NLP Team -- Q2 Report

Topic Modelling -- Indian Languages

Approaches

Explored two main approaches for topic modelling.

1. Topicvec -- this is a new technique of doing topic modelling. However the reference implementation provided by the creators of this technique only worked for English language. For other languages significant other changes needed to be made in the algorithm. We tested this approach for English, however decided not to use it for Indian languages as the effort involved was very high and there was no surety that the topicvec approach would perform well for Indian languages.
2. LDA -- This is a tested technique and works well in most of the cases. We implemented LDA for topic modelling for Indian languages.

Tasks Completed

1. Train topic modelling models for Hindi, Kannada, Tamil, Telugu and Malayalam.
2. Create a stop words list. There were no good publicly available stop words list for Indian languages. We had to go through multiple training iterations of topics generated and removed the words which looked like stop words.
3. Use stemmer. Tried multiple stemmers but most of them did not work well for Indian languages. Finally used a stemmer ‘Polyglot’. It is not very good, but did show improvement in overall performance.
4. Create test framework. We tried couple of test approaches.
    a. In the first approach, we labeled articles in multiple categories. Then measured similarity(using topic modelling results) of articles in a given category with all the articles in the same category as well as articles in other categories. The expectation was that articles in same category will show higher similarity compared to articles in other categories. This approach worked, however the problem was that we did not have some benchmark to test this number against.
    b. In the second approach, we labeled articles in multiple categories. For each article we found out N most similar articles. Then checked how many of these N articles belonged in the same category as the test article category. Ideally this should be 100%. We decided to go with this approach.

Results

Hindi

Category	**10**	**5**	**3**
lifestyle	8.84	9.32	9.33

business	7.44	7.8	7.6
entertainment	7.9	7.8	8.06
sports	9.44	9.84	9.93
politics	7.46	8.28	8.53
technology	8.38	8.52	8.46

Tamil

	10	**5**	**3**
health	8.96	9.08	9.26
business	8.7	9.16	9.4
entertainment	9.06	9.2	9
sports	9.44	9.64	9.66
politics	8.04	8.28	8.13
technology	8.36	8.56	8.66

Malayalam

	10	**5**	**3**
health	7.96	8.16	8.46
business	8.32	8.32	8.26
entertainment	9.46	9.48	9.53
sports	9.64	9.6	9.6
politics	8.52	8.92	9.2
technology	7.24	7.56	7.46

Kannada

	10	**5**	**3**
health	9.3	9.2	9.2
business	8.52	9.2	9.4
entertainment	9.02	9.2	9.46
sports	8.9	8.84	8.93
politics	8.82	8.88	9.06

technology	7.3	7.76	7.86

Telugu

	10	**5**	**3**
health	9.94	9.96	9.93
business	9.82	9.24	9.46
entertainment	8.46	8.72	8.8
sports	9.94	9.92	10
politics	8.76	9.08	8.93
technology	8.88	8.88	8.93

Keyword Extraction

Approaches

1. For English initially we used third party services like TextRazor and Cortical.io. Later we started using open source libraries like textacy. We tried multiple algorithms like TextRank, SingleRank, SGRank etc. Analysis showed that no one algorithm was perfect and different algorithm performed well in different scenarios.
2. For Indian languages we used TF-IDF based approach. The approach worked pretty well for Hindi, however for other Indian languages the performance was average. One major reason for this is that South Indian languages(Kannada, Tamil, Telugu and Malayalam) are highly inflectional(that is words get modified in different contexts). To resolve this problem we need to use good stemmers. However there are no publicly available good stemmers for Indian languages. We tried a few but all of them gave results which was at best average.

Work Done

1. Implement and test multiple techniques for English.
2. Train TF-IDF models for Hindi, Kannada, Tamil, Telugu and Malayalam
3. Create a test framework. This was one of the most challenging tasks, because we did not know many of the languages. We were able to find some news sites which provided keywords along with articles. We used them only to find very poor performance results. It turned out that the keywords were not very representative and we had to translate each of the article through Google translate and find out the keywords manually.

4. Use of stemmers. We used Polyglot stemmer which did not work very well for keyword extraction. Tried a few more stemmers but did not find any good stemmer which can work well for Indian languages.

Results

Tamil

Not stemmed

10 keywords	5 keywords	3 keywords
38	54	73

Stemmed

10 keywords	5 keywords	3 keywords
41	60	73

Telugu

Not stemmed

10 keywords	5 keywords	3 keywords
38	48	60

Stemmed

10 keywords	5 keywords	3 keywords
40	44	50

Kannada

Not stemmed

10 keywords	5 keywords	3 keywords
35	48	56

Stemmed

10 keywords	5 keywords	3 keywords
44	60	73

Malayalam

Not stemmed

10 keywords	5 keywords	3 keywords
38	48	62

Stemmed

10 keywords	5 keywords	3 keywords
41	57	70

Hindi

10 keywords	5 keywords	3 keywords
64	82	90

Support for Hindi in Lemon Keyboard App

Tasks Completed

1. Autosuggest dictionary for Hindi. Created a bigram model for Hindi using training corpus of around 150k articles. Generated a dictionary in a format requested by the lemon keyboard team.
2. Vowel combination mapping. Provided a mapping of combination of consonants of Hindi with vowels.