The Workshops of the Tenth International AAAI Conference on Web and Social Media
Wiki: Technical Report WS-16-17

Semi-Supervised Automatic Generation of Wikipedia Articles for Named Entities

Yashaswi Pochampally
IIIT Hyderabad, India
p.yashaswi@research.iiit.ac.in

Kamalakar Karlapalem
IIIT Hyderabad / IIT Gandhinagar, India
kkamal@iitgn.ac.in

Navya Yarrabelly
IIIT Hyderabad , India
yarrabelly.navya@research.iiit.ac.in

Abstract

We investigate the automatic generation of Wikipedia articles as an alternative to its manual creation. We propose a framework for creating a Wikipedia article for a named entity which not only looks similar to other Wikipedia articles in its category but also aggregates the diverse aspects related to that named entity from the Web. In particular, a semi-supervised method is used for determining the headings and identifying the content for each heading in the Wikipedia article generated. Evaluations show that articles created by our system for categories like actors are more reliable and informative compared to those generated by previous approaches of Wikipedia article automation.

1 Introduction

Wikipedia has grown to be the world's largest and most accessed free encyclopedia, in which articles are collaboratively written and maintained manually. Tasks like Entity Linking help in understanding the named entity by linking it to the corresponding Wikipedia article. But there are many such named entities which do not have pre-existing Wikipedia pages. Given the huge size and growth rate of such entities, manual creation of these articles will eventually cease to be feasible. Thus, a need arises to automate the generation of Wikipedia articles for such named entities.

Automatically generating a Wikipedia article for a named entity (topic) like a movie or an actor raises several challenges. The three major ones that are addressed in this paper are:

1. maintaining similarity in its article structure (the section headings) and article content (the content under section headings) with that of other articles belonging to the same category
2. inclusion of section headings that are uncommon but, important to the topic in specific and
3. non - redundant retrieval and organization of information as coherent sentences.

For the selection of section-headings in a topic, a supervised method would be to consider the most common headings from other articles in that Wikipedia category. This would help in maintaining similarity of article structure across articles of a Wikipedia category, but at the cost of losing uncommon headings that might be important for the topic chosen. For instance, the common headings for film actors are *early life*, *career*, *filmography* and *awards*. But the Wikipedia article generated by this method for some *expired actor*, say Uday Kiran, while including the aforementioned headings still would not contain information about his death. An unsupervised approach which clusters text related to the topic into various headings would include all topic-specific headings and solve the issue, but at the cost of losing similarity in article structure (common headings) across articles belonging to a category. Our semi-supervised method solves these issues by considering the important headings specific to the topic under consideration along with the common headings from other articles of the Wikipedia category. The content related to the topic, which is irrelevant to common headings is further subjected to topic modelling techniques to get the topic specific headings.

For retrieving the content under common headings, a supervised approach would be to process content under all headings by identifying the text similar to training data under the respective headings. This process would not be able to fetch accurate information for headings like *filmography*, where the content is the list of movies for an actor which does not have much similar text with various other articles. Other examples of headings that face the same problem include *Discography* in singers category, *books* in Writers category, etc. An unsupervised approach which processes the content for all headings using text extracted from google results with the topic and corresponding heading as the query solves this problem, but cannot utilize the similarity of information across some similar sections in different articles. Our semi-supervised method first classifies the section headings into Type-1 (those whose content follows a repeating text pattern across articles) and Type-2 (those whose content is more in the form of lists and does not share repeating text patterns across articles) and then uses a supervised method and an unsupervised method respectively for their content generation.

The subsequent discussion of this paper is organized as follows. We discuss the previous approaches for Wikipedia Automation in Section 2. In Section 3, we describe our model where we use semi-supervised approach for both, se-

lection of section headings and identification of the information corresponding to these section headings. Section 4 presents the experiments and results. Finally, we conclude in Section 5.

2 Related work

For automatic generation of an article, a structure-aware approach has been proposed by (Sauper and Barzilay 2009). Their approach generates a list of headings as a common template for all Wikipedia articles related to a particular category. To create a new article for a topic, it retrieves from the internet a set of excerpts for each heading in the template. Using a perceptron framework augmented with an ILP formulation (selection model), the system is trained to select the best excerpt for each heading. This approach raises a few issues that limit its applicability for some Wikipedia categories.

First, their approach does not utilize the similarity of information under similar headings across different articles. Instead, it directly retrieves the content related to a heading from the internet. On the other hand, work presented in WikiKreator (Banerjee and Mitra 2015) utilizes such similarity, but assumes that content sections of all headings have such similarity. This assumption may not hold in categories like Indian Actors. Using Latent Dirichlet Allocation based topic distribution vectors as features, their classfier cannot accurately learn the text for headings like *filmography* whose content has the list of movies of an actor and does not share similar topics across various articles. We address this issue by classifying the common headings into Type-1 and Type-2 section headings and then using supervised and unsupervised approaches, respectively.

Second, while choosing the content for a particular heading in its article, structure aware approach only included the most informative excerpt among the various extracted from the web. This might lead to loss of information that should ideally be in a Wikipedia article, especially when information related to a heading is scattered through Internet sources. Our approach caters to such an issue by collecting information related to the topic from various reliable web sources and then aggregating information related to each of its headings to form its content.

Third, building high-quality training data for the selection model requires a lot of manual effort for choosing the candidate excerpts and the best excerpt for each heading. Our approach does not require such manual effort since the training dataset we use is the directly available Wikipedia articles of a particular category.

Fourth, by selecting a common template, all the articles belonging to a Wikipedia category would have exactly the same set of section headings, which is practically not possible when it comes to categories like film actors. For instance, let us assume that the common template for film actors has the list of headings as *early life*, *career*, *filmography* and *awards*. Now, the Wikipedia article generated by the structure aware approach for actor Chiranjeevi would not contain any information about his Humanitarian work/Charity. The approach proposed in Autopedia (Yao et al. 2011) solves this problem by choosing the best template from the templates of various categories the topic belongs to. However, this approach cannot be followed if the various categories of the new topic are not known a priori. We deal with this problem by extracting the content related to the topic extracted from the internet, which is not considered in common headings. From such content, we extract the topic specific headings through topic modelling techniques.

Processing the content sections of Type-1 and Type-2 headings differently followed by using topic modelling for the inclusion of topic specific headings sets our approach apart from previous work on automation of Wikipedia articles.

3 Wikipedia Article Generation

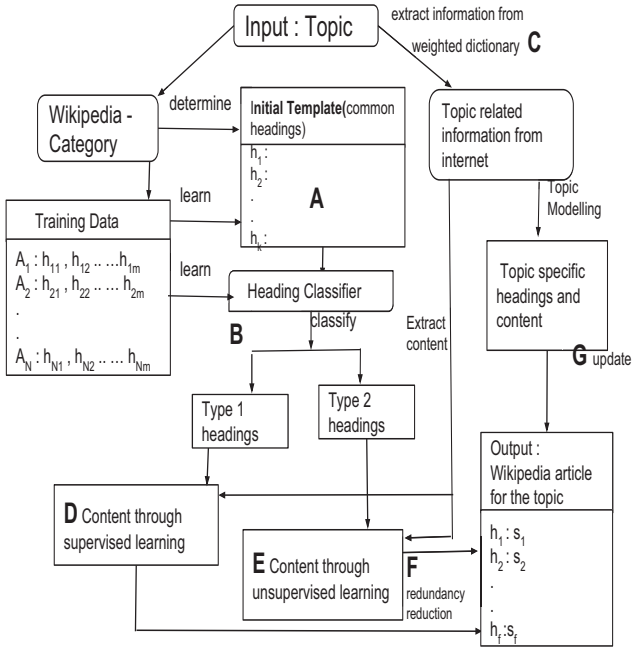

Figure 1: Framework for automatic generation of Wikipedia article with Steps(A,B,C,D,E,F,G)

The objective of our model is to produce a Wikipedia article for a given topic related to a particular Wikipedia category. We assume that

1. There are many human-edited articles related to this category
2. There is relevant information available on the internet, but scattered among several web pages with majority information included in a fixed set of reliable webdomains.

Wikipedia provides an API[1] to download articles in the XML format. Given a category, the API is capable of extracting all the articles under it. For a particular Wikipedia category, we use the API to create the training dataset with the contents in the sections across articles as instances and

[1] https://en.wikipedia.org/wiki/Special:Export

the section headings as the corresponding classes. Using this training corpus, we build an initial template with a list of common headings. These headings are then classified into Type-1 or Type-2 using a heading classifier. The content under these headings is also extracted. We get the topic specific headings and their content sections by using topic modelling techniques over the text extracted from the Internet that is irrelevant for common headings. We combine and process Type-1, Type-2 and topic specific contents together to get the final list of section headings and their corresponding content sections.

Our framework for automatic generation of Wikipedia article for any named entity is illustrated in Figure 1. We follow the Steps A, B, C, D, E, F in that order to generate the Wikipedia article for a new entity. The offline component (Section 3.1) processes the training corpus and the online component (Section 3.2) processes the relevant information for the topic that is available on the Internet.

3.1 Offline Component

Step A - Initial Template creation: We present a novel approach to find the template (ordered list of frequent headings for the new article) for a wikipedia category by representing headings as transactions of a transactional database. For a transactional database, frequent itemsets identify the items that often occur together and frequent closed sequential patterns identify the same with an additional consideration of their ordering in the transactions. In our case, frequent itemsets give the frequently occurring headings and the closed sequential patterns give the ordering of these headings. Our graph based approach for finding the template is given in Algorithm 1.

Firstly, the ordered list of headings in the articles of the training set are considered as transactions. The frequent itemsets and frequent closed sequential patterns are then extracted from these transactions using FP-growth (Han, Pei, and Yin 2000) and BIDE (Wang and Han 2004) algorithms respectively. Only the itemsets having support greater than a threshold value are considered. We then construct a directed weighted acyclic graph such that the vertices represent the headings selected in frequent item sets, directed edges represent the sequential patterns discovered and the weight of each edge represents corresponding aggregated support value from sequential patterns discovered. At each iteration of adding an edge, we check if the graph has a cycle and and delete the low weighted edge of the cycle to make the graph acyclic. Finally, the topologically sorted output of the constructed weighted acyclic graph gives the final ordered list of headings in the template.

For instance, we consider Wikipedia category: Indian Actors, an example heading transaction would be: $early\ life$, $awards$, $career$, $filmography$.

Algorithm 1 Ordering of common headings

1: $htransactions$ = []
2: **for** $Article$ in $WikipediaArticles$ **do**
3: $htransactions$.add($orderedHeadingList$)
4: **end for**
5: FIS = items of FP-growth($htransactions$) with relative support >threshold and $itemsetsize > threshold$
6: FCS = BIDE($htransactions$) with relative support >threshold [arranged in descending order of their supports]
7: $Graph$ = $empty$
8: **for** $pattern(h_1 \rightarrow h_2 .... \rightarrow h_n : sup : s)$ in FCS **do**
9: **for** (h_i, h_j) in $h_1 \rightarrow h_2$, $h_2 \rightarrow h_3 .... h_{n-1} \rightarrow h_n$ such that $h_i, h_j \in FIS$ **do**
10: **if** ($Graph$ has no edge $h_i \rightarrow h_j$) **then**
11: $Graph$.addEdge(h_i, h_j)
12: $Graph$.weight(h_i, h_j)=s
13: **else**
14: $Graph$.weight(h_i,h_j)=$Graph$.weight(h_i,h_j)+s
15: **end if**
16: **if** cycle C in $Graph$ **then**
17: delete edge (h_i, h_j) where $Graph$.weight(h_i, h_j) = min $Graph$.weight(h_a, h_b) for all (h_a, h_b) in cycle
18: **end if**
19: **end for**
20: **end for**
21: $FinalOrdering$ = topological sort($Graph$)
22: **return** $FinalOrdering$

FIS	Support
A, E, C, F	396
A, P, C, F	402
E, P, C, F	353 and so on

Table 1: Sample Frequent Itemsets

FCS	Support
$E \rightarrow F$	494
$E \rightarrow C \rightarrow A$	489
$E \rightarrow C \rightarrow F$	433 and so on

Table 2: Sample Frequent Closed Sequential Patterns

The frequent item sets (FIS) and frequent closed sequential patterns (FCS) discovered from 1700 such transactions are shown in Table 1 and Table 2. The headings $television$, $early\ life$, $filmography$, $awards$, $career$ and $personal\ life$ are denoted by T, E, F, A, C, P respectively. From FIS, the items chosen for the list of common headings are A, C, P, E, T, F which form the vertices of the graph. We add edges $E \rightarrow F$ ($weight$: 494), $E \rightarrow C$ ($weight$: 489), $C \rightarrow A$ ($weight$: 489) and so on (from FCS) to the graph. The graph formed is shown in Figure 2. The topological sorted output for the graph ($E \rightarrow C \rightarrow P \rightarrow A \rightarrow F \rightarrow T$) is the final ordering of the common headings.

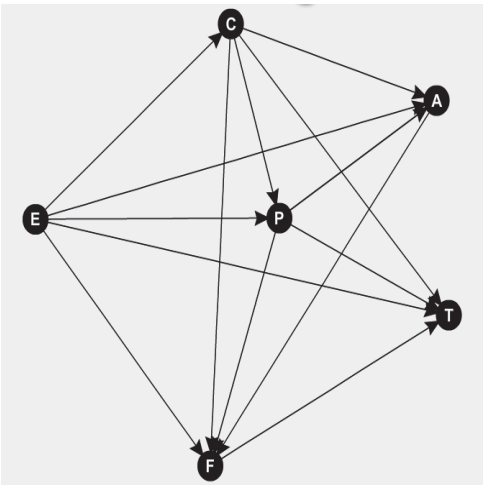

Figure 2: Graph for headings in Indian Actors Category

Step B - Classification of headings into Type-1 and Type-2: Our model classifies headings of the initial template, h_1, h_2...h_k into two classes namely, Type-1 and Type-2. Type-1 headings have more text patterns that are being followed in all wiki pages, whereas Type-2 headings do not have many such. For instance, heading *career* has patterns like *graduated from* being followed in its content and so it can be considered a Type-1 heading, whereas the heading *filmography* has the list of movies for that actor as its content (with not much content similarity across articles) and so is considered a Type-2 heading. However, patterns like the mention of the year which is included along with the movie name can be learnt even for filmography, but the presence of year alone cannot make any sentence a part of the actor's filmography. So, direct text learning may not work for such headings. It has to be noted that the more text similarity across contents in articles for the heading, the more is the probability that its content pattern can be learnt and the more is the probability that it is a Type-1 heading.

Repeating text patterns are checked for, in the content, presence or absence of which, would make the heading Type-1 or Type-2, respectively. We use a multinomial bayesian text classifier[2] which estimates the conditional probability of a particular word/term/token given a class as the relative frequency of the term in the text belonging to that class for understanding such repeating text patterns. The classifier uses the content sections for headings in the initial template as the training set, where sentences form the instances and their section headings form the corresponding classes. We extract n-gram based text patterns to learn the repeating text patterns in the content and use mutual information feature selection method, to measure how much information the presence or absence of a particular term contributes to making the correct classification.

[2]https://github.com/datumbox/NaiveBayesClassifier

Heading	Recall	Precision	F-score	Type
Introduction	0.7227	0.9997	0.8389	1
filmography	0.0661	0.9999	0.1240	2
career	0.8721	0.9999	0.9316	1
early life	0.7244	0.9991	0.8398	1
personal life	0.6860	0.9997	0.8137	1
awards	0.0052	0.9966	0.0105	2
television	0.0003	0.9933	0.0006	2

Table 3: Cross Validation measures for common headings in Category: Indian Actors

K-fold cross validation results of this classifier for sentences under each heading are calculated separately. Headings that have high F score measures (considers both precision and recall) indicate that the classifier could recognize the repeating text patterns well and classify the sentences of that heading correctly. Hence, a heading is considered Type-1 if it has F score greater than a threshold and is considered Type-2 otherwise. Table 3 shows the results of common headings for Wikipedia Category: Indian Actors. In this category, precision values of all the headings is near to 1 which makes recall the distinguishing factor responsible for high (Type-1) or low (Type-2) F score measures.

Step C - Weighted Dictionary of websites: Wikipedia editors generally cite the websites from which they have extracted the information in the external links section of the article. We leverage this fact to choose the best web domains that have to be crawled for retrieval of information for the articles.

From training data (various human edited articles of a category), we find the most frequent domains which have been cited in the external link section of the articles. Of these web domains, we exclude domains which do not stand by the guidelines of the Wikipedia[3]. In particular, we remove websites that are social networking websites, official blogs or public forums. We also exclude non-textual websites like *youtube.com* or *instagram.com* from which text cannot be extracted. The resulting list of domains with their corresponding frequencies as weights forms the weighted dictionary of websites for that category. For example, the weighted dictionary for Actors is {'imdb.com':1598, 'bollywoodhungama.com':178, 'oneindia.in':76}. By extracting information from these selected websites rather than entire web, we are not only abiding by the Wikipedia guidelines for choosing content only from reliable websites, but also reducing the probability of junk sentences creeping into the article.

3.1.1 Learning the content of Type-1 headings: The content under Type-1 headings is dealt with as a text classification problem. The training dataset here is the content-sections of Type-1 headings from the training corpus labeled with their corresponding headings. For each heading h_i, let $s_{i1}, s_{i2}...s_{ir}$ be the content sections from various articles of the training corpus. Each section s_{ij} has various sentences

[3]http://en.wikipedia.org/wiki/Wikipedia:What_Wikipedia_is_not

$sen_{ij1}, sen_{ij2}... \ sen_{ijl}$. Training data for our text classifier would be all such sentences sen_{ijk} labelled with class label h_i.

(Dai, Olah, and Le 2015) recommended paragraph vector representation for grouping Wikipedia articles together, which we are using. We have to use a text classifier like K-Nearest neighbour or Nearest Centroid that classifies the paragraph vectors which are relatively nearby each other (when plotted over space) to the same class label. Table 4 shows the evaluated results for various benchmark classifiers. Despite the high training time complexity, we choose KNeighborsClassifier as it gave better cross-validation accuracies for the training data. The KNeighborsClassifier text classifier, which is made to learn the training instances will later be used for classifying new sentences in Step D.

Classifier	Accuracy (+/- error)
Ridge	0.63 (+/- 0.01)
Perceptron	0.55 (+/- 0.23)
PassiveAggressive	0.63 (+/- 0.22)
KNeighbors	**0.71** (+/- 0.02)
RandomForest	0.67 (+/- 0.03)
NearestCentroid	0.60 (+/- 0.05)
SGDClassifier	0.64 (+/- 0.01)

Table 4: Accuracy measures for bench-mark classifiers

3.1.2 Learning the website base for Type-2 headings: For each of the Type-2 headings, we create a heading specific weighted dictionary by reordering the earlier list of weighted dictionary (from Step C). For a particular heading, the weight corresponding to a website is the average of the text similarity scores of the heading's content section from Wikipedia with that crawled from the website for various topics in the training corpus. By dividing the content of a webpage into logical sections, we also learn the section of the webpage that had maximum similarity with the content in Wikipedia.

3.2 Online Component

Step D - Extracting the information under Type-1 headings: By crawling the weighted dictionary of websites (learnt from Step C), we get the text extracts related to the topic under consideration. These extracts are split into *independent text units* using Stanford coref resolution tool (Manning et al. 2014) and Anaphora Resolution (Qiu, Kan, and Chua 2004) with a condition that all mentions of any sentence in a text unit are contained in the same text unit. Mention of the topic under consideration stands as an exception to this condition because nearly all sentences related to the topic have topic as one of their mentions. For example, the text extract "Kim Sharma is an actor. Her first movie is Mohabbatein. It was a huge hit" would be split into two units "Kim Sharma is an actor" and "Her first movie is Mohabbatein. It was a huge hit".

Each of these independent text units is classified into Type-1 headings using the text classifier built in Section 3.1.1. Only text units which got classified into h_i with a confidence greater than a threshold value are considered so that, we rule out the possibility of a similar kind of sentences getting classified into different headings. By doing so, redundancy across content sections of the headings is minimized. Let the resulting sentences for each h_i be $sen_{i1}, sen_{i2}....sen_{ir}$ where each sentence holds a score that is the weight of the website (from the dictionary of websites in Step C) from which it was extracted.

Step E - Extracting the information under Type-2 headings: For each of the Type-2 headings, we get the text extracts from each of the weighted dictionary of websites from their corresponding sections learnt in Sec 3.1.2. Each text extract holds a score, which is the weight of the website (heading specific dictionary of websites of Section 3.1.2) from which it was extracted. If there is no considerable text in these sections of the heading specific websites, then the content is crawled from other known reliable web domains. We do the same even in the case of Type-1 headings when the number of extracts that are retrieved from the weighted dictionary of websites is very less.

In each of the domains, the content is divided into some logical sections and the text is extracted from the section which has a section-heading similar to the heading we are considering. Some human intervention is needed to understand the organization of webpages in each new web domain we are dealing with. This step would require learning the html tags used for headings and their content for the webdomains included in the weighted dictionary of websites and other known reliable webdomains if any. However this human work is limited one time activity for a particular wikipedia category.

Step F - Redundancy Reduction and Sentence ordering: Redundancy Reduction has to be taken care of within content sections of Type-1 and Type-2 headings. We have sentences with the corresponding scores under each of these headings. We measure the cosine similarity score of each sentence with all other sentences belonging to the same heading and delete the sentences which are redundant, ensuring that sentences that have more words and possess higher scores (weights of the websites they are extracted from) have a greater probability of getting retained.

To improve the readability, the text units under each of Type-1 headings are ordered based on the content-sections of that heading in the training data. Sentences are first segregated based on the websites from which they are extracted. All sentences belonging to a particular website retain the order in which they are extracted to preserve the meaning and context. To compute the ordering among websites, we employ the sentence ordering technique in (Balasubramanian and Cucerzan 2009) where precedence score of each sentence in a website is computed against all sentences from other websites by aggregating the pairwise scores determined using word-based statistics of the training data.

For each website, we compute the average of overall precedence scores of all its sentences. Finally, websites (with their sentences in the retained order) are placed in the descending order of their average precedence scores. Minor

changes in the ordering of sentences done in Section 3.3 will result in completely readable content.

Step G: Updating the template We process the sentences that were irrelevant to above-processed headings. The sentences that are filtered out in Step D due to low confidence level and are not similar to text extracted for Type-2 headings are added to the section heading *Extra Information*. The content under this heading is subjected to topic modelling techniques to find the topic specific headings.

We use the Topic Modelling tool provided by Mallet (McCallum 2002) to discover topics from the content and infer the content spread of each sentence over those topics. Topics having a probability greater than some threshold are included as topic specific headings. For each such heading, the content is generated by aggregating the sentences that have content spread with a probability greater than 0.7 for that topic.

3.3 Post-Processing

After identifying the content sections for all the headings, the headings whose content sections are empty are deleted. From the resulting headings, common headings and their contents are then placed in the order got from Step A and topic-specific headings are placed at the end. Affiliated *TAGME RESTful API* (Ferragina and Scaiella 2010) is used to tag the content and link it to corresponding Wikipedia pages thereby making the text look similar to that of Wikipedia articles. Finally, the article is human edited to make any minor changes and generate the final Wikipedia page.

4 Experiments and Results

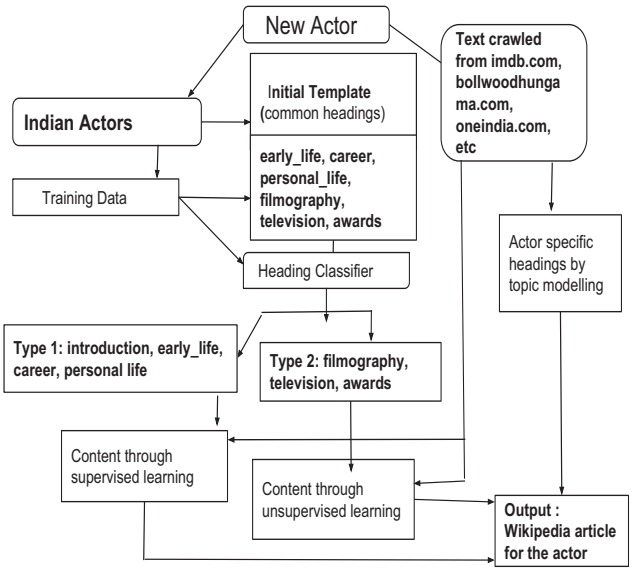

Figure 3: Workflow for Indian Actors

Figure 3 shows the overall workflow for any article in the category: Indian Actors. Online data is collected by weighted dictionary of websites which include imdb.com, bollywoodhungama.com and oneindia.com. We included text from other sources such as wikiprofile.in, filmyfolks.com, movies.dosthana.com, etc when the content extracted from the aforementioned websites is very less. We generated new Wikipedia articles for 20 actors. Figure 4 shows one such article generated by our system for the actor: Kim Sharma whose Wikipedia article[4] does not contain much information. Articles generated by our system not only helped in creating new Wikipedia pages but also helped in enhancing the existing Wikipedia pages.

Kim Sharma (full name Kim Michelle Sharma) is a Bollywood actress and model.She is also the brand ambassador of Olay in India. [Source - filmibeat.com]
She is an actress, known for Mohabbatein (2000), Fida (2004) and Yakeen (2005). [Source :imdb.com]
Kim Sharma has also done a couple of films in Tamil and Telugu language. [Source :bollwoodlife]
=====early_life=====
Kim Sharma was born on January 21 1980 in Ahmednagar, Maharashtra, India as Kim Michelle Sharma. [Source :imdb.com]
Kim was born in Ahmednagar and did her schooling from a boarding school in Panchgani. [Source :filmibeat.com]
=====career=====
She started her career with appearing in TV commercials when she was spotted by Aditya Chopra for his multi-starrer movie Mohabbatein. her role in this movie as female lead in one of the young couples was lovable and she got more Bollywood offers.A few of her films that followed are Tumse Achcha Kaun Hai, Fida, Taj Mahal: An Eternal Love Story, Heyy Babyy, Money Hai Toh Honey Hai, and Loot.But none of these films did much for her career. [Source :bollywoodlife.com]
After that she returned to her hometown and joined a junior college but hated her college. 'Having studied in a boarding school, where most of the girls were from aboard, living and studying in Ahmednagar came as a culture shock', quips Kim.So while in her twelfth, she decided to call it quits.On a trip to Mumbai, she went through an audition for Close-Up toothpaste and as destiny would have it, was selected too.After the Close-Up commercial, many others followed like Sunsilk, Pepsi, Tata Safari, Ponds, Fair & Lovely, Clean-n-Clear and Liril,Sharma caught the eye of producer Aditya Chopra, and he signed her on his next directorial venture titled Mohabbatein.The film also marked the launch of five other newcomers, Uday Chopra, Jugal Hansraj, Jimmy Shergill, Shamita Shetty and Preeti Jhangiani.Mohabbatein also starred superstars Amitabh Bachchan, Shah Rukh Khan and Aishwarya Rai Bachchan.It was a huge hit and started Kim's career in Bollywood.In fact Mohabbatein is her only hit except for Heyy Babyy and Magadheera which she made Item number in.[Source :filmibeat.com]
=====personal_life=====
Kim Sharma married Kenyan businessman Ali Punjani 2001 and since shifted to Africa. [Source :bollywoodlife.com]
=====filmography=====
***As Actress 2011 Loot Sharmili M. Sequioa / SMS, 2010 Yagam Sophie, 2009 Marega** Salaa Tanya Singhania, 2009 Daddy Cool: Join the Fun Jenny, 2009 Anjaneyulu Dancer / Singer, 2009 Magadheera Special appearance (song "Jor sey"), 2009 The Whisperers Natasha, 2008 Money Hai Toh Honey Hai Sara, 2007 Godfather: The Legend Continues, 2007 Chhodon Naa Yaar Rashmi, 2007 Heyy Babyy Special appearance in song, 2007 Nehlle Pe Dehlla Bipasha's Friend, 2006 Tom, Dick, and Harry Bijli, 2006 Zindaggi Rocks Joy, 2006 Ladies Tailor Radhika, 2006 Kudiyon Ka Hai Zamaana Kanika, 2005 Taj Mahal: An Eternal Love Story Laadli Begum, 2005 Yakeen Tanya S. Thakur, 2005 Padmashree Laloo Prasad Yadav Rita, 2004 Fida Sonia, 2003 Alai, 2002 Khadgam, 2002 Kehtaa Hai Dil Baar Baar Ritu Patel, 2002 Tum Se Achcha Kaun Hai Bobby Gujral, 2000 Mohabbatein Sanjana *As Self 2008 Desh Drohi Herself - Item Girl [Source :imdb.com]
=====awards=====
****In Awards of the International Indian Film Academy 2001 Won Popular Award** Best Female Debut Mohabbatein (2000) [Source :imdb.com]

Figure 4: Generated Article for Wiki Stub: Kim Sharma

[4]https://en.wikipedia.org/wiki/Kim_Sharma

4.1 Evaluation of human edits

The quality of the generated articles would depend on the number of changes that are made during human editing that is required to generate the final readable wikipedia article. Observations showed that these articles were readable and had good coverage of information. Only a few minor changes in the ordering of sentences or formatting were necessary. We had chosen a collection of 25 sample system generated articles which were manually edited to make them completely comprehensive. The percentage of sentences that were edited during this stage is computed. Details for 5 of them are tabulated in Table 5. The overall percentage of edited sentences for 25 documents was 12.26% which is tolerable.

Article	No of Sentences	No of edited sentences	% of edited sentences
1	33	2	6.0606
2	36	5	13.8888
3	73	4	5.4794
4	43	7	16.2790
5	85	12	14.1176

Table 5: Error Rate during human edits

4.2 Automatic Evaluation (Content Coverage)

ROGUE (Lin 2004) measures the overlapping units such as n-gram, word sequences and word pairs between the system-generated articles and their original Wikipedia articles (gold standard). We use the publicly available ROUGE toolkit to compute recall and F-score for ROUGE-1 and ROGUE-2. We considered 130 named entities which had existing Wikipedia pages and calculated the content coverage of the articles generated for these entities by fully supervised, fully unsupervised and semi-supervised (our method) approaches. In fully supervised approach, all headings are considered as Type-1 and content is generated through Text classification approach. In fully unsupervised approach, all headings are considered as Type-2 and the content under all headings is crawled directly from the internet. Results of content coverage are tabulated in Table 6. Experimental results show that our semi-supervised method outperforms the supervised and the unsupervised methods. Further processing of sentences and collection of information from some more reliable websites can increase the coverage even further.

	ROUGE-1		ROUGE-2	
System	Recall	F-Score	Recall	F-Score
Semi-supervised	**0.5658**	**0.4241**	**0.3076**	**0.2361**
Supervised	0.5190	0.3862	0.2756	0.2074
Un-supervised	0.1914	0.2189	0.0820	0.0965

Table 6: Content Coverage (ROUGE-1 and ROUGE-2 Recall values) across various approaches

5 Conclusion

In this paper, we implemented a system for automatic generation of Wikipedia article for a named entity, like a movie or an actor, where headings include both common headings in the category and additional important headings based on topic modelling. The content generation involves
i) supervised paragraph vector based text learning for headings that have repetitive content across various articles in the category and
ii) unsupervised learning for headings whose content is independent of other articles.
Our method uses the training data that is extracted directly from the wiki-dump, thereby reducing the manual effort. Extraction of text only from selected web domains of the external-links sections helped in discarding the junk data and made the resultant text reliable. Experiments showed the viability of our solution as an alternative to the previous approaches that used supervised or unsupervised methods. However, our method is more applicable for categories which have Type-2 headings in their common headings.

Future work will involve processing the extracted sentences into more readable paragraphs and examining the utility of including sub-headings within a heading using text clustering techniques. We will experiment pattern learning techniques (other than direct text learning used for Type 1 headings) for Type 2 headings rather than direct extraction from web. We will also investigate the merging of duplicate Wikipedia articles on a topic and splitting the content of a large Wikipedia article into various sub-articles.

References

Balasubramanian, N., and Cucerzan, S. 2009. Automatic generation of topic pages using query-based aspect models. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, 2049–2052. New York, NY, USA: ACM.

Banerjee, S., and Mitra, P. 2015. Filling the gaps: Improving wikipedia stubs. In *Proceedings of the 2015 ACM Symposium on Document Engineering, DocEng 2015, Lausanne, Switzerland, September 8-11, 2015*, 117–120.

Dai, A. M.; Olah, C.; and Le, Q. V. 2015. Document embedding with paragraph vectors. *CoRR* abs/1507.07998.

Ferragina, P., and Scaiella, U. 2010. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, 1625–1628. New York, NY, USA: ACM.

Han, J.; Pei, J.; and Yin, Y. 2000. Mining frequent patterns without candidate generation. *SIGMOD Rec.* 29(2):1–12.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In Marie-Francine Moens, S. S., ed., *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP nat-

ural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*.

McCallum, A. K. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Qiu, L.; Kan, M.; and Chua, T. 2004. A public reference implementation of the RAP anaphora resolution algorithm. *CoRR* cs.CL/0406031.

Sauper, C., and Barzilay, R. 2009. Automatically generating wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, 208–216. Stroudsburg, PA, USA: Association for Computational Linguistics.

Wang, J., and Han, J. 2004. Bide: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, 79–. Washington, DC, USA: IEEE Computer Society.

Yao, C.; Feng, S.; Jia, X.; Zhou, F.; Shou, S.; and Liu, H. 2011. Autopedia: Automatic domain-independent wikipedia article generation.