Sir,

Following are the write ups for the projects I discussed with you :

**1) AI Assisted Buying Platform :**

Worked for the creation of a smarter version of Apple's Siri for E-Commerce, an AI (Artificial Intelligence) powered Automated messaging platform that can process the customer orders. It is basically a chat-based personal shopping assistant.

**Major technical works I contributed towards** : NLP(Natural Language Processing),Web Crawling, Data Scrapping, Dara Normalizing.

**Platform(or Toolkit) Used** : Stanford core NLP(Natural Language Processing), which provides a set of natural language analysis tools which can take raw text input and give the base forms of words, their parts of speech, etc.

P.S. : I have prepared and attached a pdf named : **Ashish\_NLP\_Text\_Normalizer.pdf** which contains the basic overview of what I did as a part of this project.

Following is what i did as a part of this project :

**TEXT NORMALIZER**

WORD

**SUBCOMPONENTS**

• Text Cleaner

• Slang Handler

• Check in Dictionary

• Check Noun

• Spell Correction

• Keyboard Filter

• Phonetic Spell Correction

• Keyboard Suggestions

• N-gram Model (Backward and Forward)

**SUBCOMPONENTS IN DETAIL (STEP-WISE)**

**TEXT CLEANER**

Input is taken in the form of a sentence. Extra spaces, punctuations, special

characters are handled. If some letters are present 3 or more times in

succession then these are reduced to 2. While handling special characters,

email ids and URLs are kept as such.

**SLANG HANDLER**

After cleaning the text of unwanted characters, spaces and punctuations,

we split the sentence into words. Slang List has been stored in a HashMap

and hence each word is searched in the HashMap in O(1) time and we

replace the slang by its meaning stored in the HashMap.

In this step it is possible that slang is associated with more than one

meaning. To handle such slangs, before replacement, we do a contextbased

search using a 3-gram model. Hence the most probable meaning

based on the context is selected and is used as substitution for the slang.

Also consider the case when the user types Pay tm. Actually the user

meant the brand name Paytm. But our slang handler would return

trademark for tm treating it as slang. To handle such anomalies, before

look up in the slang list, we search for such possible combinations of

formation of noun by first looking up the previous word, if we don't get a

combination for noun we look for the combination with the next word. If

again we don't get a noun we look for the combination of the three words

(previous, current and next words). If we get a noun in any of the

combinations we leave it as it is, else we go for slang look up and

subsequent replacement.

**SEARCH DICTIONARY**

In the next step we search for the word in the English dictionary stored in a

HashMap. If the word is present in the dictionary we just return the word as

it is and proceed to the next word. If word is not present in the dictionary we

do the following steps.

**CHECK FOR NOUN**

If the word is present in the noun list stored as a HashMap, then we return

the noun as it is and we proceed to the next word. Just like we handled the

case of Pay tm in slang handler, similarly before proceeding further we do

the same checking at this step also because it is possible that we may not

have a slang with us and hence it might not get detected in the slang

handler.

If no noun or such combination of words, to form a noun, is found, we move

to subsequent steps.

**SPELL CORRECTION**

In this step we return a list of plausible corrected word suggestions for the

misspelt word given by the user as input. For Spell Correction we use

Symmetric Spell Check Algorithm. This algorithm handles all the cases like

deletes, inserts, substitutions and transposes by only considering deletes.

This step requires pre-processing of the corpus data. Once this preprocessing

is done, this algorithm runs very fast and returns a list of

suggestions up to a certain edit-distance (say 2).

For more details for the algorithm go to the following link: -

http://blog.faroo.com/2012/06/07/improved-edit-distance-based-spellingcorrection/

**KEYBOARD FILTER**

The list of words generated from the previous step is a very exhaustive list

and contains many redundant suggestions. Eg: For tre as input we will get

care/bare as also some suggestions, which are redundant. Hence in this

part we keep a keyboard proximity maintained in the form of a HashMap

and based on this proximity we check for various words by performing

inserts, swaps, deletes on them and redundant words are filtered.

**PHONETIC SPELL CORRECTOR**

Say user gives in the input as “bcoz”. It is quite possible that such words

are not present in the slang list. Clearly the user means “because”. But we

won’t be getting it in suggestions from Spell Correction because the edit

distance is quite high. Now for such cases we generate a list of words that

sound similar to the input word. For this part we use Soundex Algorithm

and Metaphone algorithm. Again these algorithms return some redundant

suggestions. For getting the best possible suggestions, we run both these

algorithms separately and then the common suggestions are used.

http://en.wikipedia.org/wiki/Soundex

http://en.wikipedia.org/wiki/Metaphone

**KEYBOARD SUGGESTOR**

Consider a worst possible scenario when user makes typing mistake in

almost all the letters of a word.

Eg: User types “YGSR” but actually he/she meant something else. Now

again in this scenario we won’t get correct suggestions from the spell

checker because the edit distance is too high. Hence for handling such

anomalies we keep proximity of each letter in a HashMap and find all the

possible combinations based on this proximity. Then the combination,

which forms some meaningful word, is returned. Eg: In case of “YGSR” we

get “THAT” as output.

**CONTEXT BASED SPELL CHECK**

All the suggestions obtained from spell correction (subsequent keyboard

filter), phonetic algorithm and keyboard suggestions are then sent for

Context Based Spell Correction. In this we check for context both

backwards and forward. We use a 3-gram model. The conditional

probabilities for each suggestion are calculated and the suggestion with the

maximum probability is returned as the final replacement for the misspelt

word.