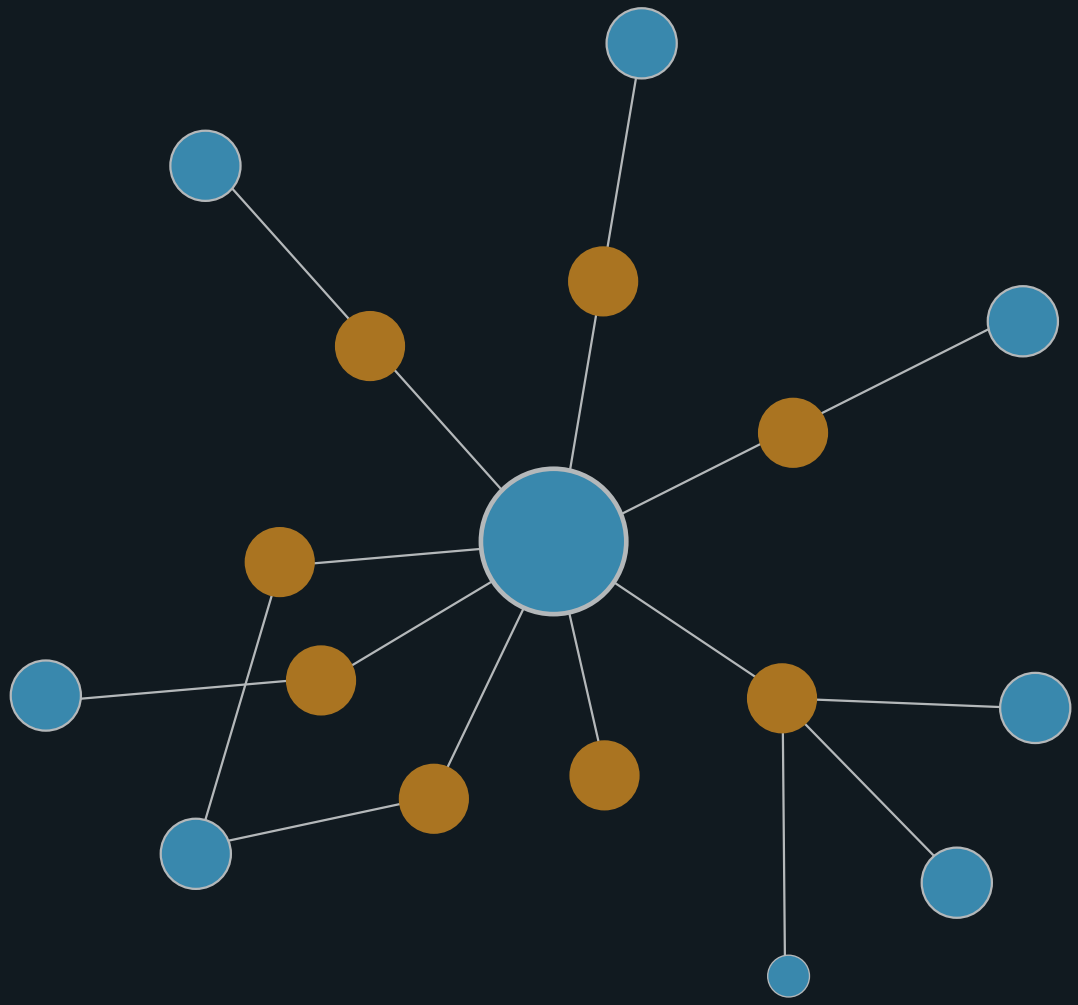
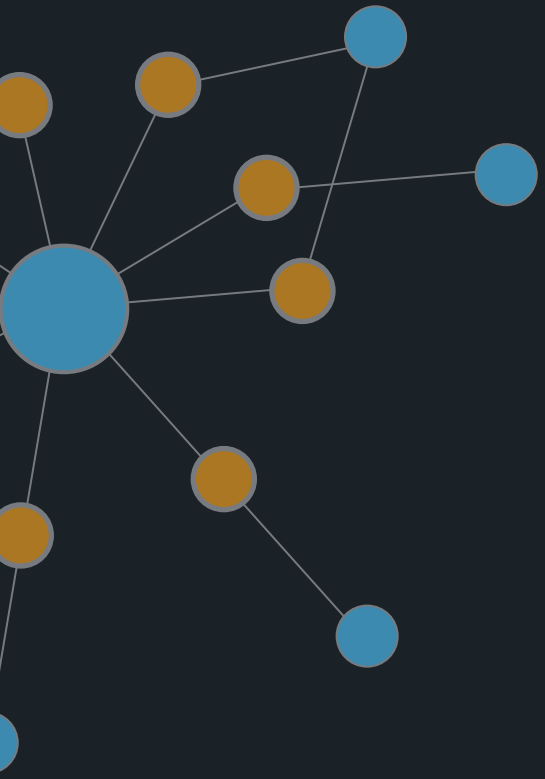
Mastering the Art of <Content Tagging>



piStats

Create • Deliver • Analyze • Personalize

www.pi-stats.com



INDEX Page No.

Introduction

Imagine you are a newspaper editor, authoring an article on the life history of 'Shah Rukh Khan'. Such an article would require extensive research on Shah Rukh Khan's entire life's story, and many previously published newspaper articles would be an excellent resource for that. However, there would be zillions of published newspaper articles and related news articles, associated with Shah Rukh Khan! A nightmare it can be, like looking for a needle in a haystack.

Even if somehow you can do that, there would still be thousands of newspaper articles, many of them having identical topics and reading all of them is practically infeasible.

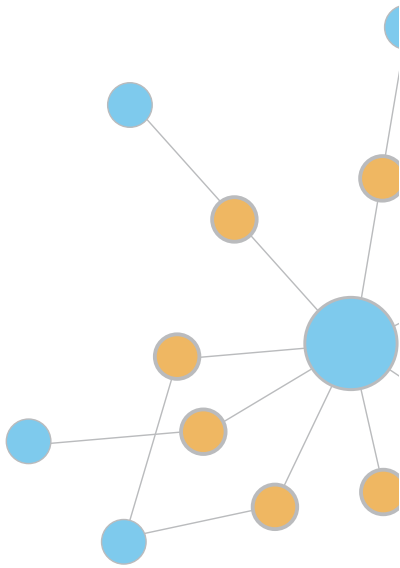
Do you feel stuck? How will you go through all the relevant newspaper articles in limited time?

Problems like these can be solved by 'tagging' all the newspaper articles with the best representative keywords. Now, you can easily filter the articles with tag "Shah Rukh Khan". You can further group the articles with similar tags to remove the identical articles from the corpus. You can also filter the articles with specific tags for different article segments, further reducing the total newspaper corpus. Thus, a simple act of article tagging could solve this major problem of organizing and summarizing information. This is one use case, think of the thousands other? Say hello to the art of 'Tagging and Tag Recommendation'.

There can be numerous applications of tags in organizing and sharing information and in increasing the capabilities of existing search engines. We have all used 'tags' while using social media applications like Facebook, Twitter, Instagram to associate our post with an important keyword. In most of the search applications, tags are being used in the background. No matter what the field is, these tags have always been the primary means of summarizing and organizing data.

Manual tagging on a large scale is a cumbersome process, requiring too much skilled manpower. Also, because of the number of people involved, the human error increases multifold; ultimately leading to an ineffective tag system.

Therefore, automatic, unsupervised **tag recommendation** is a necessity and we'll walk you through the process of automatically recommending tags for news articles.



Problem Approach

Let us suppose you have the following news article for which you must find 5 most important tags.

Mumbai:

The first poster of Shah Rukh Khan - Alia Bhatt starrer "Dear Zindagi" is out.

The official Twitter page of Red Chillies Entertainment, the production house headed by Shah Rukh and Gauri Khan, shared the first look.

"Life is a ride and you gotta keep moving! Presenting, #DearZindagiFirst-Look," they captioned the poster which features Alia and SRK riding bicycle.

The film is the second directorial venture of Gauri Shinde, who made a successful debut with "English Vinglish".

Shinde has also penned the story and is even co-producing the project. The trailer of the film will be out tomorrow.

Per reports, SRK will help Alia navigate through her life and relationships. "Dear Zindagi" scheduled to release on November 25, also stars Aditya Roy Kapur, Ali Zafar, Kunal Kapoor and Angad Bedi. Karan Johar is also on board as one the pro-ducers.

Step-by-step Approach

The chosen tags would consist (mostly) of **"Shah Rukh Khan", "Alia Bhatt", "Dear Zindagi", "first poster", "Red Chillies Entertainment".** We could do it very easily and if our system applies the same steps as we did, it should be able to get similar tags.

So, let us now go back to the start when we had only the news article and then advance step by step to state the rules for recommending tags.

At first, we removed the hyperlinks, html tags and other redundant parts from the article to do our processing on the modified content. This process of cleaning text is called Text Wrangling.

After that, we went through every word in the text to identify tags. This process of breaking text into words is called Word Tokenization.

We then removed words like "is", "the", "of" which occur frequently in most of the text and are redundant to select keyword candidates. This process involved part of speech tagging of every word. The entire process is candidate selection.

After selecting candidates, we look for certain features like keyword frequency, part of speech and then give different weight to different features, to get a score for every candidate. This is known as Feature and Score Calculation.

Finally, we select top tags having a score greater than some threshold.
Let's go through each step now in detail.

Text Wrangling

The wrangling steps taken to remove redundant content from all news articles are:

- Replaced the content in **'<>'** with space to remove html parts like ****
- Replaced the content starting with **'&'** and ending with **';'** with space to remove parts like **' '**
- Replaced website links starting with **'http, https, www'** with space
- Removed unicode characters from English text
- Removed punctuations like **'[](){}"'**
- Replaced punctuations like **'-!?_'@$'** with space
- Removed extra spaces in the beginning and end of the content

Word Tokenization

After data cleaning, our next target is to split the content at word level. These words will be then used to make unigram (single word) and multigram (multiple word phrase) candidates. Unigrams can be made by selecting every word in the content. One way to make multigram is to take all the combination of consecutive words as multigram candidates. For example, in the following sentence:

I **watched the movie. It was good.**

If we take all the possible trigrams, they would be :-
[I watched the], [watched the movie], [the movie.], [movie. It], [. it was], [it was good], [was good .]

Here **[the movie.], [movie. It], [.it was]** span over two sentences and can never be used for trigrams. So, we will take trigram candidates in a single sentence only.

Thus, we are going to break down word tokenization in following two steps:

Sentence tokenization:

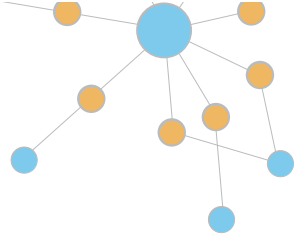
The content is broken down into a list of sentences, wherever special symbols like **'. -&;'** occur followed by white spaces. It breaks the entire content into a sentence list, wherever punctuation occur with the exemption of terms like e-commerce, Ram's son.

Special symbols are removed from the end of a sentence in a sentence list. So, in the above example, **[It was good.]** changes to **[It was good]**

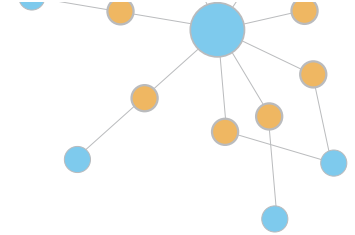
The result is a list of sentences, as follow:

['Mumbai',
'The first poster of Shah Rukh Khan Alia Bhatt starrer Dear Zindagi is out',
'The official Twitter page of Red Chillies Entertainment',
'the production house headed by Shah Rukh and Gauri Khan',
'shared the first look',
'Life is a ride and you gotta keep moving',
'Presenting',
'#DearZindagiFirstLook',
'they captioned the poster which features Alia and SRK riding bicycle',
'The film is second directorial venture of Gauri Shinde',
'who made a successful debut with English Vinglish',
'Shinde has also penned the story and is even co-producing the project',
'The trailer of the film will be out tomorrow',
'According to reports',
'SRK will help Alia navigate her life and relationships',
'Dear Zindagi scheduled to release on November 25',
'also stars Aditya Roy Kapur',
'Ali Zafar',
'Kunal Kapoor and Angad Bedi',
'Karan Johar is also on board as one the producers']

Word Tokenization: We split every sentence into a list of words. The result is a list of words.



```
[
['Mumbai'],
['The', 'first', 'poster', 'of', 'Shah', 'Rukh', 'Khan', 'Alia', 'Bhatt', 'starrer', 'Dear', 'Zindagi', 'is', 'out'],
['The', 'official', 'Twitter', 'page', 'of', 'Red', 'Chillies', 'Entertainment'], ['the', 'production', 'house',
'headed', 'by', 'Shah', 'Rukh', 'and', 'Gauri', 'Khan'],
['shared', 'the', 'first', 'look'],
['Life', 'is', 'a', 'ride', 'and', 'you', 'gotta', 'keep', 'moving'], ['Presenting'],
['#DearZindagiFirstLook'],
['they', 'captioned', 'the', 'poster', 'which', 'features', 'Alia', 'and', 'SRK', 'riding', 'bicycle'],
['The', 'film', 'is', 'second', 'directorial', 'venture', 'of', 'Gauri', 'Shinde'], ['who', 'made', 'a', 'successful',
'debut', 'with', 'English', 'Vinglish'], ['Shinde', 'has', 'also', 'penned', 'the', 'story', 'and', 'is', 'even',
'co-producing', 'the', 'project'],
['The', 'trailer', 'of', 'the', 'film', 'will', 'be', 'out', 'tomorrow'], ['According', 'to', 'reports'],
['SRK', 'will', 'help', 'Alia', 'navigate', 'her', 'life', 'and', 'relationships'], ['Dear', 'Zindagi', 'scheduled',
'to', 'release', 'on', 'November', '25'], ['also', 'stars', 'Aditya', 'Roy', 'Kapur'],
['Ali', 'Zafar'],
['Kunal', 'Kapoor', 'and', 'Angad', 'Bedi'],
['Karan', 'Johar', 'is', 'also', 'on', 'board', 'as', 'one', 'the', 'producers']
]
```



Separate lists of unigram (having one word), bigram (having two words) and trigram (having three words) is selected.

From these lists, three lists of keyword phrase candidates are made based on following conditions:

1) None of the words in the phrase should be a stopword. Stopwords are words which occur frequently in most of the articles and cannot be selected as tags like "a","is","of", etc. We used the "Stopword List" from NLTK to check every word.

2) Part of speech of all the words in the phrase should be noun, proper noun, adjective or verb.

In the case of trigrams, these conditions are only applied for first and last word, as the second letter in a tag could be a stopword and preposition e.g. Line of Control

Unigram candidates:

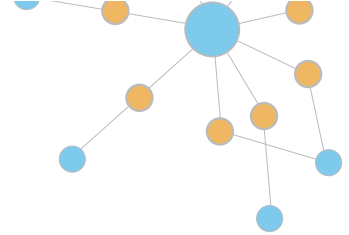
['Mumbai', 'first', 'poster', 'Shah', 'Rukh', 'Khan', 'Alia', 'Bhatt', 'starrer', 'Dear', 'Zindagi', 'official', 'Twitter', 'page', 'Red', 'Chillies', 'Entertainment', 'production', 'house', u'head', 'Shah', 'Rukh', 'Gauri', 'Khan', u'share', 'first', 'look', 'Life', 'ride', 'gotta', 'keep', u'move', 'Presenting', '#DearZindagiFirstLook', u'caption', 'poster', u'feature', 'Alia', 'SRK', u'rid', 'bicycle', 'film', 'second', 'directorial', 'venture', 'Gauri', 'Shinde', u'make', 'successful', 'debut', 'English', 'Vinglish', 'Shinde', u'pen', 'story', 'co-producing', 'project', 'trailer', 'film', 'tomorrow', 'According', u'report', 'SRK', 'help', 'Alia', 'navigate', 'life', 'relationships', 'Dear', 'Zindagi', u'schedule', 'release', 'November', u'star', 'Aditya', 'Roy', 'Kapur', 'Ali', 'Zafar', 'Kunal', 'Kapoor', 'Angad', 'Bedi', 'Karan', 'Johar', 'board', 'producers']

Part of Speech tagging

As our mind sees different forms of a verb (like eat, eats, eating) and singular & plural words (cat & cats) as a single raw form (eat and cat) while selecting candidates, we will convert every word into its raw format too. This is called lemmatization and we are using NLTK wordnet lemmatizer for converting every word in its raw format.
Part of speech tagging plays an essential role in candidate selection and we are using NLTK POS tagger for the same. The results are :

[('Mumbai', 'NNP'), ('The', 'DT'), ('first', 'JJ'), ('poster', 'NN'), ('of', 'IN'), ('Shah', 'NNP'), ('Rukh', 'NNP'), ('Khan', 'NNP'), ('Alia', 'NNP'), ('Bhatt', 'NNP'), ('starrer', 'NN'), ('Dear', 'NNP'), ('Zindagi', 'NNP'), (u'be', 'VB'), ('out', 'RP'), ('The', 'DT'), ('official', 'JJ'), ('Twitter', 'NNP'), ('page', 'NN'), ('of', 'IN'), ('Red', 'NNP'), ('Chillies', 'NNP'), ('Entertainment', 'NNP'), ('the', 'DT'), ('production', 'NN'), ('house', 'NN'), (u'head', 'NN'), ('by', 'IN'), ('Shah', 'NNP'), ('Rukh', 'NNP'), ('and', 'CC'), ('Gauri', 'NNP'), ('Khan', 'NNP'), (u'share', 'NN'), ('the', 'DT'), ('first', 'JJ'), ('look', 'NN'), ('Life', 'NNP'), (u'be', 'VB'), ('a', 'DT'), ('ride', 'NN'), ('and', 'CC'), ('you', 'PRP'), ('gotta', 'VBP'), ('keep', 'VB'), (u'move', 'NN'), ('Presenting', 'VBG'), ('#DearZindagiFirstLook', 'NN'), ('they', 'PRP'), (u'caption', 'VBP'), ('the', 'DT'), ('poster', 'NN'), ('which', 'WDT'), (u'feature', 'NN'), ('Alia', 'NNP'), ('and', 'CC'), ('SRK', 'NNP'), (u'rid', 'JJ'), ('bicycle', 'NN'), ('The', 'DT'), ('film', 'NN'), (u'be', 'VB'), ('second', 'JJ'), ('directorial', 'JJ'), ('venture', 'NN'), ('of', 'IN'), ('Gauri', 'NNP'), ('Shinde', 'NNP'), ('who', 'WP'), (u'make', 'VBP'), ('a', 'DT'), ('successful', 'JJ'), ('debut', 'NN'), ('with', 'IN'), ('English', 'NNP'), ('Vinglish', 'NNP'), ('Shinde', 'NNP'), (u'have', 'VBP'), ('also', 'RB'), (u'pen', 'VBP'), ('the', 'DT'), ('story', 'NN'), ('and', 'CC'), (u'be', 'VB'), ('even', 'RB'), ('co-producing', 'VBG'), ('the', 'DT'), ('project', 'NN'), ('The', 'DT'), ('trailer', 'NN'), ('of', 'IN'), ('the', 'DT'), ('film', 'NN'), ('will', 'MD'), ('be', 'VB'), ('out', 'RB'), ('tomorrow', 'NN'), ('According', 'VBG'), ('to', 'TO'), (u'report', 'NN'), ('SRK', 'NNP'), ('will', 'MD'), ('help', 'VB'), ('Alia', 'NNP'), ('navigate', 'VB'), ('her', 'PRP$'), ('life', 'NN'), ('and', 'CC'), ('relationships', 'NNS'), ('Dear', 'NNP'), ('Zindagi', 'NNP'), (u'schedule', 'NN'), ('to', 'TO'), ('release', 'VB'), ('on', 'IN'), ('November', 'NNP'), ('25', 'CD'), ('also', 'RB'), (u'star', 'NN'), ('Aditya', 'NNP'), ('Roy', 'NNP'), ('Kapur', 'NNP'), ('Ali', 'NNP'), ('Zafar', 'NNP'), ('Kunal', 'NNP'), ('Kapoor', 'NNP'), ('and', 'CC'), ('Angad', 'NNP'), ('Bedi', 'NNP'), ('Karan', 'NNP'), ('Johar', 'NNP'), (u'be', 'VB'), ('also', 'RB'), ('on', 'IN'), ('board', 'NN'), ('as', 'IN'), ('one', 'CD'), ('the', 'DT'), ('producers', 'NNS')]

We make a parts of speech dictionary, which has lemmatized words, as keys and their parts of speech as value.



Bigram candidates:

[('first', 'poster'), ('Shah', 'Rukh'), ('Rukh', 'Khan'), ('Khan', 'Alia'), ('Alia', 'Bhatt'), ('Bhatt', 'starrer'), ('starrer', 'Dear'), ('Dear', 'Zindagi'), ('official', 'Twitter'), ('Twitter', 'page'), ('Red', 'Chillies'), ('Chillies', 'Entertainment'), ('production', 'house'), ('house', u'head'), ('Shah', 'Rukh'), ('Gauri', 'Khan'), ('first', 'look'), ('gotta', 'keep'), ('keep', u'move'), (u'feature', 'Alia'), ('SRK', u'rid'), (u'rid', 'bicycle'), ('second', 'directorial'), ('directorial', 'venture'), ('Gauri', 'Shinde'), ('successful', 'debut'), ('English', 'Vinglish'), ('help', 'Alia'), ('Alia', 'navigate'), ('Dear', 'Zindagi'), ('Zindagi', u'schedule'), (u'star', 'Aditya'), ('Aditya', 'Roy'), ('Roy', 'Kapur'), ('Ali', 'Zafar'), ('Kunal', 'Kapoor'), ('Angad', 'Bedi'), ('Karan', 'Johar')]

Trigram candidates:

[('poster', 'of', 'Shah'), ('Shah', 'Rukh', 'Khan'), ('Rukh', 'Khan', 'Alia'), ('Khan', 'Alia', 'Bhatt'), ('Alia', 'Bhatt', 'starrer'), ('Bhatt', 'starrer', 'Dear'), ('starrer', 'Dear', 'Zindagi'), ('official', 'Twitter', 'page'), ('page', 'of', 'Red'), ('Red', 'Chillies', 'Entertainment'), ('production', 'house', u'head'), (u'head', 'by', 'Shah'), ('Rukh', 'and', 'Gauri'), (u'share', 'the', 'first'), ('gotta', 'keep', u'move'), (u'caption', 'the', 'poster'), ('poster', 'which', u'feature'), ('Alia', 'and', 'SRK'), ('SRK', u'rid', 'bicycle'), ('film', u'be', 'second'), ('second', 'directorial', 'venture'), ('venture', 'of', 'Gauri'), (u'make', 'a', 'successful'), ('debut', 'with', 'English'), (u'pen', 'the', 'story'), ('co-producing', 'the', 'project'), ('According', 'to', u'report'), ('SRK', 'will', 'help'), ('help', 'Alia', 'navigate'), ('navigate', 'her', 'life'), ('life', 'and', 'relationships'), ('Dear', 'Zindagi', u'schedule'), (u'schedule', 'to', 'release'), ('release', 'on', 'November'), (u'star', 'Aditya', 'Roy'), ('Aditya', 'Roy', 'Kapur'), ('Kapoor', 'and', 'Angad')]

Feature and score calculation

In this step, we take statistical and linguistic properties for each key phrase candidate, for the three lists separately.

The features computed are:

1) Phrase frequency:

This feature gives preference to the phrases appearing more often in the article.

Phrase frequency = (number of times phrase occurs in article) / (number of keyword phrases in the list)

2) POS value:

This feature gives preference to a phrase, if it has more number of nouns or proper nouns in it. If the phrase is a noun or a proper noun, the pos value is 1 else 0.25

POS value = (Number of nouns or proper noun words * 1.0 +Number of other words*0.25)/Total number of words in the phrase

3) Phrase depth:

This feature gives preference to phrases appearing in the initial lines of the document

Phrase depth = 1- (number of words preceding the phrase's first appearance / Total number of words in document)

Score Calculation: Equal weight of 0.33 has been given to each feature. These were later modified giving POS value more weightage.

Score= (weight1*feature1+weight2*feature2+weight3*feature3) / (weight1+weight2+weight3)

Tags Selection

Top five tags from each list with score greater than a threshold value (Here 0.5) are taken as final recommended tags.

Unigram tags:

```
[('Mumbai', 0.6704980842911878), ('poster', 0.6671352573113922), ('Shah', 0.6623390942418479), ('Rukh', 0.6599410127070757), ('Alia', 0.6589762672620525)]
```

Bigram tags:

```
[(('Shah', 'Rukh'), 0.6730060445230724), (('Rukh', 'Khan'), 0.6614329942503318), (('Khan', 'Alia'), 0.6586318738021524), (('Dear', 'Zindagi'), 0.6561993218339968), (('Alia', 'Bhatt'), 0.6558307533539732)]
```

Trigram tags:

```
[(('Shah', 'Rukh', 'Khan'), 0.6626037802508391), (('Rukh', 'Khan', 'Alia'), 0.65933580639463), (('Khan', 'Alia', 'Bhatt'), 0.6560678325384208), (('Alia', 'Bhatt', 'starrer'), 0.6527998586822117), (('Bhatt', 'starrer', 'Dear'), 0.6495318848260025)]
```

The tags we have received are far from the expected tags "Shah Rukh Khan", "Alia Bhatt", "Dear Zindagi", "first poster", "Red Chilles Entertaiment".

A few apparent problems are:

1) The subsets of a name are being received in the tags. e.g. For Shah Rukh Khan, we are getting Shah, Rukh, Shah Rukh, Rukh Khan, Shah Rukh Khan.
2) Two different proper nouns have been clubbed into a single tag, e.g. Rukh Khan Alia, Khan Alia Bhatt
3) The proper nouns have been clubbed with nouns and adjectives. E.g. Bhatt starrer dear

In the subsequent parts, we will try a few enhancements to improve the quality of our tags.

Modification Techniques

1) Combining post title and post content: The title of a post is a summary of the entire article and would consist of important keywords itself. In some articles, it was noticed that there were some good keywords present in the post title but not in post content. So, the modified content was made by post title followed by wrangled content and further operations were done on modified content.

For the above article, the post title was

Tags	Before	After
Unigram	[('Mumbai', 0.6704980842911878), ('poster', 0.6671352573113922), ('Shah', 0.6623390942418479), ('Rukh', 0.6599410127070757), ('Alia', 0.6589762672620525)]	[('Dear', 0.677536231884058), ('Zindagi', 0.6752214170692432), ('look', 0.6669685990338163), ('UNVEILED', 0.6610305958132046), ('Mumbai', 0.6587157809983897)]

Tags	Before	After
Bigram	[(('Shah', 'Rukh'), 0.6730060445230724), (('Rukh', 'Khan'), 0.6614329942503318), (('Khan', 'Alia'), 0.6586318738021524), (('Dear', 'Zindagi'), 0.6561993218339968), (('Alia', 'Bhatt'), 0.6558307533539732)]	[(('Dear', 'Zindagi'), 0.6904761904761905), (('look', 'UNVEILED'), 0.6664730933023617), (('Shah', 'Rukh'), 0.6608594657375145), (('Rukh', 'Khan'), 0.6502129307007356), (('Khan', 'Alia'), 0.6475029036004646)]
Trigram	[(('Shah', 'Rukh', 'Khan'), 0.6626037802508391), (('Rukh', 'Khan', 'Alia'), 0.65933580639463), (('Khan', 'Alia', 'Bhatt'), 0.6560678325384208), (('Alia', 'Bhatt', 'starrer'), 0.6527998586822117), (('Bhatt', 'starrer', 'Dear'), 0.6495318848260025)]	[(('Shah', 'Rukh', 'Khan'), 0.6527777777777778), (('Rukh', 'Khan', 'Alia'), 0.6496031746031746), (('Khan', 'Alia', 'Bhatt'), 0.6464285714285715), (('Alia', 'Bhatt', 'starrer'), 0.6432539682539683), (('Bhatt', 'starrer', 'Dear'), 0.640079365079365)]

We can see that the score of phrases like Dear, Zindagi, Unveiled, Dear Zindagi, etc. have increased as they occur in title.
On a large scale, the improvement in the tags was not very big and so, we revoked this change.

2. Considering proper noun keywords separately: In the above trigram tags, we are getting [Bhatt, Starrer, Dear] as a candidate which should not have been considered. In a tag, the proper noun words should not be clubbed with words having part of speech other than proper noun. So, we made a separate list which consists of only adjacent proper nouns clubbed together. The unigram, bigram and trigram list were made with the same process previously followed without considering proper nouns. The phrases from proper noun list were then sent into the three lists (unigram, bigram and trigram) depending on the number of words in the phrase. The phrases having more than three words e.g. **Chhatrapati Shivaji International Terminal** were included in trigram list.

Also, in the wrangling step, we no longer replace punctuations like '-!?_'@$' with space as they help in separating proper noun phrases e.g. "Shah Rukh Khan – Alia Bhatt" can be separated into Shah Rukh Khan and Alia Bhatt by using '-' in between.

Proper Noun list:

```
[['Shah', 'Rukh', 'Khan'], ['Alia', 'Bhatt'], ['Dear', 'Zindagi'], ['Twitter'], ['Red', 'Chillies', 'Entertainment'], ['Shah', 'Rukh'], ['Gauri', 'Khan'], ['Life'], ['Alia'], ['SRK'], ['Gauri', 'Shinde'], ['English', 'Vinglish'], ['Shinde'], ['SRK'], ['Alia'], ['Dear', 'Zindagi'], ['November'], ['Aditya', 'Roy', 'Kapur'], ['Ali', 'Zafar'], ['Kunal', 'Kapoor'], ['Angad', 'Bedi'], ['Karan', 'Johar']]
```

Result

Tags	Before	After
Unigram	[('Mumbai', 0.6704980842911878), ('poster', 0.6671352573113922), ('Shah', 0.6623390942418479), ('Rukh', 0.6599410127070757), ('Alia', 0.6589762672620525)]	[('poster', 0.6716450216450217), ('Alia', 0.6573593073593074), ('starrer', 0.6465367965367966), ('Twitter', 0.62987012987013), ('page', 0.6274891774891775)]
Bigram	[(('Shah', 'Rukh'), 0.6730060445230724), (('Rukh', 'Khan'), 0.6614329942503318), (('Khan', 'Alia'), 0.6586318738021524), (('Dear', 'Zindagi'), 0.6561993218339968), (('Alia', 'Bhatt'), 0.6558307533539732)]	[(('Shah', 'Rukh'), 0.6714285714285715), (('Dear', 'Zindagi'), 0.6678571428571428), (('Alia', 'Bhatt'), 0.6603174603174603), (('production', 'house'), 0.6214285714285714), (('house', u'head'), 0.6186507936507938)]
Trigram	[(('Shah', 'Rukh', 'Khan'), 0.6626037802508391), (('Rukh', 'Khan', 'Alia'), 0.65933580639463), (('Khan', 'Alia', 'Bhatt'), 0.6560678325384208), (('Alia', 'Bhatt', 'starrer'), 0.6527998586822117), (('Bhatt', 'starrer', 'Dear'), 0.6495318848260025)]	[(('Shah', 'Rukh', 'Khan'), 0.6712655424970194), (('Red', 'Chillies', 'Entertainment'), 0.6259580991313236), (('production', 'house', u'head'), 0.6194856072219384)]

This removed many redundant tags from our list e.g. [Khan, Alia, Bhatt], [Bhatt, starrer, Dear]

3) Proper nouns given importance in bigrams and trigrams: We can see that in bigrams and trigrams, there are noun tags like 'production house head' which should not come. So, in POS value feature, only the proper nouns were given a weightage of 1. Nouns in bigrams and trigrams will have a weightage of 0.25.

Tags	Before	After
Unigram	[('poster', 0.6716450216450217), ('Alia', 0.6573593073593074), ('starrer', 0.6465367965367966), ('Twitter', 0.62987012987013), ('page', 0.6274891774891775)]	[('poster', 0.6716450216450217), ('Alia', 0.6573593073593074), ('starrer', 0.6465367965367966), ('Twitter', 0.62987012987013), ('page', 0.6274891774891775)]
Bigram	[(('Shah', 'Rukh'), 0.6714285714285715), (('Dear', 'Zindagi'), 0.6678571428571428), (('Alia', 'Bhatt'), 0.6603174603174603), (('production', 'house'), 0.6214285714285714), (('house', u'head'), 0.6186507936507938)]	[(('Shah', 'Rukh'), 0.6714285714285715), (('Dear', 'Zindagi'), 0.6678571428571428), (('Alia', 'Bhatt'), 0.6603174603174603), (('Gauri', 'Khan'), 0.601984126984127), (('Gauri', 'Shinde'), 0.5214285714285715)]
Trigram	[(('Shah', 'Rukh', 'Khan'), 0.6712655424970194), (('Red', 'Chillies', 'Entertainment'), 0.6259580991313236), (('production', 'house', u'head'), 0.6194856072219384)]	[(('Shah', 'Rukh', 'Khan'), 0.6712655424970194), (('Red', 'Chillies', 'Entertainment'), 0.6259580991313236)]

It is apparent that many redundant noun phrases in bigrams and trigrams have been removed e.g. [Production, House, Head]

4) Removing subsets of a multigram in proper noun list: We can see that both Shah Rukh and Shah Rukh Khan are being chosen as tags instead of Shah Rukh Khan. The same is also true for Alia and Alia Bhatt. So, after making proper noun list, we delete the smaller subsets of proper noun phrase from the list.

Tags	Before	After
Proper Noun List	[['Shah', 'Rukh', 'Khan'], ['Alia', 'Bhatt'], ['Dear', 'Zindagi'], ['Twitter'], ['Red', 'Chillies', 'Entertainment'], ['Shah', 'Rukh'], ['Gauri', 'Khan'], ['Life'], ['Alia'], ['SRK'], ['Gauri', 'Shinde'], ['English', 'Vinglish'], ['Shinde'], ['SRK'], ['Alia'], ['Dear', 'Zindagi'], ['November'], ['Aditya', 'Roy', 'Kapur'], ['Ali', 'Zafar'], ['Kunal', 'Kapoor'], ['Angad', 'Bedi'], ['Karan', 'Johar']]	[['Shah', 'Rukh', 'Khan'], ['Alia', 'Bhatt'], ['Dear', 'Zindagi'], ['Twitter'], ['Red', 'Chillies', 'Entertainment'], ['Gauri', 'Khan'], ['Life'], ['SRK'], ['Gauri', 'Shinde'], ['English', 'Vinglish'], ['SRK'], ['Dear', 'Zindagi'], ['November'], ['Aditya', 'Roy', 'Kapur'], ['Ali', 'Zafar'], ['Kunal', 'Kapoor'], ['Angad', 'Bedi'], ['Karan', 'Johar']]
Unigram	[('poster', 0.6716450216450217), ('Alia', 0.6573593073593074), ('starrer', 0.6465367965367966), ('Twitter', 0.62987012987013), ('page', 0.6274891774891775)]	[('poster', 0.6723443223443223), ('starrer', 0.6468864468864469), ('Twitter', 0.6302197802197802), ('page', 0.6278388278388279), ('production', 0.6135531135531136)]
Bigram	[(('Shah', 'Rukh'), 0.6714285714285715), (('Dear', 'Zindagi'), 0.6678571428571428), (('Alia', 'Bhatt'), 0.6603174603174603), (('Gauri', 'Khan'), 0.601984126984127), (('Gauri', 'Shinde'), 0.5214285714285715)]	[(('Dear', 'Zindagi'), 0.6694444444444444), (('Alia', 'Bhatt'), 0.6611111111111112), (('Gauri', 'Khan'), 0.6027777777777779), (('Gauri', 'Shinde'), 0.5222222222222223), (('English', 'Vinglish'), 0.5027777777777778)]

Tags	Before	After
Trigram	[(('Shah', 'Rukh', 'Khan'), 0.6712655424970194), (('Red', 'Chillies', 'Entertainment'), 0.6259580991313236)]	[(('Shah', 'Rukh', 'Khan'), 0.6712655424970194), (('Red', 'Chillies', 'Entertainment'), 0.6259580991313236)]

We can see "Alia" and "Shah Rukh" have been removed from the tags.
Now, if we take top 2 tags from each list, they would consist of poster, starrer, Dear Zindagi, Alia Bhatt, Shah Rukh Khan, Red Chillies Entertainment.
These are the tags which aptly describe the article.
Now we checked tags for all the articles and tried a few methods to improve tags for them all.

5) Using TFIDF instead of phrase frequency:

The trigram tags for an article were the following:

```
[(('Pearl', 'V', 'Puri'), 0.633289124668435), (('popular', 'TV', 'show'), 0.4110669024462128), ((u'take', 'a', 'toll'), 0.3939728853521957), (('toll', 'on', 'actor'), 0.3896993810786914 5), (('Actor', 'Anshuman', 'Malhotra'), 0.3854258768051872)]
```

Here, we would want the tag 'Actor Anshuman Malhotra' to have a better score than the tags like "popular TV show" and "take a toll". We tried TFIDF instead of phrase frequency for that.

```
[(('Pearl', 'V', 'Puri'), 0.658831908831909), (('popular', 'TV', 'show'), 0.43660968660968663), (('Actor', 'Anshuman', 'Malhotra'), 0.410968660968661), (('take', 'a', 'toll'), 0.3896011396011397), (('toll', 'on', 'actor'), 0.3874643874643875)]
```

It did not improve the tags very much and latency increased, as we had to go through all the documents for recommending tags of a single article. So, we revoked this change.

6) Stanford log linear POS tagger: In many articles, we were not getting good tags, because our POS tagger was not giving correct results. E.g. in Mahesh Sharma, Mahesh was tagged as adjective in an article. We tried Stanford log linear POS tagger. The results were a little better but the latency was very high as the tagger code was in Java and our code was in python. So, we revoked this change.

7) Allowing numbers at the end of a proper noun phrase: Numbers can appear as last word in a proper noun phrase like Big Boss 10, Kahaani 2

8) Verbs not to be selected while doing candidate selection: leaving only noun and adjective thus removing more redundant tags

9) Increasing the stopwords list: NLTK stopwords list was not sufficient to remove all frequently occurring words. So, we made our own extensive word list which should not occur in a tag.

10) Phrase frequency and phrase depth for multigrams to be calculated based on individual word: While deleting smaller subsets in a proper noun list, we created a problem. While calculating phrase frequency, now "Shah Rukh Khan" phrase's frequency will only be shown as 1 while it has come at another place as Shah Rukh too. So, instead of calculating the frequency of whole phrase, we calculate the frequency of individual word, sum them for a phrase and then divide by the number of words to get the average frequency. Similarly, phrase frequency will be calculated as the distance of the first word of the phrase from the start.

11) Spacy POS Tagger: POS tagging was the major reason for getting bad quality tags. We replaced NLTK tagger with spacy POS tagger and received much better results compared to NLTK and Stanford tagger.

12) "Name Entity Recognition Presence" Feature: There were some redundant proper nouns like month names, positions (like secretary) coming in within our tags. To discourage them, we made a list of name entities like person names, company names using spacy.

NER Presence = Number of phrase words in NER list / number of words in the phrase.

We replaced it later by increasing weightage for POS value feature.

13) Modifying dictionary convert function: When we convert our parts of speech tags into dictionary, each word gets a single tag. In many cases, a single word in a document could have different parts of speech, because of parts of speech tagger not being 100% efficient. Thus "Dear" in "Dear Zindagi" could be a proper noun at one place and adjective at a later place. Converting them into dictionary will consider "Dear" as an adjective, thus not taking "Dear Zindagi" as a proper noun candidate.

To solve this problem, we modified our dictionary convert function. While getting a keyword and parts of speech value, if the word has a proper noun, parts of speech, anywhere in the document, the value would remain as proper noun, irrespective of other parts of speech values. After proper noun, preference goes to noun. The other parts of speech values will be updated as same.

14) Title presence feature: Title of a document consists of important keywords themselves. So, we added a new extra feature, which gives preference to phrases which are in title.

Title Presence = Number of phrase words in title / Number of words in the phrase

15) Taking proper noun from name entity recognition: In an article, we were getting "Prime Minister Narendra Modi as feature"; so, we tried name entity recognition, to get only Narendra Modi as a tag. However, it was not giving all the proper nouns properly. So, we revoked it.

16) Making a list of important words: Words like surgical strike, Indian Army, have increasing score for them.

After applying these changes, the tags we received were good enough for recommendation.

Tag Corpus Reduction

The purpose of tags is not just to summarize, but organize too. For 1000 articles, we got 5140 tags with 2445 unique tags. The number of tags, which appear in a single article is 1701.

To reduce the number, we took the following steps:

1) Removing noun bigrams and trigrams which are not in title: Making this modification removed many redundant noun multigrams.

In the tag corpus, there were many similar proper noun tags which represent the same person like "Prime Minister Narendra Modi", "PM Narendra Modi", "Narendra Modi". To consolidate these tags in a single tag, we took the following approaches:

2) Jaccard Index:

Jaccard Index = (Number of common words in two tags) / (Number of unique words in the union of the two tags)

Jaccard index of all the possible phrase pairs was found, and if the jaccard index for a pair was greater than 0.6, then the phrases were replaced by the phrase having greater frequency.

For 50000 article, it reduced the unique tags from 48000 to 40000. However, there were some false replacements too like "Pardesh me hai Mera Dil" and "Is desh me hai Mera Dil". Also, the latency was very high. So, it was not used.

3) Levenshtein distance: The 'Levenshtein distance' between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. The results were not good enough. So, it was not used in final implementation.

4) Wikipedia search: 'The Wikipedia' title of the phrases was searched, and the phrase was replaced by it. However, for many phrases, Wikipedia title was not present and so, we would get incorrect search results. This was therefore revoked.

Future Objectives:

There is a scope of improvement in every solution, and our system is not an exception to it! Although, the tags our system recommends, are as per industry standards, there are still a few improvements which can be applied to it:

1) Reducing the unique tag corpus to organize the articles more efficiently.

2) Changing the feature weights: At present, we have taken the feature weights to get the best tags. However, the quality of the tags has only been checked by manually visualizing samples and could be far from the best. We are storing the tags we have recommended, and the tags selected by the editor in mongoDB. After getting a considerable amount of such data, we can apply machine learning to optimize the weights, to recommend better tags.

3) Improving proper noun keywords: Spacy is doing a very good job of tagging parts of speech of every word, but there are still some areas where it can be improved. E.g. Spacy tags each of the words "Prime Minister Narendra Modi" as proper noun. So, our tag recommender makes it a single tag, while a better system should recommend two tags "Prime Minister" and "Narendra Modi". We need a part of speech tag recommender which could also identify the words, which occur together. E.g. the system gives the output as Prime//NNPC Minister//NNP Narendra//NNPC Modi//NNP. Such a system will improve the quality of tags, as well as, help in reducing tag corpus.

We have now come to the end of the journey of recommending tags for English newspaper articles. If you have any suggestions, do let us know.

We will now modify the system to recommend the tags for news articles in Indian languages.

Hindi News Articles

For Hindi news articles, following changes were made from the English API

1) Part of Speech Tagging: As Spacy part of speech tagging is not available for Hindi language, we tried other taggers.

A) NLTK Hindi Corpus: It consists of 9500 tagged words which was not ample for training a POS tagger. So, the results were not good!

B) RDRPOS Tagger: The results were better than NLTK tagger but still far from a good quality tagger.

C) Creating training corpus: We manually created a corpus of 23000 words for training. Three NLTK taggers were then used on this corpus for training.

TNT tagger – 78% accuracy	**Perceptron Tagger – 82% accuracy**	**CRF tagger - 82 % accuracy**

As perceptron tagger is also used in Spacy, we used the same for our purpose.

The format of the training corpus used is as follows:

1) All symbols are to be considered as separate words
2) Every word of the article should have a part of speech written along with it, separated by two forward slashes i.e. //
3) The different word combinations must be separated by space
4) The article has to be broken into sentences, wherever a new sentence is started based on symbols Like- (. : ?)
5) The different sentences of the article are to be separated by four hyphens i.e. ----

The tagged content was then converted into list of sentences. Each sentence is a list of tuple I.e. (word, POS_tag). This format was then used for training.

Content:

iवाराणसी: पीएम नरेंद्र मोदी के संसदीय क्षेत्र में पछिले दनिों जय गुरुदेव के समागम में हुए हादसे की गाज डीएम और एसएसपी पर गरिने से यहां लोगों में काफी गुस्सा है. लोगों का कहना है कि सरकार अपनी नाकामी दूसरे पर नही थोप सकती है. ट्रांसफर को लेकर लोगों में गुस्सा 15 अक्टूबर को यूपी के वाराणसी में भगदड़ की घटना के बाद जलि के एसएसपी आकाश कुलहरी और फरि डीएम वजिय करिण आनन्द के ट्रांसफर को लेकर यहां आम आदमी के साथ-साथ प्रबुद्धजनों में काफी गुस्सा देखने को मलि. तबादलों को नरिस्त करने की मांग

Part of Speech Tagged content:-

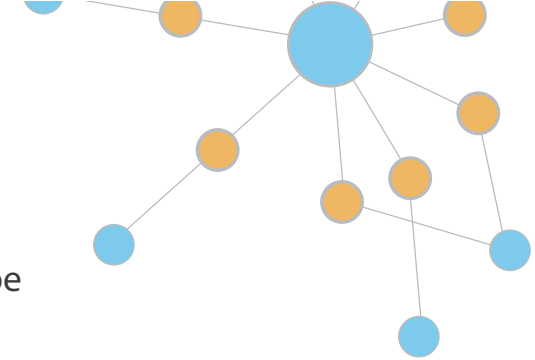
वाराणसी//NNP----पीएम//NN नरेंद्र//NNPC मोदी//NNP के//PSP संसदीय//JJ क्षेत्र//NN में//PSP पछिले//JJ दनिों//NN जय//NNPC गुरुदेव//NNP के//PSP समागम//NN में//PSP हुए//VM हादसे//NN की//PSP गाज//NN डीएम//NN और//CC एसएसपी//NN पर//PSP गरिने//VM से//PSP यहां//PRP लोगों//NN में//PSP काफी//QF गुस्सा//NN है//VM----लोगों//NN का//PSP कहना//VM है//VAUX कि//CC सरकार//NN अपनी//PRP नाकामी//NN दूसरे//PRP पर//PSP नहीं//NEG थोप//VM सकती//VAUX है//VAUX----ट्रांसफर//NN को//PSP लेकर//VM लोगों//NN में//PSP गुस्सा//NN----15//QC अक्टूबर//NNP को//PSP यूपी//NNP के//PSP वाराणसी//NNP में//PSP भगदड़//NN की//PSP घटना//NN के//PSP बाद//NST जलि//NN के//PSP एसएसपी//NN आकाश//NNPC कुलहरी//NNP और//CC फरि//RB डीएम//NN वजिय//NNPC करिण//NNPC आनन्द//NNP के//PSP ट्रांसफर//NN को//PSP लेकर//VM यहां//PRP आम//JJ आदमी//NN के//PSP साथ//NN -//SYM साथ//RDP प्रबुद्धजनों//NN में//PSP काफी//QF गुस्सा//NN देखने//VM को//PSP मलि//VM----तबादलों//NN को//PSP नरिस्त//VM करने//VAUX की//PSP मांग//NN

The different Parts of speeches used are:

S.No.	Tag	Description	Example
1	NN	Common Nouns	लड़का, लड़के
2	NST	Noun Denoting Spatial and Temporal Expressions	ऊपर, पहले
3	NNP	Proper Noun	मोहन, राम, सुरेश
4	PRP	Pronoun	वह, वो, उसे

5	DEM	Demonstrative	वह, वो, उसे
6	VM	Verb main	खाता, सोता, रोता,खाते , सोते
7	VAUX	Verb Auxiliary	है , कर
8	JJ	Adjective	पुरानी
9	RB	Adverb	धीरे
10	PSP	Postposition	को
11	RP	Particles	भी, तो
12	QF	Quantifiers	ऊपर, पहले
13	QC	cardinals	एक, दो, तीन
14	CC	Conjuncts	क्यों, क्या
15	WQ	Question Words	पुरानी
16	INTF	Intensifier	बहुत , थोडा, कम
17	INJ	Interjection	अरे , हाय
18	NEG	Negative	ना
19	SYM	Symbol	? , ; : !
20	XC	Compounds	केंद्र/NNC सरकार/NN
21	RDP	Reduplications	धीरे /RB धीरे /RDP
22	ECH	Echo Words	चाय-व्हाय
23	UNK	Foreign Words	English,বাংলা

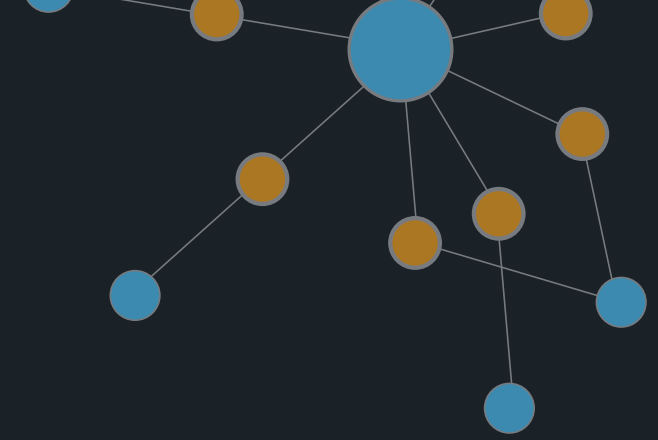
Converting Hindi tags to English:



Hindi tags must be converted into English, so that both English and Hindi versions of articles can be associated with same tags. It is done by using following heuristics:

A) All the proper noun phrases are transliterated into English. At first, we used Sanskrit library and Indic NLP library for getting transliterated texts. They did not give good enough results. We did our own conversion of each Hindi character into English, and then elastic search was applied to get similar tag from our English tag corpus.

B) All the other phrases are translated using Microsoft translator.



piStats | Create • Deliver • Analyze • Personalize

www.pi-stats.com