Using N-grams to Process Hindi Queries with Transliteration Variations

Anand Natrajan, Allison L. Powell
and James C. French

Technical Report No. CS-97-17
July 16, 1997

Contact : anand@virginia.edu

Web : ftp://ftp.cs.virginia.edu/pub/techreports/CS-97-17.ps.Z

Using N-grams to Process Hindi Queries with Transliteration Variations

Anand Natrajan*, Allison L. Powell, James C. French†
Email: {anand, alp4g, french}@virginia.edu
Dept. of Computer Science, University of Virginia, Thornton Hall, Charlottesville, VA 22901

Abstract

Retrieval systems based on N-grams have been used as alternatives to word-based systems. N-grams offer a language-independent technique that allows retrieval based on portions of words. A query that contains misspellings or differences in transliteration can defeat word-based systems. N-gram systems are more resistant to these problems. We present a retrieval system based on N-grams that uses a collection of Hindi songs. Within this retrieval system, we study the effect of varying N on retrievability. Additionally, we present an alternative spell-checking tool based on N-grams. We conclude with a discussion of the number of N-grams produced by different values of N for different languages and a discussion of the choice of N.

1 Introduction

N-grams are consecutive overlapping N-character sequences formed from an input stream. In Figure 1, we explain a few of the many techniques for extracting N-grams by means of an example with N = 3 and the string: “salt in the coffee”. In the first strategy (a), we consider the entire text stream after collapsing multiple runs of white spaces into one space (shown as ϕ in the figure). In the second (b), we break the string into individual words and pad with one space before and after each word. Finally in (c), we break the string into individual words. One- or two-letter words are left unchanged. Note that the successive strategies result in fewer N-grams and the different representations have considerable overlap. Strictly speaking, only method (a) is truly language-independent because it avoids the concept of “words.” While all these techniques and variations on them have been used in the literature, we conducted our experiments using (c).

Strategy	3-grams
(a)	sal alt ltϕ tϕi ϕin inϕ nϕt ϕth the heϕ eϕc ϕco cof off ffe fee
(b)	ϕsa sal alt ltϕ ϕin inϕ ϕth the heϕ ϕco cof off ffe fee feϕ
(c)	sal alt in the cof off ffe fee

FIGURE 1: N-gram strategies

1.1 Background

N-grams have been used as alternatives to word-based retrieval in a number of systems. DeHeer used syntactic traces to demonstrate an efficient strategy for retrieval when thesaurus-based and multi-attribute search techniques are unsuitable [DeH74]. Adams and Meltzer used trigrams and inverted files for exact matches with query terms [Adams93]. They reported 100% recall with very high precision for their experiments and recommended trigram-based search as an acceptable alternative to word-based search and a superior method for retrieval of word fragments. N-grams were used for TREC-2’s retrieval and routing tasks with promising results [Cav94a]. Since N-grams are a language-independent technique, the strategies used for retrieval can be used for document collections in languages other than English [Cav95, Cohen95]. N-grams have been used along with word-based systems for effectively retrieving compound nouns in Korean [Lee96]. Also, N-grams can be used to distinguish between documents in different languages in multi-lingual collections and to gauge topical similarity between documents in the same language [Cav94b, Dam95]. Retrieval based on N-grams has been shown to be robust to spelling errors or differences and garbling of text [Cav95, Huff96, Rob92].

In this paper, we present a retrieval system for a Hindi document collection using N-grams and the vector-space model [Salton75]. We demonstrate an example retrieval system based on N-grams wherein queries could be transliterated differently or garbled. Also, we present a spelling correction system based on N-grams. Trigram-based (N = 3) retrieval gave us the best results.

* Contact author. Tel: (804) 982-2291. Fax: (804) 982-2214.
† This work supported in part by Dept. of Energy Grant no. DE-FG05-95ER25254, NSF grant CDA-9529253 and a NASA Graduate Student Researchers Program fellowship.

1.2 The Problem

Transliteration is a process wherein an input string in some alphabet is converted to a string in another alphabet based on the phonemes in the string. In contrast, *translation* attempts to express the meaning of the string in another language. Subscribers to Hindi film music newsgroups on Usenet access collections of transliterated Hindi songs frequently. The online collections may contain entire songs along with categorical information about lyricists, singers, music directors, etc., or may contain just a few words in the song or any information between these two extremes. Typically, accesses to these collections require the user to enter the first line of the song for which information is required. The first lines of Hindi songs may be regarded as song titles since Hindi film songs do not have explicit titles. These "titles" may be indexed into a song database. Given that many Hindi words can be transliterated in many different ways, the query-to-index mapping may not be accomplished readily and automatically.

The Devnagari[‡] alphabet used for Hindi is different from the Roman alphabet. Absence of direct correspondence between phonemes in the two alphabets results in multiple ways to transliterate a word in Devnagari into a word in the Roman script. There does not exist one accepted system of transliteration — governmental, phonetic, conventional or other — and users are likely to employ individual transliteration schemes convenient to themselves. In addition, users may not be consistent in transliteration. Despite the diversity in transliteration schemes, the transliterations of a single word somewhat resemble each other. For example, some transliterations of the Hindi word for "law" are: "kaanoon", "kanoon", "kaanun" and "kanun". A person conversant in Hindi may read all alike despite the differences in pronunciation evident to an English reader.

Given the plethora of transliteration possibilities, it is unrealistic to expect users to adhere to any one scheme that may be used in a particular retrieval system. However, the lack of a rigid transliteration scheme means that traditional word-based retrieval may be defeated by unexpected transliterations. Therefore, the retrieval system must be able to locate the collection-dependent correct transliteration given the user's input. Our system fulfills this role by presenting the user with a number of ranked responses matching a query. The responses may be viewed either as hits on a search query or candidate correct transliterations of the user's query.

As an additional experiment, we emulated spell-checking as a retrieval operation based on N-grams.

2 Approach

2.1 Methodology

In the course of our work, we encountered many situations where we had to choose between alternate techniques. The various N-gram extraction techniques referred to in Section 1 are an example. Our choice of technique reflected the ease with which we could effect a comparison with word-based techniques. Another example is the choice of N for the N-gram. We present a comparison of results using a range of values for N.

The vector space model for document retrieval represents documents as vectors of (term, weight) tuples [Salton75]. Each "term" is either a word in the collection or a baseword from the set remaining after optional stopword removal and stemming. We chose a similar representation for our techniques, with the difference being that every "term" was an N-gram of the text. Unless otherwise noted, the weight for each term was calculated as $w_{ij} = tf_{ij} \times \log(n/df_i)$, where tf_{ij} is the term frequency of term i in document j or the number of times term i occurs in document j, n is the total number of documents and df_i is the number of documents that contain term i. The choice we faced at this point was whether to use multiple values of N to create the terms which composed a single vector space or to restrict the terms in a vector space to be those for a single value of N. We made this choice on a per-application basis.

2.2 Document Preparation

The document collection we used was a list of the first lines of 3837 Hindi film songs. We used merely the first lines of every song because of availability, and in order to keep the experiment within manageable bounds. Our system can be extended easily on procurement of a full-text collection of songs. Alternatively, the canonical responses from our system may be used as indices into databases of complete songs that use the same transliteration for all song titles.

We chose the vector-space model to represent the song title documents in our collection for our experiments. Each document (song) in our collection had multiple vector representations. For each song, we generated N-grams for

---

[‡] This word itself may be transliterated differently as "Devanagari", "Devanagri" or "Devnagri."

N = 1, 2, 3, 4, 5, 6. Treating these N-grams as vector terms, we built separate vector representations for each song for each value of N. In order to compare N-gram retrieval with word-based retrieval, we built word-based vector representations for every document in our collection. Given our strategy for generating N-grams, we mimicked word-based algorithms by choosing an arbitrarily large value for N, for example, N = 100. Naturally, stopword removal and stemming were not applied because we wanted language-independent techniques.

We built a retrieval system based on N-grams for our particular collection. As shown in the sample query and output presented in Figure 2, users may enter a few words of a desired song as a query. The system responds with a number of songs sorted in decreasing order of similarity with the query; similarity is calculated as the cosine of the angle between the query vector and a song vector. The N-gram strategy can be varied easily in order to compare the effect of changing N. In the example in Figure 2, the output was truncated after ten songs were returned. For experiments described in Section 3, we did not truncate the list of responses.

Query: jane na nazar jigar pehchanay
--------- Results of Search for N = 3 ----------
0.762 *jaane na nazar pehchaane jigar yeh kaun*
0.490 *pehchaan to thi pehchaana nahin maine apne*
0.432 *dil jigar nazar kya hai main to tere liye jaan bhi de doon*
0.415 *bechain nazar betaab jigar*
0.401 *pal bhar ki hi pehchaan mein*
0.365 *nazar ke saamne jigar ke paas koi rehta hai woh ho tum*
0.352 *ek nazar ek nazar*
0.349 *ae jaan-e-jigar*
0.335 *ek nazar bas ek nazar*
0.334 *main dil hoon ek armaan bhara tu aake mujhe pehchaan zara*
--------- Results of Search for N = 5 ----------
0.722 *jaane na nazar pehchaane jigar yeh kaun*
0.505 *pehchaan to thi pehchaana nahin maine apne*
0.425 *pal bhar ki hi pehchaan mein*
0.338 *ajnabi tum jaane pehchaane se lagte ho*
0.333 *main dil hoon ek armaan bhara tu aake mujhe pehchaan zara*
0.329 *dil jigar nazar kya hai main to tere liye jaan bhi de doon*
0.317 *agar bevafa tumko pehchaan jaate khuda ki qasam*
0.314 *ek nazar ek nazar*
0.291 *nazar ke saamne jigar ke paas koi rehta hai woh ho tum*
0.285 *raahi naye naye rasta naya naya*

FIGURE 2: Results for a Hindi song query

2.3 Query Garbling

In the system described in Section 2.2, users are not expected to enter queries transliterated in the same way that the song titles in the collection were transliterated. We were interested in determining a way to provide effective results given potentially non-trivial degrees of query transliteration differences. To quantify the difference in transliteration and to produce a large number of queries, we simulated alternative transliterations by using garbled queries. Garbled strings are generated by randomly replacing, adding or deleting letters or space from the original string. The probability of each character being garbled was deemed the garbling percentage. Thus, a 25% garbled string meant that every character in the original string had a 25% chance of being garbled. Example garbled strings are shown in Figure 3. The 0% garbled string is the original string. Note that the garbled strings bear decreasing resemblance to the original as the garbling percentage increases.

%	Garbled String
0	*madhuban mein raadhika naache*
1	*madhuban mein raadhika naache*
2	*madhuban mein raadhika naac e*
3	*maduban meip raadhika naache*
5	*madubanmein raadhika naacre*
8	*madhubaqn mein raaedhika naache*
10	*madubpn mpin raadhikaknaache*
15	*manhuban meiy ryadhika naachi*
25	*manhubn meyn yaahhira iaalpe*
50	*xhuuban vegn rardeakw naace*

FIGURE 3: Garbled queries

For every song in the collection, we submitted the garbled song as a query to our system. Our goal was to find the original song despite the garbling. In Section 3 we will examine our ability to retrieve the desired song after different amounts of garbling. We report the number of times the desired song is found and the average rank at which the desired song is returned when it is found. In addition we test the effect of different values of N for the N-gram strategy and compare that with the performance of the word-based strategy.

Ideally, the expected song should always appear as the first-ranked response. However, since a 25% garbled query has little resemblance to the original query, we expect retrievability to suffer with increased garbling. In a realistic scenario, users are unlikely to enter such poorly transliterated text. Therefore, our system is expected to perform better in realistic scenarios. In a second experiment designed to demonstrate the performance of our system for realistic queries, we selected six songs from the collection and asked six subjects to transliterate all six songs. These hand-transliterated queries were submitted to our retrieval system to determine whether the desired song was retrieved effectively.

3 Experiments and Results

3.1 Initial Experiment

In our experiments, we sought to evaluate the efficacy of N-grams for retrieving the desired song given a garbled query. Each song in the collection was garbled and submitted as a query. Responses were returned as a ranked list of songs ordered by similarity to the query. In our initial experiment, we chose to evaluate performance based on the rank at which the original song was returned. Due to (a) the probabilistic nature of our garbling routine, (b) the high probability with which we garbled queries for some data points and (c) the short length of some songs, there were cases in which the correct song (and possibly many other songs) had a similarity of zero with the query. There were a number of ways we could have chosen to represent this result. One possibility was to assume that the correct song would be the first zero-similarity song returned. The correct song's rank would then be one greater than the rank of the least similar non-zero-similarity song. However, there was the possibility that this approach would produce artificially favorable results. If a query was garbled sufficiently that it matched no songs, we would report that the correct song was returned at a rank of one. Therefore, we chose to impose a heavy penalty if the correct song had a similarity of zero with the query; we gave it a rank of 3838, the worst rank possible.

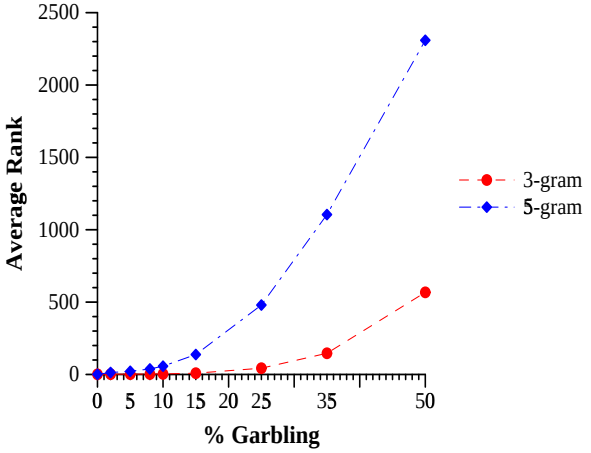

FIGURE 4: Average Rank for Initial Evaluation Measure

For this experiment, we used each of the 3837 song titles as a query, varying the value of N used in the N-grams and varying the garbling percentage. We performed 100 iterations for each garbling percentage for each value of N. The results were averaged to compensate for the probabilistic nature of the garbling routine. The results for 3-grams and 5-grams are shown in Figure 4. The results are favorable for low garbling percentages. On average, for garbling percentages less than or equal to 5% for 5-grams, and 15% for 3-grams our system returns the correct song within ranks 1 to 25. However, performance degrades rapidly for garbling percentages greater than 10% for 5-grams and 25% for 3-grams. This is due to the high penalty that we imposed if the correct song had zero similarity to the query. Even at low garbling percentages, short song titles have the potential to be garbled beyond recognition. Given the single evaluation measure, it was impossible to discern the nature of the poor performance. With a single measure, finding the correct song with very low similarity and not finding the correct song at all can be reported similarly. Therefore, we refined our evaluation measure, as described in Section 3.2**.**

3.2 Refined Experiment

Our refined experiments were performed using the same combinations of parameters used for the initial experiments. The data points from the initial experiment were sufficiently consistent that we performed 10 iterations for each garbling percentage for each value of N. This allowed us to collect data for a wider range of values of N.

For a given query, our refined evaluation measure records the rank at which the correct song was found if the correct song has a non-zero similarity with the query. If the similarity is zero, this is noted. For each value of N, we report two measures at different garbling percentages: the average rank of the correct song if that song had a non-zero similarity with the query and the average recall, the percentage of non-zero similarity matches. These results are shown in Figure 5.

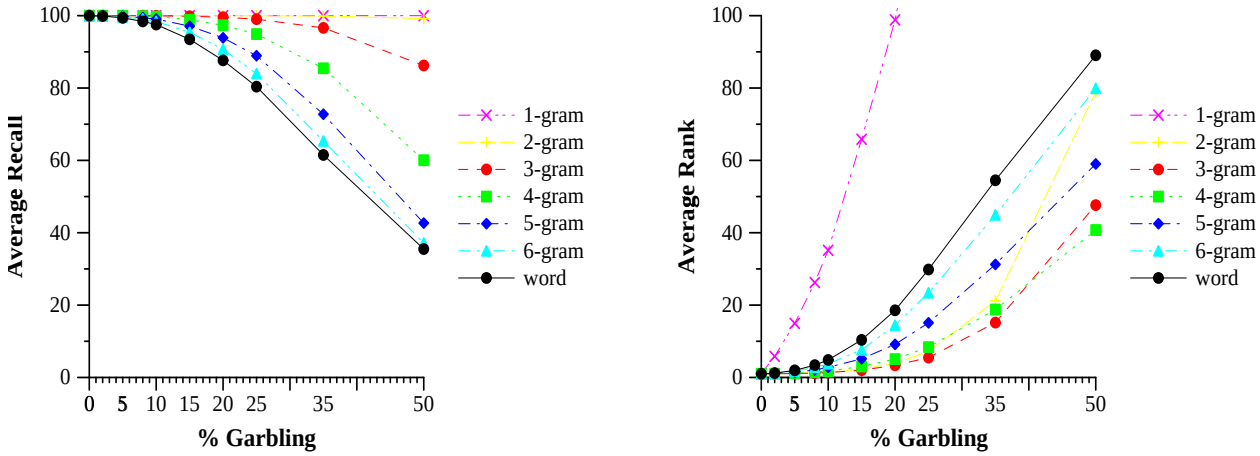

FIGURE 5: Average Recall and Average Rank for Garbled Queries

The revised measure allows a fuller characterization of the system's performance given progressively more garbled queries. Note that for garbling percentages up to 20%, the correct song is located on average over 80% of the time. In addition, when the song is located, it is returned in the top 20 songs of a ranked list (except for 1-grams).

3.3 Hand-transliteration

Finally, we examined how the system performed given hand-transliterated queries. Six Hindi speakers were asked to transliterate six song titles. The candidate song titles were presented to the volunteers written in the Devnagari alphabet. Example variant transliterations are shown in the first column of Figure 6.

In general, our system performed very well for hand-transliterated queries. All queries had a non-zero similarity with the target song and over all values of N the target song was returned at rank one 94% of the time. Results were very similar for N = 2, 3, 4, 5, 6 and the word-based approach.

In order to compare the results for transliterated queries to those for garbled queries we need to represent the hand-transliterated queries in terms of garbling percentage. For each of the 36 hand-transliterated queries, we calculated the edit distance between the hand-transliterated query and the song title in the database which that query should retrieve. The edit distance is the number of characters thatmust be inserted, deleted, substituted or transposed to turn *string1* into *string2*. Subsequently, we divided the edit distance by the length of the song title in the database and multiplied the resultant fraction by 100. This percentage is roughly analogous to the corresponding garbling percentage. If multiple queries differed from their target by the same percentage edit distance, their ranks were averaged. However, with only 36 queries, many of the values in Figure 7 represent one data point. Percentage edit distances for the example hand-transliterated queries are shown in the second column of Figure 6.

Variant Transliterations of: **sachchaai chhup nahin sakti banaavat ke asoolon se ke khushboo aa nahin sakti**	Percentage Edit Distance
sacchaye chup nahin sakti banavat ke usoolon se ki khusboo aa nahin sakti	10
sacchayee chup nahin sakti banaawat ke usoolon se ke khushboo aa nahin sakti	10
sacchayi chup nahin sakthi banaawat ke husoolon se ke khushboo aa nahin sakthi	10
sachchaayi chup nahin sakti banaavat ke usulon se ke khushboo aa nahin sakti	6
sachchai chup nahi sakti banaavat ke usulon se ke khushboo aa nahi sakti	9
sachaayi chhup nahin sakti banavat ke usoolon se ke khushboo aa nahi sakti	8

FIGURE 6: Example Hand-transliterated Queries

In Figure 7, we show the results for the hand transliterated queries in the same format as the results for the garbled queries. Each query had a non-zero similarity with the correct song. Since our recall was always 100%, the recall portion of the results is not graphed. The results are so similar that it is difficult to determine differences in Figure 7. For N = 2, 3, 4, the target song was always returned at rank 1. For N = 5, 6 and for the word-based approach, the target song was always returned in rank 1 to 15.

There was little performance difference for different values of N. In general, the word-based approach performed as well as N-grams for the hand-transliterated queries. Given that N-grams outperformed the word-based approach for the garbled queries, we examined the hand-transliterated queries to hypothesize about the cause. On examining the hand-transliterated queries, we discovered that differences were not spread uniformly across a query string. In some instances, transliteration differences were highly localized in a few words of the query, leaving enough overlap to allow the word-based approach to perform well. Generally, humans are more systematic in transliteration differences than a garbling function. The hand-transliterated queries showed different styles of representing certain phonemes. On average, queries were 11.5 words long and had an average overlap of 7.5 words with the desired song. The average Jaccard coefficient between a query and the correct song was 0.48. The Jaccard coefficient is $|X \cap Y| \div |X \cup Y|$ where X is the set of terms in the query and Y is the set of terms in the correct song. Typical queries are expected to have much fewer words than the entire song titles that we used for the hand-transliteration experiment. For typical queries, we expect word-based retrieval to fare poorly.

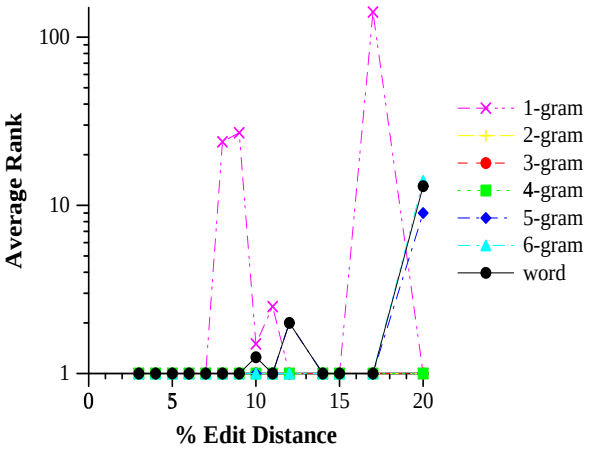

FIGURE 7: Average Rank for Hand-transliterated Queries

N-grams performed at least as well as the word-based approach for all queries and outperformed the word-based approach for queries with a larger transliteration difference.

3.4 Spell-checking

Traditionally, spell-checking has been performed using Hamming distances or edit distances between the misspelled word and candidate corrections [Kuk92]. Instead, spell-checking may be viewed as a retrieval operation wherein the misspelled word is the query and the dictionary is the document collection. Based on this view of spell-checking we constructed vectors for every word in /usr/dict/words that had only lowercase letters. For this application we used multiple N-grams of the same word within the vector representation for that word. For example, for a 7-letter word we generated the terms by combining all the 1-, 2-, 3-, 4-, 5-, 6- and 7-grams of that word. The weights for each term were merely the term frequencies, not the product of term frequency and inverse document frequency that we used for the earlier application. Given a misspelled word, the system returned the top few normalized similarity measures over all the words in the dictionary. As Figure 8 shows, our method, named `correct`, returned more options (ranked, moreover) for the misspelled word than a traditional system, in this case, `webster`. Traditional systems make the restricting assumption that the first letter of the misspelled word is also the first letter of the intended word. Our system does not make that assumption, and thus can furnish options missed by traditional systems.

Command: webster pecify
No definition for 'pecify'.
Maybe you mean:
1. pacify

Command: correct pecify

0.866	*specify*
0.587	*specific*
0.524	*pacify*
0.501	*specie*
0.438	*crucify*

FIGURE 8: Spell-checking

4 Analysis

In this section we present arguments influencing the choice of N for the N-gram technique.

4.1 Number of N-grams

In a word-based system using the vector space model, the number of terms in a document vector is the number of unique words in the document. In an N-gram system, the number of terms is the number of unique N-grams in the document. Thus the potential number of N-grams increases exponentially with N. Specifically, given alphabet Σ with cardinality $\aleph$, for a specific value of N = n, the number of potential N-grams using our N-gram extraction technique is $\aleph^n + \aleph^{n-1} + \ldots + \aleph^2 + \aleph^1$. However, a large number of these N-grams never occur in a realistic document. In Figure 9, we plot the number of unique N-grams in sample collections against different values of N. Notice that the number of unique N-grams is much less than the number of potential N-grams for large values of N. We determined the number of unique words for one collection. The number of N-grams increases exponentially for small N. For large N, the number of N-grams drops asymptotically towards the number of unique words in the collection. The N-gram curve peaks at N = 5, indicating the maximal number of unique N-grams. Interestingly, barring a few

exceptions, the N-gram curve peaks at N = 5 for most collections in different languages. However, the number of N-grams does not offer any intuition about the final choice of N for any application although it does suggest $N \leq 5$.

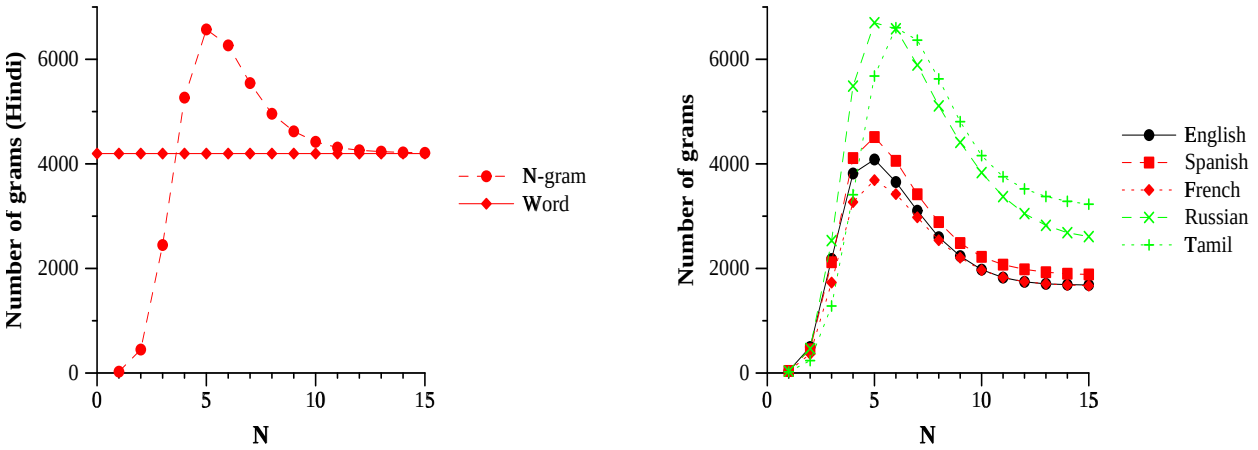

FIGURE 9: Number of N-grams in sample collections in Hindi and other languages

4.2 Choice of N

The choice of N made by the various systems in the literature is largely *ad hoc*. The STORES system suggested a a value of 3 for their polygrams because it yielded the best selectivity in search access rate [DeH74]. However, the results of the corresponding experiment are not shown. Other systems have used trigrams in order to conserve memory or disk accesses [Adams93]. Cavnar used bigrams and trigrams together in the same system because he believed that bigrams provided better matching for individual words while trigrams provided the connections between words to improve phrase matching, thus complementing each other [Cav94a]. Cohen [Cohen95] and Damashek [Dam95] used 5-grams, while Robertson and Willett [Rob92] used 2- and 3-grams; none gave reasons for the choice. Acquaintance used 5-grams initially and later changed to 4-grams to improve their results with 20% garbled text [Huff95, Huff96]. For Korean text retrieval, bigrams used in conjunction with N-grams provided the best 11-point average precision [Lee96].

Our experiments suggest that N = 3 is an acceptable compromise between values of N that result in high recall and values of N that return the desired song early in the list of ranked responses. Indeed, on an average, 3-grams result in the best ranks at acceptable recall rates. Other considerations that may influence the choice of N could be the size of the document vectors and the time taken to process a query. We found that the size in bytes of the document vectors increased for N = 1, 2 and then fell for N = 3, 4, 5 and 6 with the word-based vectors being the smallest. Query processing times fell as N was increased, with word-based queries being the fastest. Word-based queries took almost half as much time to process as queries based on 1-grams. The difference in speed is explicable in part to the reduced vector sizes, but mostly due to the fact that as N increased fewer songs returned non-zero similarity with the query. Therefore, the number of responses to sort and rank was reduced.

5 Conclusion

Our work examines some of the well-known uses of N-grams, such as in retrieval and spell-checking. The retrieval system performed over the Hindi database is novel. Word-based searches performed poorly when submitted garbled queries; N-gram searches retrieved documents fairly accurately despite garbled queries. N-gram techniques are language-independent. Therefore, they are well-suited for collections having documents in different languages or multi-lingual documents. Future work will address these kinds of collections.

Based on our completed studies, we recommend N-grams as a strong alternative to word-based search techniques when spelling variants are an issue.

6 References

[Adams93] Adams, E. S., Meltzer, A. C., *Trigrams as Index Elements in Full Text Retrieval Observations and Experimental Results*, ACM Computer Science Conference, February 1993.

[Cav94a] Cavnar, W. B., *N-gram-Based Text Filtering for TREC-2*, The Second Text REtrieval Conference (TREC-2), February 1994.

[Cav94b] Cavnar, W. B., Trenkle, J. M., *N-gram-Based Text Categorization*, Symposium on Document Analysis and Information Retrieval, April 1994.

[Cav95] Cavnar, W. B., *Using an N-gram-Based Document Representation with a Vector Processing Retrieval Model*, The Fourth Text REtrieval Conference (TREC-3), April 1995.

[Cohen95] Cohen, J. D., *Highlights: Language- and Domain-Independent Automatic Indexing Terms for Abstracting*, Journal of the American Society for Information Science, 46(3), 1995.

[DeH74] De Heer, T., *Experiments with Syntactic Traces in Information Retrieval*, Information Storage Retrieval, Volume 10, January 1974.

[Dam95] Damashek, M., *Gauging Similarity with n-grams: Language-Independent Categorization of Text*, Science, Volume 267, February 1995.

[Huff95] Huffman, S., Damashek, M., *Acquaintance: A Novel Vector-Space N-gram Technique for Document Categorization*, The Third Text REtrieval Conference (TREC-3), April 1995.

[Huff96] Huffman, S., *Acquaintance: Language-Independent Document Categorization by N-grams*, The Fourth Text REtrieval Conference (TREC-4), October 1996.

[Kuk92] Kukich, K., *Techniques for Automatically Correcting Words in Text*, Computing Surveys, 24(4):377-440, December 1992.

[Lee96] Lee, J. H., Ahn, J. S., *Using n-grams for Korean Text Retrieval*, 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1996.

[Rob92] Robertson, A. M., Willett, P., *Searching for Historical Word-Forms in a Database of 17th-Century English Text using Spelling-Correction Methods*, 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992.

[Salton75] Salton, G., Wong, A., Yang, C. S., *A Vector Space Model for Automatic Indexing*, Communications of the ACM, Volume 18, Number 11, November 1975.