View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Technical Disclosure Common

Technical Disclosure Commons

Defensive Publications Series

April 24, 2017

Identifying Similar Meaning In Word Sequences

Jakob Uszkoreit

Follow this and additional works at: http://www.tdcommons.org/dpubs_series

Recommended Citation

Uszkoreit, Jakob, "Identifying Similar Meaning In Word Sequences", Technical Disclosure Commons, (April 24, 2017)
http://www.tdcommons.org/dpubs_series/480

This work is licensed under a Creative Commons Attribution 4.0 License.

This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.

Identifying Similar Meaning In Word Sequences

ABSTRACT

A system for determining semantic similarity of phrases (word sequences) uses weak supervision of neural networks to generate an embedding space for determining phrase similarity. The training of the neural networks uses input from two orthogonal sources: click distribution of queries and phrase translations.

DISCLOSURE

Language, both in queries and documents, can express very similar meaning in a wide variety of ways. This problem is at the heart of application problems like query understanding for search features or question answering both on the end of understanding questions and of understanding possible answers.

One way of identifying sequences of words with similar meaning but different expression is representing multiword, variable-length sequences of words (such as queries or sentences) as a high dimensional vector that represents the “entirety” of the meaning of the sequence. In other words, the sequence of words can be embedded into a vector space, otherwise known as an embedding. Once represented by an embedding, a system may use neighborhoods defined over notions of distance, such as Euclidian distance or cosine similarity, to identify synonymy or high similarity of different sequences. With such a measure of similarity, systems, such as search systems, may use the embeddings to improve question answering or query understanding. For example, a system may compare a given understanding (e.g. a sequence projected back into query space by mapping it onto canonical queries) with a query at hand. In another example, the embeddings allow canonicalization of the real query space into a smaller, easier to cover space of equally expressive queries.

Implementations provide a system for generating an embedding model that can be used to generate high-dimensional vectors used to identify sequences of words that have a similar meaning but different expressions. The system uses weak supervision, making the system scalable. The system uses a combination of orthogonal sources of training data for weak supervision. Examples of such sources are statistics of user’s clicks on web results (or other search features) after issuing a query or other forms of user behavior, such as execution or pre-execution abandonment of voice actions, clicks on documents when presented an answer or lack thereof, etc., as well as parallel data. Parallel data is sentences, queries or phrases and their translations into various languages.

FIG. 1 is a block diagram of a word sequence semantic similarity system in accordance with an example implementation. The system 100 may be used to generate training examples to train a word sequence embedding model, which can then be used to find semantically related word sequences. For example, the sequence embedding model may be used to enhance query understanding and question answering in a search system. Implementations need not include every element or functionality described with regard to system 100.

The word sequence semantic similarity system 100 may be a computing device or devices that take the form of a number of different devices, for example a standard server, a group of such servers, a rack server system, or in a personal computer, for example a laptop computer. The word sequence semantic similarity system 100 may include modules, such as an indexing engine 126, a search engine 120, a sequence embedding model 140, and a sequence similarity training engine 150. The indexing engine 126 may maintain an index 130 of search items 134 for use by the system 100. The search items 134 may be any items in a collection. For example, search items 134 may be documents available over the Internet, documents available on an Intranet, mobile applications, web applications, or plugins via a web store, songs in an online music store, items for sale via an online marketplace, etc. The indexing engine 126 may process the search items 134 and update index entries in the index 130, for example, using conventional or other indexing techniques.

The sequence similarity training engine 150 may generate training examples to train the sequence embedding model 140. The sequence embedding model 140 may include an encoding network and/or a regression network. An encoding network is a multilayer neural network of various possible architectures including simple, feedforward networks, LongShortTerm Memory (LSTM) networks and other forms of recurrent neural networks. An encoding network takes as input is a single, fixed or variable length sequence of words over a fixed alphabet. Their output is effectively a higher dimensional, real-valued embedding of the input sequence. Typically, the vector space into the system embeds words or sentences is low-dimensional, when compared to the much higher-dimensional, trivial representation of words, such as one-hot vectors over a fixed vocabulary like in the classic vector space model of information retrieval. The encoding network can use various sources of weak supervision. A regression network neural network, again possibly of the various architectures mentioned above, takes as input two or more of the higher-dimensional embedding vectors (e.g., from the Encoding network). A regression network outputs one of a variety of measures of similarity (e.g., Euclidian distance or cosine similarity or another similarity measure).

The combination of two encoders in the sequence embedding model 140 (possibly instances of the very same network with tied parameters) and a regression network can be trained using back-propagation and its various variants (BPTT, etc.) when presented with two or more input sequences and an expected output value. Multiple regression networks can be combined with the same set of two or more embedding networks to allow using multiple different kinds of such data consisting of two or more input sequences and one expected output value.

To train the sequence embedding model 140, the sequence similarity training engine 150 may generate training tuples. A training tuple is a pair of fixed or variable length sequences of words and the expected output value for the pair. The expected output value is an indication of how similar (e.g., semantically) the two sequences of words are. In some implementations, the expected value may range from zero to one, with zero indicating no semantic similarity and 1 indicating highest semantic similarity. The sequence similarity training engine 150 may obtain training tuples from various sources including from search records 138 and translation items 132.

Search records 138 may keep statistics and metadata about queries and responsive search items submitted to the search engine 120 without storing any personally identifying information, or may anonymize such information, for example by storing a city or zip code for the query information only. For example, the search records 138 may include a click distribution for a query, or information from which such a distribution can be determined. The click distribution represents the search items 134 (or search features) that have been selected after being displayed to the query requestor. In other words, when a user using client 170 submits a query 182 to the search engine 120, the search engine 120 may provide search items 134 responsive to the query 182 as result 184. The search records 138 may include data that indicates what search items 134 were included in the search result 184 as well as which items the user of client 170 selected. As indicated above, such data may be stored without personally identifying information or an anonoymized version of the information. Word sequence semantic similarity system 100 may be in communication with client(s) 170 over network 160.

The sequence similarity training engine 150 may use the search records 138 to find statistics (i.e., an expected value) of users' clicks on web results or other search features after issuing a query. The sequence similarity training engine 150 may capture the statistics as the similarity of distributions of user clicks on links to documents in the result set 184 generated by search engine 120 for the query 182 the user issued or on buttons executing other functions shown in response to that query 182. The sequence similarity training engine 150 may set the expected value as a measure of similarity of the click distributions between two or more queries, which form the two or more inputs to the embedding networks. In other words, the two or more queries and the measure of similarity of the click distributions are provided as a training tuple.

In order to find the two or more queries, the sequence similarity training engine 150 may sample from queries that share at least one document in their result set or share at least one rendered search feature a user might click on. Another strategy the sequence similarity training engine 150 may use is to sample queries that share at least one or more terms but are not identical. The expected value associated with any pair of queries is the similarity of click distributions over all possible results and search features. Such distributions can be determined between many or ideally all times this query was issued by any user.

In addition to search records 138, the sequence similarity training engine 150 may use translation items 132 to generate training tuples. Translation items 132 may represent human translations of sequences of words mined automatically using various techniques, such as those disclosed in U.S. Patent Pub. 2012/0047172. Here the expected output is a measure of "translatedness" of the two word sequences in different languages, forming the inputs to the embedding networks. Translatedness can be expressed in binary (1 if the two inputs are translations or each other, 0 if not) or a fuzzier metric, e.g. a so-called forced-alignment score obtained from an existing machine translation system.

To generate training tuples from the translation items 132, the sequence similarity training engine 150 may use "positive" (truly translated) word sequence pairs (positive training tuples) sampled from the set of known translation pairs mined using methods mentioned above. The sequence similarity training engine 150 can also generate "negative" word sequence pairs (negative training tuples) by selecting a true translation pair first, and then finding alternative sequences for one of the members of the pair by searching a large corpus of word sequences for a sequence that has some similarity with the original member of the pair to be replaced (e.g. where one or a few but certainly not all words are the same). Other similar techniques are possible.

In some implementations, the sequence similarity training engine 150 may sample training tuples according to heuristic strategies to make learning the embeddings possible and to optimize quality of the learning. Using training tuples from multiple sources provides a benefit to the trained sequence embedding model 140. For example, click distributions have specific patterns of errors in query similarity: because many documents answer various different, but related questions (e.g. how tall a person is and how old that same person is might be answered in the same web pages, therefore users asking either of these questions are likely to click on the same documents) models trained solely on clicks will not be able to reliably differentiate between, say, questions related in this manner. Translations, even from noisy data sources such as large-scale automatic mining, do not exhibit this problem (but others). Together they form pretty orthogonal source of information, likely addressing the other's shortcomings.

In some implementations, the sequence similarity training engine 150 may use training tuples from various sources in alternation to derive an error term based on the deviation of the output of the network combination given its current parameters from the expected value. Based on the error term, back propagation computes a weight update to be applied to the embedding and regression networks' parameters. Typically in a setting with multiple kinds of data, or supervision, only one of the kinds of expected values is present for any given pair or tuple of inputs. In that case, back propagation can only be executed using the error term for that specific output. In cases where multiple expected values are present, multiple back propagation steps can be executed or a joint back propagation step can be determined based on a weighted combination of the resulting error terms. The latter is basically equivalent to providing a third layer of neural network with fixed parameters to combine the produced values for each kind of expected value.

Of course, in some implementations the sequence similarity training engine 150 could train similarity measures independently on clicks and translations and mix them. The sequence similarity training engine 150 has the ability to combine orthogonal data from clicks and translations along with other possible sources of weak supervision (e.g., click-reinforced phrasal synonyms, subsequent queries issued, etc.) in a single network architecture.

Once the sequence embedding model 140 is trained, it is able to generate an embedding for a sequence of words. An embedding model is used to represent things, such as the sequence of words, in a feature vector. The system 100 can use these emebddings to find similar sequences of words, whether in search items 134 or queries similar to a query 182.

In some implementations, the indexing engine 126 or sequence similarity engine 122 or another module not shown may use the sequence embedding model 140 to generate an embedding for a sequence of words found in search items 134 or a query 182. In some implementations such an embedding can be stored in the index 130. For example, the indexing engine 126 may obtain an embedding of a multi-word phrase to be indexed and the embedding may be stored with the phrase in the index 130. The system 100 may use this embedding space to determine similarity between search items or queries by using the distance between points.

Word sequence semantic similarity system 100 represents one example configuration and other configurations are possible. In addition, components of system 100 may be combined or distributed in a manner differently than illustrated. For example, the sequence similarity training engine 150 may use execution or pre-execution abandonment of voice actions or clicks on documents when provided an answer to generate training tuples. This may be in addition to or instead of the other sources of training tuples discussed above.

FIG. 2 illustrates a flow diagram of an example process for training an embedding model to learn an embedding usable in a word sequence semantic similarity metric, according to an implementation. Process 200 may be performed by a word sequence semantic similarity system to generate training tuples to train a sequence embedding model, such as model 140 of FIG. 1. Process 200 may begin with the system selecting query pairs from search records that share at least one document or rendered search feature in their result set (205). Alternatively or additionally, the system may select query pairs that share at least one term but are not identical (210). The system may use heuristic sampling to select the pairs from all possible pairs. The system may, for each query pair selected (e.g., sampled), determine the similarity of click distributions over all possible results and search features to determine an expected value for the query pair (215). Each query pair and its respective expected value is a training tuple. The system may use the training tuples as positive training examples for the sequence embedding model (220). The system may also select positive translation pairs for a sequence of words (250). These pairs of words may be positive training tuples. The system may also generate negative translation pairs using the sequence of words in the positive training tuples (255). The system may, for each pair, calculate an expected value as a measure of translatedness (260) and use the training tuples to train the sequence embedding model 140 (265), as described above.

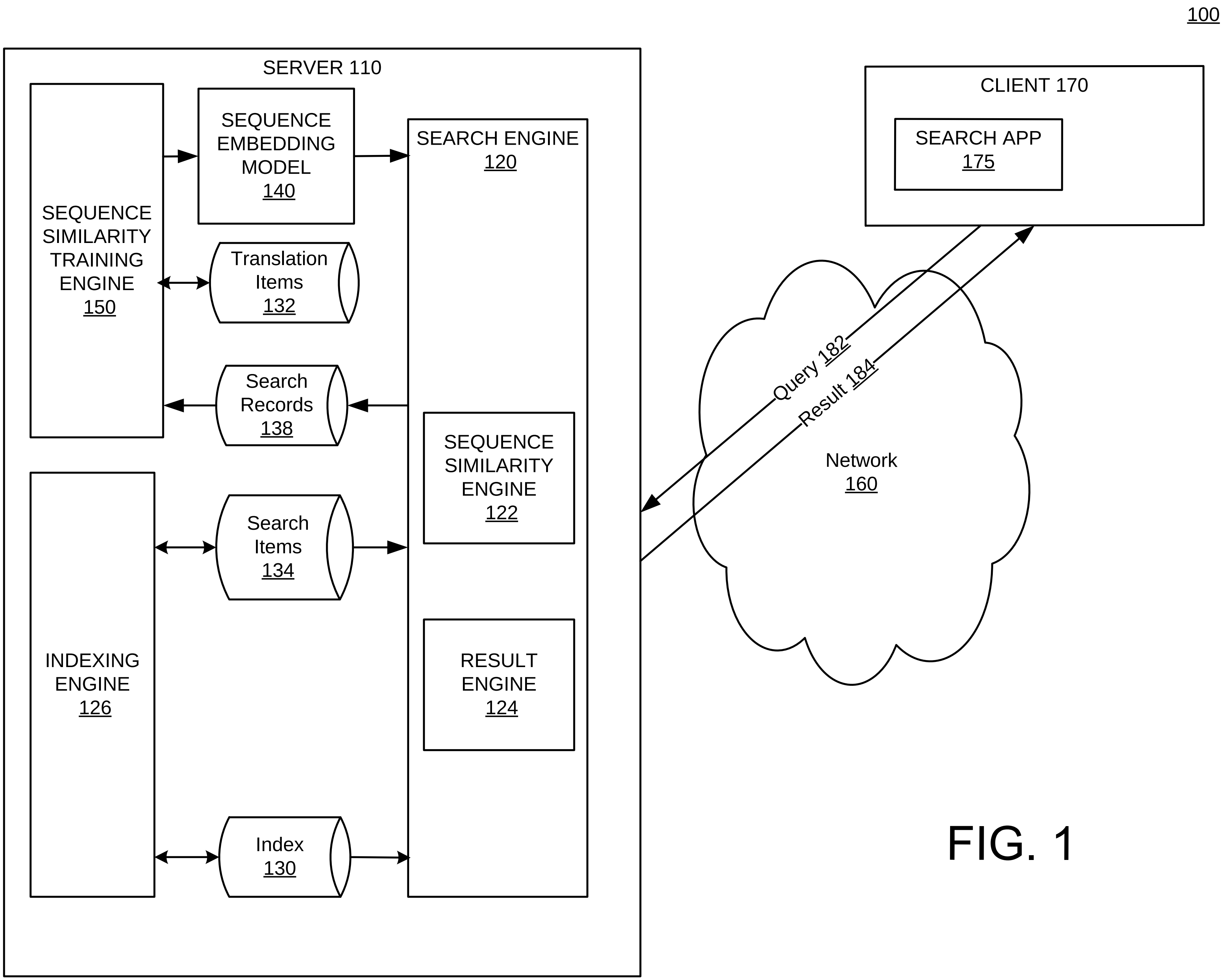
100
SERVER 110
SEQUENCE SIMILARITY TRAINING ENGINE 150
SEQUENCE EMBEDDING MODEL 140
Translation Items 132
Search Records 138
SEARCH ENGINE 120
SEQUENCE SIMILARITY ENGINE 122
Search Items 134
RESULT ENGINE 124
INDEXING ENGINE 126
Index 130
CLIENT 170
SEARCH APP 175
Query 182
Result 184
Network 160

FIG. 1

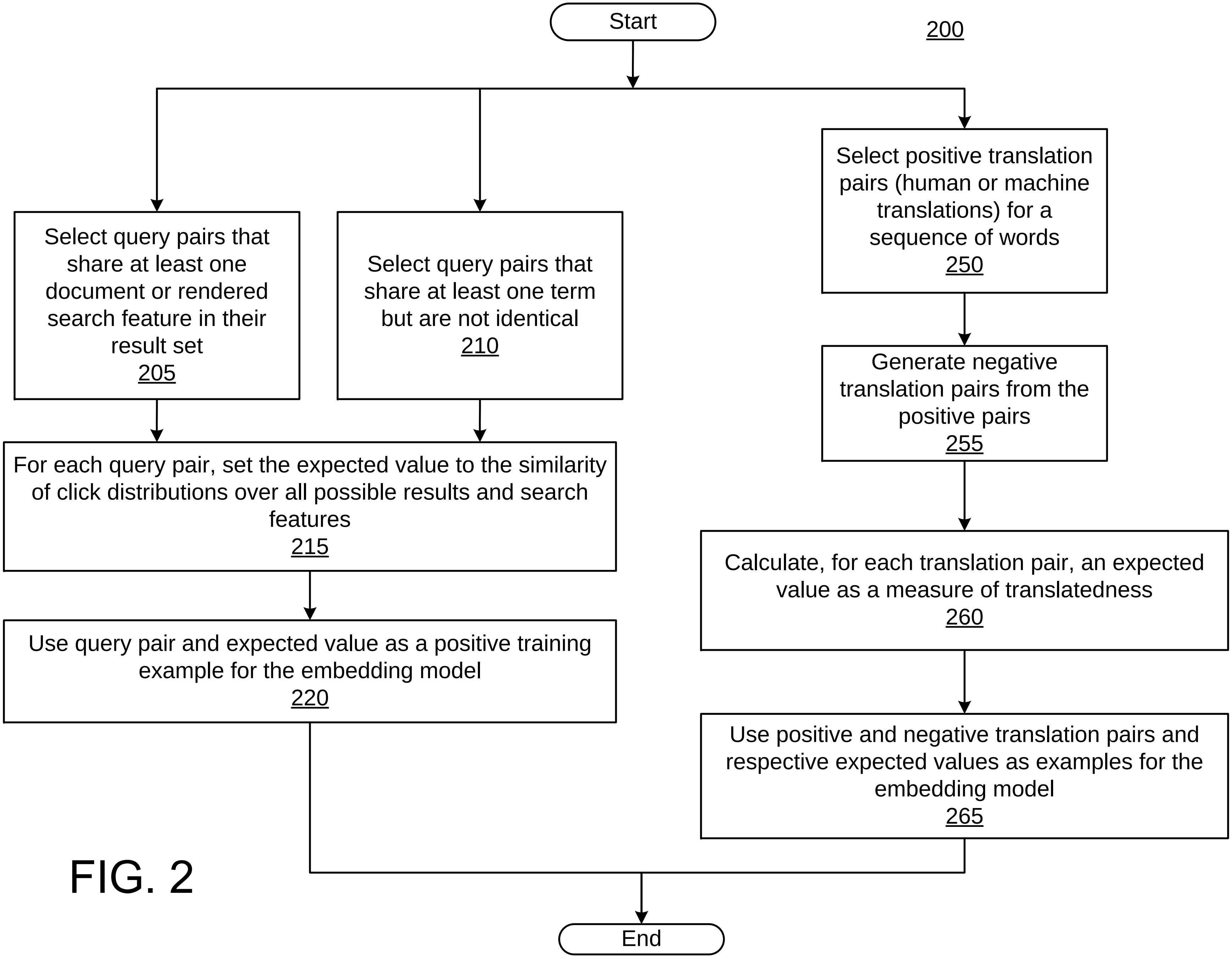
Start
200
Select query pairs that share at least one document or rendered search feature in their result set
205
Select query pairs that share at least one term but are not identical
210
For each query pair, set the expected value to the similarity of click distributions over all possible results and search features
215
Use query pair and expected value as a positive training example for the embedding model
220
Select positive translation pairs (human or machine translations) for a sequence of words
250
Generate negative translation pairs from the positive pairs
255
Calculate, for each translation pair, an expected value as a measure of translatedness
260
Use positive and negative translation pairs and respective expected values as examples for the embedding model
265
End

FIG. 2