

Natural Language Processing
Prof. Pawan Goyal
Department of Computer Science and Engineering
Indian Institute Technology, Kharagpur

Lecture - 07
Weighted Edit Distance, Other Variations

Hello everyone and welcome to the second lecture of second week. So in the last lecture we talked about what is the basic edit distance concept given 2 strings as input, what is a dynamic programming algorithm that you can use find out, what is the edit distance between these 2 strings. And remember we defined certain operations, certain edit operations and we gave weights or cost for each of these operations. But all these cost for equal. If you are substituting a character by another character irrespective what are the 2 characters? The cost remains the same. Same goes for inserting a character. If the cost remains same and also for deletion of a character it remains same so in this lecture we will briefly discuss that is there any way in which we can try to discriminate between various sorts of edit operations.

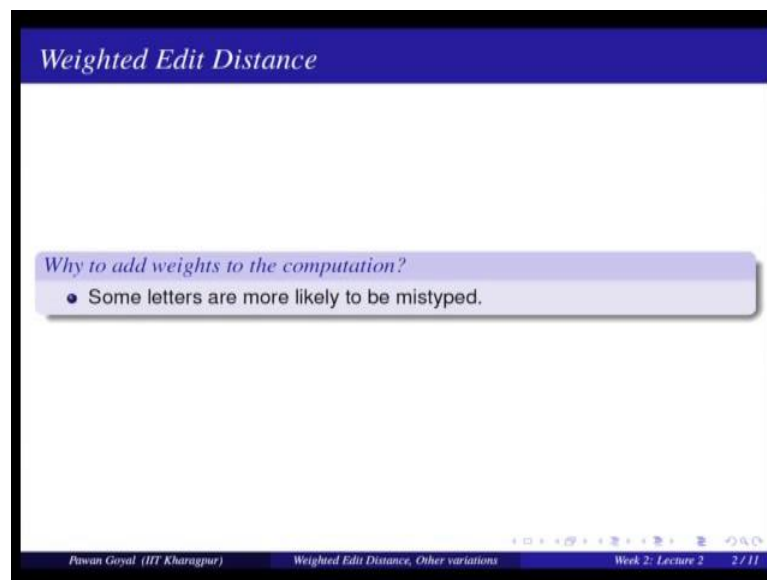
Firstly is this, should we discriminate between various operation, should there we assigned the same cost or should be try to give different cost and different operations. Then we will also talk about the practical problem that given a query, Query I mean a term that might be incorrectly spelt, how do I come up with list of terms that are actual, that for actual terms that should have been used instead of the incorrect so what should be, what is a good or efficient algorithm to find out all the set of terms that are variation of edit distance of say edit distance of 1 or 2 of that query.

So starting with the concept of weighted edit distance so can we weight the edit distance cost depending on what are the characters that are involved in the operation. So firstly, is it required? So, do you think substituting a by e should have the same cost substituting a by m so what should depend on? What kind of; so for example, there are certain errors that you make very commonly and certain errors that you make very rarely. Suppose error is very common should I assigned it a higher cost or a lower cost? So think about it. What is it mean to an assigning it a higher cost? Assigning it a higher cost means that the edit distance between the 2 strings will increase if that edit operation is present. So that means, the probability of obtaining the other string as a candidate we will go down if the

operation as a high cost, but if something is some sort of edit operations are very common, some spelling mistakes are very common we would like to give them a lower cost so that I can the actual candidate have a small edit distance with the incorrect word

So, it is important to understand what kind of characters will have a lower edit distance and what kind of character will have a higher edit distance. Can be somehow give a weight. So this is the basic idea.

(Refer Slide Time: 03:52)



The slide is titled "Weighted Edit Distance" in a blue header. Below the title, there is a light blue box containing the text "Why to add weights to the computation?". Underneath this box, a bullet point states "Some letters are more likely to be mistyped." At the bottom of the slide, there is a footer with the text "Pravun Goyal (IIT Kharagpur)", "Weighted Edit Distance, Other variations", "Week 2: Lecture 2", and "2 / 11".

So, why do we need to add various weights to the computation? So the idea is that some letters in a language are more likely to be mistyped than others.

(Refer Slide Time: 04:10)

Confusion Matrix for Spelling Errors

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X \ Y	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	9	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	2	0	6	1	0	7	36	8	5	0	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Navigation icons

Pawan Goyal (IIT Kharagpur) Weighted Edit Distance, Other variations Week 2: Lecture 2 3 / 11

So, let us take some statistics. So this is a confusion matrix for spelling errors that was found in a corpus. So x is the incorrect character and y is the correct character. So what is this stability is going. How often in that corpus they found an incorrect x for a correct y. So for example, if you try to read the table, the entry for the first row, the anti-corresponding to a e, that means, how many times instead of the correct e and incorrect a bar substituted. And that number comes up to be 342. So that means in the corpus 340 times, at an instead of e a was written. Similarly, 118 times instead of i, a was written. Similarly, 76 times instead of o, a was written

On the other hand, if you see b, it never happened that a b was written as a. So that means, people make the mistake of converting or instead of writing e writing a, this is a very common mistake on the other hand b to a is not very common. So you can look at this confusion matrix and you will find a lot of nice patterns. So one thing you will see is that the errors among the vowels are very common. So substituting i for e, e for I, a for o they are very common. So there may be because the person is may be not knowing the spelling or so is even if you know the spelling by mistake is typing the other verbal. So this is also possible.

Now, there are some other kinds of errors that you can see here that come because of the way keyboard also designed. So you will see the characters that that come very close in the keyboard, so sometimes you miss type them. So this is again a common source of a

spelling error. So for example, m and n are very close together, and they also sound very similar. So it is very likely that you mistake m for n, or n for m. So that is this kind of errors, are very common. So some errors come because some of the characters' sound very similar, so like the all the vowels or certain characters. And also some characters are very close in keyboards. So that is again another source of errors.

So, now once we know that some errors are more likely than others, can I use that this statistic to design my weighting scheme. So as I said earlier if some error is more likely, the addit cost from one character to another character should be low so that I can easily find what should be the actual candidate given in the erroneous word. So the smaller edit distance means that is more likely to be the correct candidate. That is why we will give a smaller cost to those spelling mistakes that more likely and in larger cost or a higher cost those spelling mistakes that are not so like, that are very rare.

(Refer Slide Time: 07:36)



So, keyboard design is again one thing that gives rise to many of the errors that we have seen in the previous slide. So again try to correlate among the characters that are coming very close in the keyboard. So we will see that many a times people make mistakes in typing one character for another.

(Refer Slide Time: 07:59)

Weighted Minimum Edit Distance

Initialization:

$$D(0,0) = 0$$
$$D(i,0) = D(i-1,0) + \text{del}[x(i)]; \quad 1 < i \leq N$$
$$D(0,j) = D(0,j-1) + \text{ins}[y(j)]; \quad 1 < j \leq M$$

Recurrence Relation:

$$D(i,j) = \min \begin{cases} D(i-1,j) + \text{del}[x(i)] \\ D(i,j-1) + \text{ins}[y(j)] \\ D(i-1,j-1) + \text{sub}[x(i),y(j)] \end{cases}$$

Termination:

$D(N,M)$ is distance

Pawan Goyal (IIT Kharagpur) Weighted Edit Distance, Other variations Week 2: Lecture 2 5/11

So, now suppose you have some data and by analyzing the data you can find out what kind of errors are common, what kind of errors are rare. And accordingly you can design your weights for different edit operations. So now, once you have each weight that depends on the actual characters you can modify your initial algorithm. So how can you modify your algorithm? So for example, here so I will, so if you if you try to compare this with an algorithm that we saw in the previous slide, small reference here is that we should have giving a uniform cost for deletion insertion substitution.

What we are doing here so deletion is now conditioned on the actual character that is been deleted. So you see $D(i,0)$ is $D(i-1,0)$ plus a cost for deletion of $x(i)$. So what is the cost for deleting that particular character? Same goes for insertion. What is the cost for inserting that particular character? Even in substitution you might have cost for so that separate cost for deletion insertion and substitution of one character by another. And everything else remains the same the only thing that changes is that instead of using equivalent cost for each operation you are now giving cost that are dependent on the actual characters that are inserted or substituted or deleted. So that is my weighted edit distance.

(Refer Slide Time: 09:39)

How to modify the algorithm with transpose?

Transpose

- $\text{transpose}(x, y) = (y, x)$
- Also known as metathesis

Pawan Goyal (IIT Kharagpur) Weighted Edit Distance, Other variations Week 2: Lecture 2 6/11

So, now so what to do you do with, so how do you modify your algorithm for taking care of transpose for instance. So that is the algorithm were we had 3 operations. We had insertion, substitution and deletion. They were 3 operations that we why we are using in that previous operation. So there is another very common edit distance operation that is used and this is called transpose. So if I am transposing toward 2 characters. And you will receive this is a common error that sometimes you make.

(Refer Slide Time: 10:19)

$al \leftrightarrow la$

transpose (metathesis)

$\begin{matrix} x & y \\ \hline x_i & y_j \end{matrix} \rightarrow \begin{matrix} y & x \\ \hline y_j & x_i \end{matrix} \rightarrow 1 \text{ for transposition}$

$$D(i, j) = \min \begin{cases} D(i-2, j-2) + 1 & (\text{transpose}) \\ \text{if } X[i-1] = Y[j] \\ \text{or } X[i] = Y[j-1] \\ \text{insertion} \\ \text{deletion} \\ \text{substitution} \end{cases}$$

© CSE
IIT KGP

So, for example, so I am wanting to type al, and instead I typed it as la. So this is called

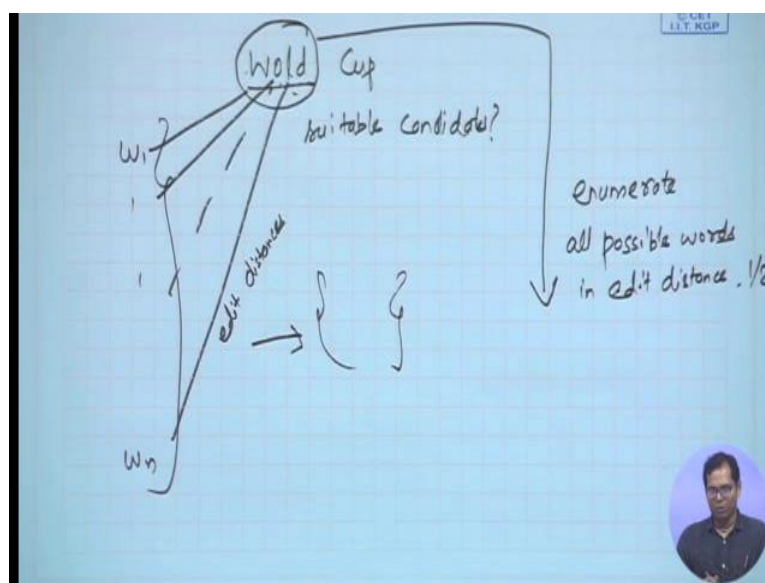
transpose. So in general if I write xy as yx . So this is called transpose another name for this is metathesis. So now, how can I modify my earlier algorithm, to take care of this case? So till now we have 3 operations insertion deletion substitution. In insertion I see the previous insertion deletion I see only till the previous word, but what do I do in the case of transposition. So in the transposition I will have to so because it is we have go to characters before so I need to go to D in my cost matrix $D[i-2][j-2]$. And suppose I give a cost of one for transposition.

So, I will write $D[i][j]$ as so there are other cost that we had defined earlier, but for transposition I will say $D[i-2][j-2] + 1$ and what will be the condition - only if the previous word in x , so I had defined my strings as x and y . Here so this is my string as x this is my string as y . So what will happen here? This is up to i so this $x[i-1]$ will be same as the j th character in y . So if $x[i-1]$ is $y[j]$ and $x[i]$ is equal to $y[j-1]$. So if I find that so x is still i is and y it is will j . So if I find that the i minus one character of x is same as the j th character of y , and the i th character of the x is same as the j minus one character of y , then I will say that this cost is $D[i-2][j-2] + 1$.

So, again I will do a minimum of this. This if this happens this is my transpose and then they I have other conditions for insertion deletion and substitution. And that is how I can modify my algorithm by initial algorithm to also include transposition. So with all these operation this is the only other operation that is also commonly used in computing edit distance. So now, so we talk about edit distance and also about doing including another edit operation like transpose.

So as I said earlier so we will also discuss briefly, so what is the practical scenario. So the practical scenario here is you are given an input word. So remember in the introduction slide we 2 are talking about this error.

(Refer Slide Time: 14:17)



Say wold cup and by mistake I have written wold instead of writing world. So now, the practical problem here is once I know that this is not the correct word in my vocabulary, how do I find out what are the possible candidates that might come in place of w o l d. So that is what are the suitable candidates. Now as per our definition so the suitable candidates are those that are having a small edit distance from the actual error word. So intern my problem is how go I find out words that are within some a small error some small edit distance of word.

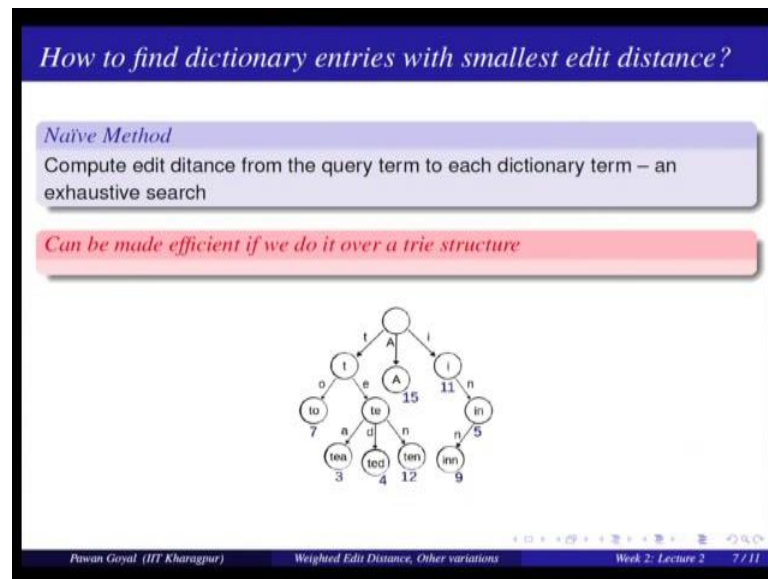
So, now how do I solve this problem; given a word like wold find out all the words that are within the additions of say 1 or 2. Now one simple thing you might say that I will list down all the words by vocabulary somewhere, so I have all the words in my vocabulary starting from w_1 to w_n and I will find out what is edit distance of w_1 with world, w_2 with world, and up to w_n . I will find out all the edit distances and then I will among those I will choose all those that are coming up to the very small edit distance some top entries that are coming out to be within a small edit distance. So I will find out this type of entries. This is my candidate.

But you can immediately see that this may not be a very efficient solution because my vocabulary size can be quite huge and I have to find out edit distance from this to each of the words in my vocabulary. So that will be really time consuming, so instead can I so can I do something more efficient. So one simple thing is I should try to search in an

efficient data structure for searching all the words in my dictionary.

So in case you do not know so one of the very efficient edit structures that is used for searching the strings is called the trie structure. So using the trie structure you can do search efficiently in the linear time, so linear in the length of the input string.

(Refer Slide Time: 16:53)



So, I am giving you one example here. So I can use the trie structure. So idea is that so you take any word you can start from the initial root node of this trie structure and keep on following the arcs and in the time linear to the length of your word you will reach to a node and find out if the word is available in the vocabulary or not. So by that you will be able to search all the words in the vocabulary. And this trie structure can be used efficiently for computing the distance of any new word with a word in this trie. So, this is one possibility.

But this again it requires you to do a lot of computations with respect to all the words in your vocabulary. What can be other possibility? So other possibility if your thing would be so instead of trying out to find out the edit distance of this error word to all the words in the dictionary, can I start from my error word and try to enumerate all possible words within some edit distance, say 1 or 2 I am trying to enumerate all the possible words edit distance 1 or 2. So I will find out the word like world and everything else and then I can have a further check to see if these words there in my dictionary and all, and there I can use maybe a trie structure, that whether these words are available in my dictionary or not.

So, this can be the possibility. Now what is, do you see do think this will be efficient this approach. So let us see some numbers.

(Refer Slide Time: 18:57)

How to find dictionary entries with smallest edit distance?

- Generate all possible terms with an edit distance ≤ 2 (deletion + transpose + substitution + insertion) from the query term and search them in the dictionary.
- For a word of length 9, alphabet of size 36, this will lead to 114,324 terms to search for
- For Chinese alphabet size is 70,000 (Unicode Han Characters)

Pawan Goyal (IIT Kharagpur) Weighted Edit Distance, Other variations Week 2: Lecture 2

So, yes so I can possible generate all possible terms is starting from the error word that are within edit distance of less than equal to 2. And there I can take care of all the deletion substitution addition and transposition. I can try to take care of all the possible edit operations. So starting where would I generate all the possibilities with the edit distance of 2.

Now, so if you just try to do it is a simple math, try to compute how many different possibilities you will have to explore. So let us see. So suppose if you take a word of length 9 that is 9 characters and suppose in you are taking a language like English and alphabet is 36 and so that 26 plus other characters. And if you just do, that so this will give you roughly 114,324 different possibilities to explore starting from the word of length 9 in which of the 4 you think will give you a lot of possibilities. Deletion will probably not give you many possibilities, but substitution will give you lot of possibilities and insertion should also give you lot of possibilities.

Substitution you can substitute each character by and you have 36 characters. So that gives you a huge number of possibilities. So this is again huge number. So it is starting from a word of length 9 you are going to explore 114,324 different possibilities, but English is still ok. So you have 36 characters. So think about a language like Chinese

where the alphabets size is 70,000. So you cannot even think of applying an approach like that.

So, this approach is slightly better than then before, but it is may not be very efficient again, depending on the alphabet size. So what can be the other possibilities? So other possible algorithm is that the idea that all this edit distances can be attained if you try to do a symmetric delete from the current word and the possible candidate words, so all the edit distance of operation 2 can be found out just by doing deletes in both the words.

(Refer Slide Time: 21:26)

How to find dictionary entries with smallest edit distance?

Symmetric Delete Spelling Correction

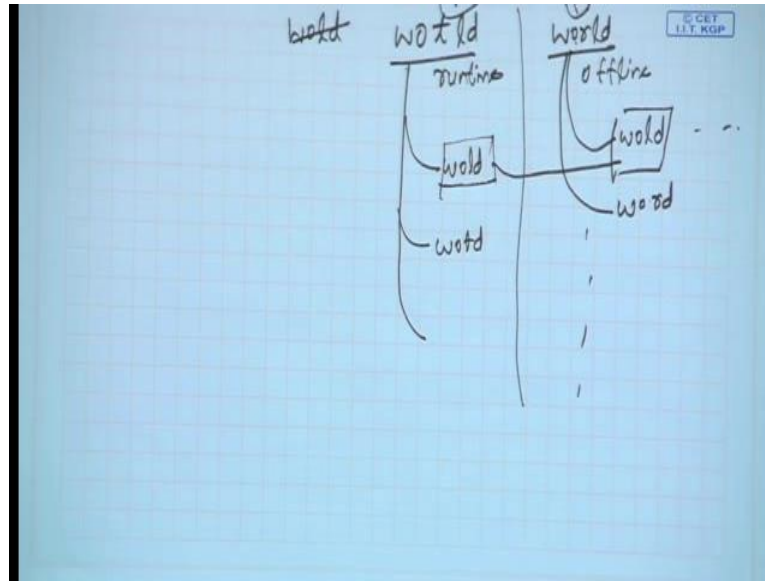
- Generate terms with an edit distance ≤ 2 (deletes) from each dictionary term (offline)
- Generate terms with an edit distance ≤ 2 (deletes) from the input terms and search in dictionary

Pawan Goyal (IIT Khargpur) Weighted Edit Distance, Other variations Week 2: Lecture 2 9/11

So, what is the idea? So this is called the symmetric delete a spelling correction. So idea is that from your query word and so from your query word you will try to find out all the words that come up within delete of 2. From the query word and from your dictionary for each of the word you will try to find out all possibility that come within and edit distance 2 as per only the deletes. And that this will cover all the possible variations of edit distance of 2.

So, you can we can take a simple example. So let us take the same example of wold or so even or we can try to take a different example.

(Refer Slide Time: 22:05)



Say wotld instead of world that is the correct word and this is the error word. So what we will do is symmetric delete operation. So this is this I will do offline. It is an offline a starting from word I will store all the possibilities within edit distance of 1 and 2. So I will store with world that there is word wold that comes up word and so on and from these again you will have this is delete 1 and you will also have again either candidates for delete 2. At run time when you get this query like wotld that is incorrect, you will do the same thing. So this is at run time and you will again generate all the possibilities with delete of 1 and 2.

So, we are doing here one only to show the possible. So you will find wold, wotd and so on. And now you will try to match if one of this is available in this dictionary that is built by this symmetric delete operation. And you will find wold and wold is a match. And once this is a match you will go back to the original word, and then confirm that this word and this word have an edit distance of less than or equal to 2. And that is how you can find out all the possible words within edit distance of 2, without I am doing a lot at run time. Now can you tell why; because you only doing deletes at run time and deletes a not mind.

So, if you do one delete if you heard of 9 characters you have only 9 possible deletes. So this is not very inefficient at run time and this all is done at offline. And later you will have a further check to see that these 2 words have additions of 2. If your 2 or 3 sort. So

yeah if you take number of deletes of 2 for a word of length 9 it will be at most 45. So this is not a very big number. So this is one approach that you can use for finding out all the possible dictionary terms within some edit distance efficiently at run time.

(Refer Slide Time: 24:55)

Non-word spelling errors

Non-word spelling error detection

- Any word not in a dictionary is an error
- The larger the dictionary the better

Non-word spelling error correction

- Generate candidates: real words that are similar to the error word
- Choose the best one:
 - Shortest weighted edit distance
 - Highest noisy channel probability

Pawan Goyal (IIT Khargpur) Weighted Edit Distance, Other variations Week 2: Lecture 2 11/11

So, now so when we talk about edit distance and some variations. So we will not focus on the task of spelling correction with which we had a started this module on edit distance and all. So when we talk about spelling errors there are 2 kind of errors that you might have to deal with. So one that are easy to steep to handle errors, so I will say slightly easier they are called non word errors because they are easy to detect. So the erroneous word is not in my vocabulary.

So in this case so erroneous word is b e h a f. That is not in my vocabulary I can deduct decision erroneous word and my task is to correct it, to the actual word b e h a l f, behalf and this is also called non word error. What is slightly more difficult is something called a real word error. When the word that is in error is also in my dictionary, but somehow the user has missed spelt to a word that is in my dictionary so here some examples. So like typographical error so instead of writing there somebody has write 3, now 3 in my vocabulary so it is not easy to that this is in error, now correcting this is called a real word error correction.

Similarly, they can be something for because of the homophones, because of say piece and peace the 2 different variations of pieces you might misspell very easily. And too and

two is very common error. So they are also called real word errors. Now to handle both of the non-word error and real word error, you need to use various probabilistic models. And this is what we will be focusing. So in non-word spelling errors so the non-word means the word that is not in my edit dictionary, so any word that is not in the dictionary is called an error. And so for handling this problem it is better that you have a good very good lexicon tells you all the possible word in your lexicon. And this lexicon may also be dependent on the particular corpus that you are processing so, for you if you are processing scientific corpus, you might have different corpus, then if you are processing use corpus. So you should know what your words in the in the dictionary and when you encounter word that is not in your dictionary you will treated as an errors and try to correct it.

So in general what you will do in non-word spelling correction? So starting from the error word you will try to find out candidates that are similar to the error word, and how will you define the similarity with respect to the error word. So you will say the words that are within form a small at a distance. That can be one possible criteria and if you are using noisy channel model, that will be the topic for the next module, so you will see the one that is coming out to be as per the highest noise and probability. So this will be your candidate words. So you will choose some of the candidate words like that.

(Refer Slide Time: 28:12)

Real word spelling errors

For each word w , generate candidate set

- Find candidate words with similar pronunciations
- Find candidate words with similar spelling
- Include w in candidate set

Choosing best candidate

- Noisy Channel

Pawan Goyal (IIT Kharagpur) Weighted Edit Distance, Other variations Week 2: Lecture 2 12/11

On the other hand, if you are dealing with the real word spelling error, you will try to

find out so now you have the; you did not know in the sentence which of the word is actually erroneous. So what you will do for each word you will find out other words that are similar in edit distance or some other criteria, but in the candidate you will also include your actual word. So two is there so we will include too also, with two and you say that the actual sentence might contain any of these 2 words. And then finally, you will try to maximize the probability of the sentence using some noisy channel model. So this is what we will be discussing in the next lecture, where we will talk about noisy channel model in detail and how it can be used for doing the task of spelling correction.

Thank you.