already developed. Integrating keyword and semantic search will dramatically simplify

the arduous process of searching for specific passages in digitised Tibetan corpora. Search systems like this are not computationally intensive to build and do not require extensive knowledge of software design, and to do so will only get easier as language models for Tibetan improve.

Finally, we ought to draw attention to a few reasonable improvements to the tool. The most obvious is to convert the interface to a proper web-based language like JavaScript and deploy it securely on a standard website. We also want to integrate a keyword search functionality, like TF-IDF, alongside the semantic search system in the same interface, to allow users to easily switch between short and long queries, and between identical and similar results. We additionally hope to implement the ability to subselect returned results and re-sort them, filtering or removing unwanted results from the initial search. The system costs roughly US\$2 per month to maintain, between server costs, API requests, and usage-based subscriptions to services like Bing Translate; higher traffic will naturally tend to increase those costs, though not into outrageous amounts. However, because the understanding spaces are properties of external corporate entities, and because those entities change, recalibrate, decommission, or close off their current services or models, it is not possible to be certain how long access and functionality will remain available. As NLP solutions for low-resourced languages like Tibetan gravitate to transformer-based technologies, the reliance of these solutions on external services and corporations could make their long-term sustainability increasingly uncertain.

## Bibliography

Blei, David M., Andrew Ng, and Michael I. Jordan

“Latent Dirichlet Allocation.” *Journal of Machine Learning Research* 3, 2003. pp. 993–1022. Available online at <https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf> (accessed January 26, 2025).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.

“BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *North American Chapter of the Association for Computational Linguistics*, 2019. Available online at <https://arxiv.org/pdf/1810.04805> (accessed January 15, 2025).

Faggionato, Christian and Marieke Meelen

“Developing the Old Tibetan Treebank.” In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Varna: INCOMA, 2019, pp. 304-312. [doi:10.26615/978-954-452-056-4\\_035](https://doi.org/10.26615/978-954-452-056-4_035)

Grootendorst, Maarten

“BERTopic: Neural Topic Modeling with a Class-based TF-IDF Procedure,” *arXiv preprint*, 2022. [doi:10.48550/arXiv.2203.05794](https://doi.org/10.48550/arXiv.2203.05794).

Hackett, Paul G.

“Automatic Segmentation and Part-Of-Speech Tagging For Tibetan: A First Step Towards Machine Translation.” In *Proceedings of the 9th Seminar of the International Association for Tibetan Studies*, 2000, pp. 1-18. Available online at <http://hdl.handle.net/10022/AC:P:10471> (accessed on December 18, 2024).

Jiang Di

“A New Perspective for Modern Tibetan Machine Processing and its Development: an Insight into the Method of

Computerized Automatic Understanding of Natural Languages in Terms of Chunk Identification." In Xu, B., MS. Sun, and GJ. Jin (eds.) *Some important problems in Chinese language information processing*. Beijing: Science Press of China, 2003, pp. 438–448.

Kyogoku, Yuki, Franz Xaver Erhard, James Engels, and Robert Barnett  
 "Leveraging Large Language Models in Low-resourced Language NLP: A spaCy Implementation for Modern Tibetan",  
*Revue d'Etudes Tibétaines* 74, 2025, pp. 187–222.

Meelen, Marieke, Élie Roux, and Nathan W. Hill  
 "Optimisation of the Largest Annotated Tibetan Corpus Combining Rule-Based, Memory-Based, and Deep-Learning Methods," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* 20 (1), 2021, pp. 1-11. [doi:10.1145/3409488](https://doi.org/10.1145/3409488).

Schwartz, Ronald, and Robert Barnett  
 "Religious Policy in the TAR, 2014–24: Topic Modelling a Tibetan Language Corpus with BERTopic," *Revue d'Etudes Tibétaines* 74, 2025, pp. 221–260.

Spärck Jones, Karen  
 "A Statistical Interpretation of Term Specificity and its Application in Retrieval," *Journal of Documentation* 28 (1), 1972, pp. 11–21. [doi:10.1108/eb026526](https://doi.org/10.1108/eb026526).

Sun Yuan, Liu Sisi, Deng Junjie, and Zhao Xiaobing  
 "TiBERT: Tibetan Pre-trained Language Model," *arXiv preprint*, 2022. [doi:10.48550/arXiv.2205.07303](https://doi.org/10.48550/arXiv.2205.07303).

Tao Jiang, Li Jia, Ma Cao Wan, and Jia Hao Meng  
 "The Text Modeling Method of Tibetan Text Combining Word2vec and Improved TF-IDF." *Journal of Physics: Conference Series* 1601, 2020. [doi:10.1088/1742-6596/1601/4/042007](https://doi.org/10.1088/1742-6596/1601/4/042007).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, and Aidan N. Gomez

“Attention is All You Need,” *Advances in neural information processing systems* 30 (1), 2017, pp. 261–272. [arXiv:1706.03762](https://arxiv.org/abs/1706.03762).

Zhang, Jiangyan, Deji Kazhuo, Luosang Gadeng, Nyima Trashi, and Nuo Qun

“Research and Application of Tibetan Pre-training Language Model Based on BERT.” In *Proceedings of the 2022 2nd International Conference on Control and Intelligent Robotics*, 2022, pp. 519–524. [doi:10.1145/3548608.3559255](https://doi.org/10.1145/3548608.3559255).

