A teacher is required to submit the class notes he/ she prepares for the class to the contextualizer. In addition to this, the system also requires an image of a real life object or scenario. This gives a direction in which the context of concepts is to be searched. Hence, the teacher submits the notes and an image to the contextualizer using a mobile phone. The system extracts main concepts from the teacher's notes. From the main concepts, it identifies the core teaching concepts the teacher wants to teach in the entire class. From the image, it detects the object present. In case there are multiple objects, the system pulls out the central theme of the image. In some cases, a direct linkage between some of the core learning concepts and the theme exist. However, in most of the scenarios, it is observed that the idea or the image is too general and the core teaching concepts were too specific to the subject. In such cases, although the two might be related, contextualizer could not find any relation. Hence, the system tries to find out the terms similar to the image theme. These terms are either synonyms of image theme, or are closely related to the image theme. The contextualizer ends up collecting a large number of such terms. Out of these, it takes into consideration only the terms that are somehow related to the subject.

With the pairs of core teaching concepts and various terms similar to or related to image theme, the system searches for web resources available. This generated a huge number of search results, some of which are highly relevant while others are not at all related to the concepts or image. Contextualizer then applies a filter to get results that best represents the context of the concepts. Each article appearing on the search engine result page is passed through initial filtering. The information present is also assessed for semantic relevance with the two inputs the contextualizer takes. In the end, contextualizer recommends the most relevant context that is related to both- the concepts and the image.

The architectural diagram in figure 1 depicts the integral components of contextualizer. We explain its working in the following subsections:

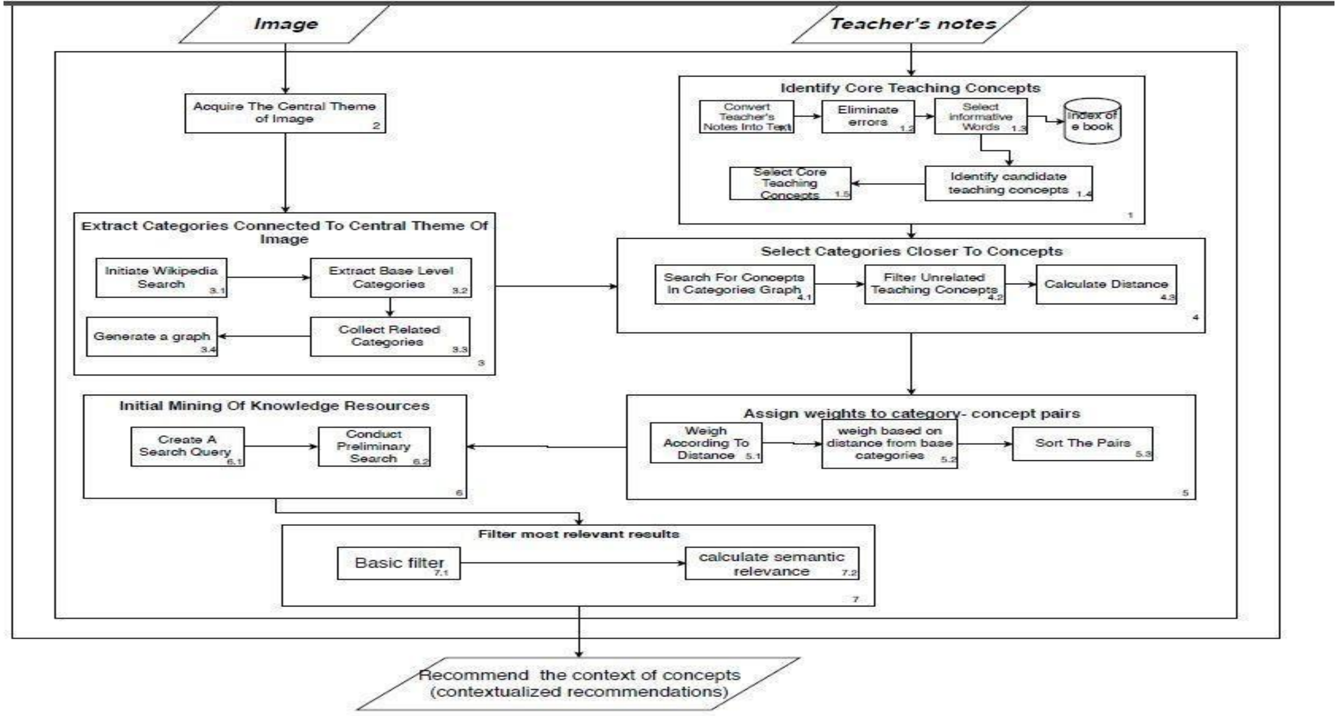


Image
Teacher's notes
Acquire The Central Theme of Image 2
Identify Core Teaching Concepts
Convert Teacher's Notes Into Text
Eliminate errors 1.2
Select informative Words 1.3
Index of e book
Select Core Teaching Concepts 1.5
Identify candidate teaching concepts 1.4
1
Extract Categories Connected To Central Theme Of Image
Initiate Wikipedia Search 3.1
Extract Base Level Categories 3.2
Generate a graph 3.4
Collect Related Categories 3.3
3
Select Categories Closer To Concepts
Search For Concepts In Categories Graph 4.1
Filter Unrelated Teaching Concepts 4.2
Calculate Distance 4.3
4
Initial Mining Of Knowledge Resources
Create A Search Query 6.1
Conduct Preliminary Search 6.2
6
Assign weights to category- concept pairs
Weigh According To Distance 5.1
weigh based on distance from base categories 5.2
Sort The Pairs 5.3
5
Filter most relevant results
Basic filter 7.1
calculate semantic relevance 7.2
7
Recommend the context of concepts (contextualized recommendations)

Fig. 1: Architectural diagram of contextualizer

Step 1: Preprocess teacher's notes

The first step of teaching with context for the teacher is to upload the topic notes. Teachers upload the notes either by taking a photo of handwritten notes or in the form of pdf. To preprocess the notes, it is important that we convert them into a form that the computer can process and understand. So, in order to preprocess and analyze, the system converts notes from any format to text. Also, teacher's notes contain many concepts to be discussed in class. However, in handwritten notes, there may be many misspell words and noise. The notes are first preprocessed to eliminate noise and to deal with spelling mistakes. Following are preprocessing steps applied to the notes uploaded to the system by the teacher:

1.1 Convert teacher's notes into text: To convert notes in a format that the computer can process, the contextualizer uses an Optical Character Recognition (OCR) system. With the help of an OCR, the contextualizer transforms handwritten notes into text format. It only identifies characters and ignores symbols like ":" Now we have a set T of words that are there in the notes.

1.2 Rectify spelling errors: To deal with spelling errors, the system uses a spell-checker (Perkins J., 2010). The spell checker takes every word W from T and searches for it in the dictionary. If W is present in the dictionary, it is left as it is. If W does not match any dictionary word, it is assumed that there is a spelling mistake. Spell checker also auto corrects the spelling mistakes by changing it to the word that is a closest match to W.

Step 2: Extract Core teaching concepts

With the notes, the teacher also inputs the name of the relevant e-book for the subject he is teaching. Since what we get in T is a set of general words and no phrases, there are high chances of losing meaningful information. It is important to get phrases and terms relevant to the subject. For instance, "*first come first serve*" is a scheduling algorithm. But, in T you will get separate words "first", "come", "first", "serve". These four words are present in T, but lose their actual meaning. Since we are dealing with teacher's notes, there will many such occurrences. Hence, we use index of an e book to get phrases specific to the subject. Using the index too leads to a huge number of index terms. These are reduced with the help of clustering. In the end of this step, the system is finally able extracts only the fundamental concepts embedded within the notes. Here are the steps to get the core teaching concepts:

2.1 Select informative words: There are a lot of words present in T. Some of them carry meaningful information while others may not be of any value in context of giving some information. The system initially filters out the least informative words and moves on with words that carry some information. Since it is a widely accepted fact that nouns convey most of the crucial information present in notes (Kaur J and Gupta V.2010), the contextualizer simply selects all nouns present in T. The set of nouns thus selected are stored in the set $T_{N.}$

$T_N = \{W: W \epsilon T \ \& \ W \text{ is a noun}\}$

***2.2 Refer E-book*:** The system maintains an index of most of the e-books of computer science courses that are available online. The teacher is required only to enter the name of the book he wants to be referred. Index of a book contains the list of topics along with the page numbers on which the topic is explained in detail. For every W in T_N, the system searches the entries in the index. This helps in conversion of fragmented words into meaningful phrases.

2.3 Select candidate teaching concepts: Contextualizer now chooses all the single or multi-word phrases present in the index that match with the words in T_N. It also selects the index phrases that contain any W as a part. These represent the candidate fundamental teaching concepts stored in C

C= {I: W=I or W is a substring of I }

If there are general terms in the notes, C is going to have a large number of index terms. For example, if the teacher wants to teach some topic of operating system. The notes may contain words like "system" or "memory". The system will end up selecting all index terms having such general terms. Hence there is a need to reach the core teaching concepts with the use of candidate teaching concepts.

2.4 Identify core teaching concepts: C contain index terms corresponding to every noun present in the teacher's notes. However, there are cases when the teacher is just referring to what he taught in previous class. There are also chances that the teacher is just citing a concept that is somehow related to the core teaching concepts, but not the part of the concepts he is planning to teach. There is a need to remove such anecdotal mentions to keep the results of contextualizer related to the core concepts only. For this, the contextualizer creates clusters of all index entries in C with respect to the corresponding page numbers. If two concepts are present in near proximity in terms of page numbers, there are high chances that they are closely related to each other. Hence clustering is done on the basis of page numbers.

2.4.1 The first concept in C makes a cluster.

2.4.2 If the distance between page number of the first candidate concept and the next is not more than 10 pages, the concept is added to the same cluster.

2.4.3 Else, the system puts the candidate teaching concept in a new cluster.

2.4.4 This process is repeated until all the candidate concepts in C are clustered.

A dense cluster having nearby pages indicates a core theme of the class. On the other hand, a sparse cluster usually indicates noise or anecdotal mention. Thus, the contextualizer chooses the densest cluster from all the clusters. In case there are two or more clusters with exactly equaly candidate concepts, all of them are taken as D to avoid loss of information. The topics/ index phrases appearing in the densest cluster represent core teaching concepts [TC].

TC= {c: c ϵC & c ϵ D}, here D is the densest cluster.

The table below shows the candidate teaching concepts that contextualizer gets from one paragraph of software engineering notes. Corresponding page numbers at which the topic is present is also given.

S. No.	Candidate teaching concept	Page no.
1.	Requirement specification	64, 77

	Coverage analysis tools	235	
3.	Boundary Value Analysis	210	
4.	Subsystem analysis	197	
5.	Facilitated application specification technique	15	
6.	Object oriented requirement analysis	107, 139	
7.	Structured analysis	103	
8.	Non functional requirements	66	
9.	Requirement elicitation	68	
10.	Functional requirement	65	

Table 1: candidate teaching concepts and corresponding page numbers

One the basis of clustering described in step 2.4, the system divides the concepts mentioned in table one in the following clusters:
Cluster 1: [15]
Cluster 2: [64, 65, 66, 68, 77]
Cluster 3: [103, 107]
Cluster 4: [139]
Cluster 5: [197]
Cluster 6: [210]
Cluster 7: [235]
By choosing the densest cluster of the candidate teaching concepts, the candidate concepts belonging to page numbers in cluster two qualify to become core teaching concepts. Following are the core teaching concepts from table 1:
{Functional requirements; Non - functional requirements; Requirement specification; Requirements elicitation}

Step 3: Acquire the central theme of image

The system now has the core teaching concepts that are present in the teacher's notes. Second input to the contextualizer is an image. Almost every course of computer science is applicable in a wide variety of real life objects and scenarios. However, in order to keep the contextualization directed and interesting, the system takes as input a particular scene/ object/ environment rather than giving random real life examples. So, the teacher clicks any picture he/ she thinks would be interesting for the students and submits it to the contextualizer.

The next task is to gather all the useful information present in the image. The image a teacher uploads may be of a single object, or it may have multiple objects. Sometimes, the teacher may also capture an entire scene comprising of a real life situation and submit it to contextualizer.

The contextualizer then processes the image attempts to acquire the central theme of the image. The central theme is usually one idea that represents the entire image. If it is an object, the idea is simply the name of the object. In case there are multiple objects, it would be a phrase or some phrases describing the entire scene. Let I be the idea/ set of ideas the system extracts from image.

Step 4: Identify categories connected to central theme of image using wikipedia

The theme accurately represents the image. But, there are less chances of finding a connection between the core teaching concepts and general idea of the image. This is because the terms in TC are subject specific terms picked up from index of the e book. The image theme on the other hand comprises of general terms that are more related to real life. Hence, to increase the probability of digging out meaningful connections from the web resources, the system widens the central theme of the image. It does so by finding some more closely related words/ terms. To gather terms that are actually related to the idea of the image, the contextualizer uses wikipedia. Wikipedia is an online encyclopedia freely available to all. The system utilizes these specialized concepts available in the form of categories mentioned at every wikipedia page as hyperlinks.

With the use of I we get in step 3, the system now creates a cluster of related terms using wikipedia.

4.1 Initiate the wikipedia search: To initiate the wikipedia search, the contextualizer first collects the addresses of wikipedia pages directly related to the image theme/ object. The system generates Uniform Resource Locators (URL's) of wikipedia page corresponding to every idea in I obtained from step 3. U_I represents a set that contains the list addresses of wikipedia pages corresponding to I.

U_I= {URL_1, URL_2, URL_3, ….URL_k}; where k is the number of ideas in I.

4.2 Extract base level categories from U_I: Every wikipedia page has category/ categories attached with each page. These categories can be viewed as terms or phrases directly related to the title of the wikipedia page. So, if we traverse the wikipedia categories, we can get a cluster of closely related terms. The second step towards this is to extract the base level categories. For every URL in U_I, the contextualizer gets all the categories and stores them in a list U_{Icat}.

U_{Icat}= {cat_1, $cat_{2…..}$ cat_m}; here, cat represents the wikipedia categories and m >=k.

4.3 Collect related categories by traversing selected wikipedia links: With the base level categories, the system now pulls out several other related categories. This will give a set of terms/ phrases that are somehow related to each other. Hence, the contextualizer now populates UI_{cat} with other categories interlinked with each other. To accomplish this, it keeps on visiting the wikipedia pages linked via categories in the following manner:

4.3.1 Reaching the wikipedia pages linked: The system visits the URL corresponding to every category present in UI_{cat}. At every wikipedia page, there will be one or more categories present. The categories are added to UI_{cat}. The process can turn into infinite searching and adding, or may result in collection of all /categories available at wikipedia. So, there is a need to define when to stop.

For every category in UI_{cat} repeat the following:

- Visit the wikipedia page corresponding to category
- Store the categories present on the page in U_{Icat}
- Repeat the same for i iterations.

Let U_{I1} represent the URLs in U_I.

Similarly, U_{I2} stores the URLs of categories present on the URLs in U_I and so on.
So,
U_{Ii} represents the URLs corresponding to the i^{th} iteration. Here, i represent the depth of search following the wikipedia links.

For the sake of illustration, we take the value of i as 10. So, the system visits wikipedia pages only till the depth of 10 levels. Now, U_{Icat} contain categories from all URLs starting from U_{I1} to U_{I10}.

4.3.2 Update U_{Icat}: Starting from the URLs available in U_I to U_{I10}, the system extracts categories available at every wikipedia page present in the sets using the method available in step 4.2.

4.3.3 Limit the number of categories: In case U_{Icat} has more than 500 terms, the system selects first 500 categories starting from U_{I1} traversing down. This threshold is set to tackle overload of terms that may occur in some cases. Else, if there are less than 500 terms after traversing up to the depth U_{I10}, the system moves on with all the categories available.

4.4 *Create a graph to see the connections:* The system then creates a graph that represents the connections between categories.

4.4.1 Generate a graph: With the help of a recursive function, the system generates a graph G_{cat} of the wikipedia categories available in U_{Icat}.

4.4.2 Eliminate redundancies: The graph is checked for redundancies of categories. If a category appears more than once, the one near to the base category is stored. Only one category is kept and the rest are deleted.

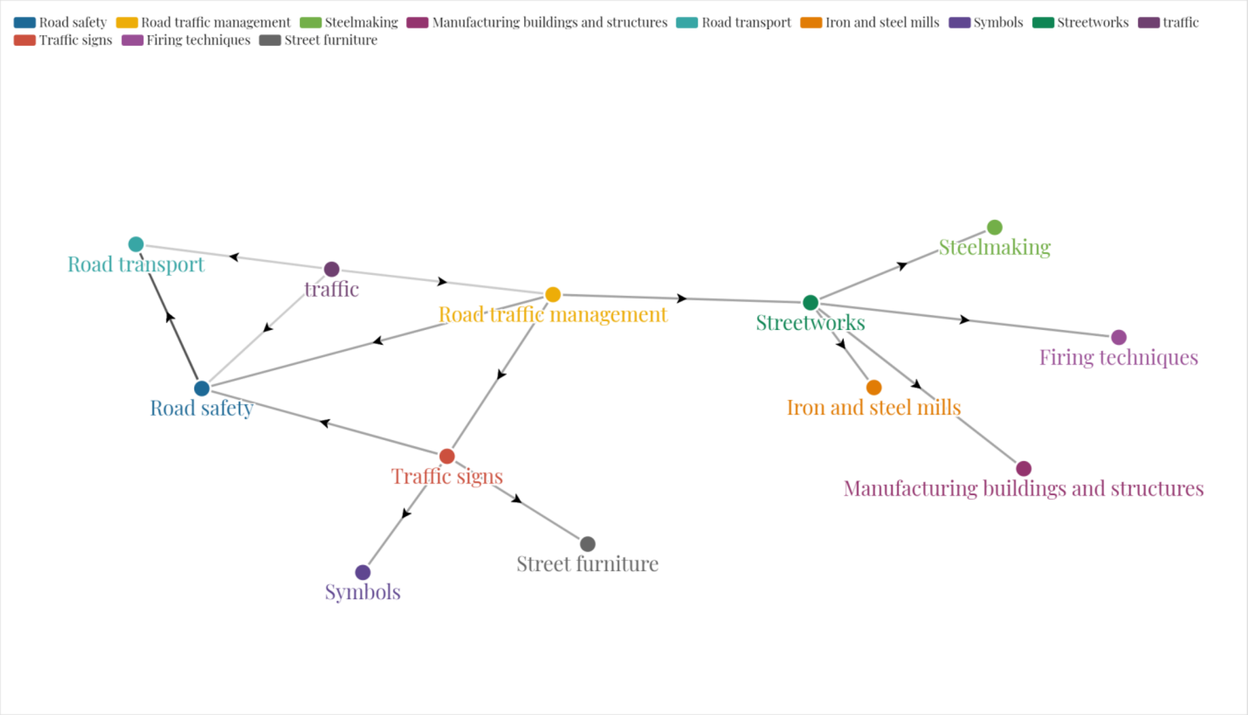
4.4.3 Remove self loops: The system also checks for loops in the graph. Suppose

Every graph thus generated is represented as a directed graph G, where

$G= \{V, E\}$

Here, V is a set of wikipedia categories available in UI_{cat} and E is the set of edges. Each directed edge represents that the target vertex is present as a category in the wiki page of the source vertex.

Figure 1 shows a sample graph starting from the image idea- “road traffic management”. For illustration, the extraction of categories is limited only to 12 categories. Every node the graph represents a wikipedia category. Directed edges between two vertices indicate the target node is present in the wikipedia page of source node as a category.



Step 5: Connecting categories and concepts

By now, the system has a collection of some 500 categories that are either directly or indirectly linked with each other through wikipedia. The categories in the base level are closely related to each other and the title of the wikipedia page too. However, as we go down, the relatedness usually decreases. To keep the search confined only to highly related or similar terms derived from categories collected so far, the system uses a weighing mechanism.

Also, searching for web resources using all the 500 categories combined with Core teaching concepts in TC is inefficient. Such a system would give diverse and irrelevant results. Hence to improve the accuracy and to suggest real life applications of the concepts present in the image, the contextualizer also tries to find out categories that are closely related to the categories in G_{cat}. Thus, the contextualizer now finds out how much near or far every category in the graph G_{cat} is to the concepts the teacher plans to teach in class. Following are the steps for filtering only the categories that are somehow related to the core teaching concepts in TC.

5.1 ***Search for concepts in categories graph:*** For every core teaching concept present in TC, initiate a search in the graph G_{cat}. If the phrase representing core teaching concept is present in the graph, it is assumed that there is some relation between the two. However, if the core teaching concept is not present in G_{cat}, this indicates that the image and the concept are hardly connected to each other.

5.2 ***Filter out unrelated teaching concepts:*** If a concept is present in G_{cat}, store it in $TC_{R.}$

$TC_R= \{c_r: c_r \epsilon \text{ TC } \& \ c_r \epsilon \ G_{cat} \}$

If a particular core teaching concept is present in G_{cat}, it is assigned weight as per the rules given in step 6. Else the system assigns the phrase from TC an infinite value. This will filter out the core teaching concepts that are completely unrelated to the categories representing the image idea.

5.3 ***Calculate distance:*** After filtering out unrelated concepts, the contextualizer now tries to see how close or far a particular concept in TC_R is from the image idea. For this, the system calculates distance between all pairs of phrases in TC_R and categories in G_{cat}. It calculates distance using the simple Dijkstra's shortest path algorithm (Dijkstra EW et al., 1959). The unit of distance here is the number of nodes between the two nodes under consideration. If the two nodes are directly connected, the system assigns the pair a value of 0.

Let D_{pair} store distance corresponding to all possible pairs of categories from G_{cat} and phrases from TC_R. The total number of pairs thus generated are (G_{cat}*TC_R). Now, the contextualizer moves on with pairs assigned distances. To reach even more specific and closely associated pairs of concepts and image idea, the system assigns weight. Weight assigning helps prioritize certain combinations of the two inputs. Prioritization in turn increases the chances of getting relevant results.

Step 6: Assign weights to category- concept pairs

Weighing is crucial to give directed and relevant results as the dynamic environment has endless possibilities. To prioritize ideas emerging from images that are highly related to notes according to the linkages built upon wikipedia categories, contextualizer follows a weighing mechanism. Following are the steps to assign weight to every pair present in D_{pair}.

6.1 Evaluate the closeness of concepts and categories representing image idea: The idea is to assign higher weight to concepts- categories pairs that are closely related. This is one way to dig out meaningful relationships between class notes and real life objects. To achieve this, the system weigh the pairs according to distance between the category and concept of every pair in D_{pair}. Let P_i represent a particular pair in D_{pair}. Initially, the system assigns weight according to the following rule:

$$W_I (P_i)= d / n \qquad \text{…….eqn. 2}$$

Here, W_I is initial weight,

d= number of nodes between a particular concept and category from D_{pair} and

n= total nodes in the graph G_{cat}.

6.2 Closeness of category with the image idea: If the phrase in P_i is present in base level category the system finds in step 4.2, initial weight is divided by 1. Else, the final weight is proportional to the distance of the category from the base level categories.

$$W_F (P_i)= W_I (P_i)/i; \qquad \text{…..eqn. 3}$$

where, W_F is the final weight and i is the level of each wikipedia category specified in step 4.3.1.

6.3 ***Sort the pairs:*** The system then sorts the pairs from D_{pair} on the basis of decreasing order of their weights.

Step 7: Initial mining of knowledge resources

The contextualizer we propose utilizes both TC, the set of index phrases filtered and G_{cat}, the categories that are present in the list of pairs in D_{pair} to scout for relevant websites. For experimental purposes, contextualizer uses only top 100 pairs from the sorted D_{pair} list. Searching the web uses only 100 pairs with higher weights.

7.1 Create a search query: The system formulates search queries using the terms from P_z and name of the subject, S. It initiates a search for every combination phrase, category and subject name for the top 100 entries in D_{pair}.

Search query = <"phrase from TC" + "category from G_{cat}" + "S">

7.2 Conduct preliminary search: Using an appropriate search engine, the system stores top 10 results corresponding to every search query from search engine result page (SERP) in the list IR (initial results). In this case, IR thus contains a total of 100*10 URLs of web pages.

It is important to note here that each website in IR originates from a distinct index phrase from via one of its relevant categories somehow related to the image.

Step 8: Filter most relevant results

The contextualizer then applies a secondary ranking algorithm to select the best and most relevant results from IR.

8.1Basic filtering: The system screens initial websites with the following filters to exclude irrelevant websites from IR:

- At least one of the phases present in pairs of D_{pair} must be present on the web page, else the system deletes it from IR.
- Contextualizer also removes community forums since they provide too generic information.
- All the Web pages present in IR are checked for a preset list of stop_words such as 'amazon', and 'youtube', and finally eliminated.

***8.2 Check semantic relevance*:** Contextualizer now evaluates semantic closeness of every website present in IR after initial filtering. The semantic closeness is assessed with respect to both- the technical phrase as well as categories available in D_{pair}. We employ pre-trained word embeddings based on the GloVe (Global Vectors for Word Representation) algorithm by Stanford [Pennington J, Socher R, and Manning C. 2014]. GloVe represents words with vectors. These vectors are mathematical representations of semantic information of the words. Following are the steps for finding semantic relevance:

8.2.1 Deal with special characters: For every website present in IR, the system replaces special characters with blank spaces. Since these special characters do not add meaning to the article in any way, the system doesn't lose any information.

8.2.2 *Remove stops words if any:* Contextualizer also removes all the stop words present in all web sites of IR.

8.2.3 *Tokenize the articles:* The content on every website is converted into a set of tokens. Each token represents pairs of words and the corresponding frequency of the word in the article.

$A=\{(w_1,f_1), \ldots (w_i,f_i), \ldots (w_n,f_n)\},$ …..eqn 4

where w_i occurs with a frequency f_i [Kanika et al, 2019].

Let us consider a multi-word phrase $X = \{l_1, l_2, l_3, \ldots , l_n\}$ which may be either a category arriving through image or a phrase chosen from the index terms in D_{pair}. The semantic similarity between article *A* and *X* is given as:

$$Similarity(A, X) = \frac{\sum_{(w_i, f_i) \in A} \sum_{l_j \in X} f_i \times Similarity_{ij}}{|A||X|}$$

……..eqn 5

$$Similarity_{ij} = \frac{\vec{w}_i \cdot \vec{l}_j}{|\vec{w}_i||\vec{l}_j|}$$

……..eqn 6

Where $\vec{w_i}$ and $\vec{l_j}$ are the vector embedding representation of w_i and l_j respectively and $|A|$, $|X|$ are the respective lengths of the article A and phrase X [Kowser, S., Rahman, S., Shojol, D. A., and Kabir, M. 2019].

8.2.4 Find out overall similarity score: The overall semantic similarity score for a given image category cat in G_{cat} and any index phrase in D_{pair}. L is a linear combination of a prefixed weighting factor α [Kanika et al. 2019]. Overall similarity is calculated using equation 7.

$$Similarity_{score} = \alpha \times Similarity(A, p) + (1 - \alpha) \times Similarity(A, m)$$

……eqn 7

The weighting factor α determines whether the recommended site is semantically closer to the image or to the class notes.

Step 9: Recommend the context of concepts

The websites that are most similar to both- the concepts to be taught in class represented by selective index phrases as well as categories adding context to the concepts are recommended to the teachers.

9.1 Sort the list of websites: Once we have the similarity scores of all the websites in filtered IR, the system sorts the list according to decreasing order of similarity score.

9.2 Recommend: The contextualizer recommends top n websites to the teachers as contextual examples.