VIRTUAL ASSISTANT FOR COLLEGE

Guide: Kavitha K V
Assistant Professor
CSE

Submitted By: Athira S Pillai(SCT15CS015)
Disha Dinesh Majithia(SCT15CS021)

CONTENTS

INTRODUCTION

- Chatbots are currently used to answer financial queries, provide customer support, diagnose healthcare issues, and even offer counseling. They’re already starting to make an impact on education and corporate learning.
- Before starting to work on such services, the fundamental points to be kept in mind are:
  - Domain Knowledge - What does a user expect this bot to understand?
  - Personality - What tone or vocabulary does the bot employ?

DOMAIN: College related queries PERSONALITY: Assistant bot

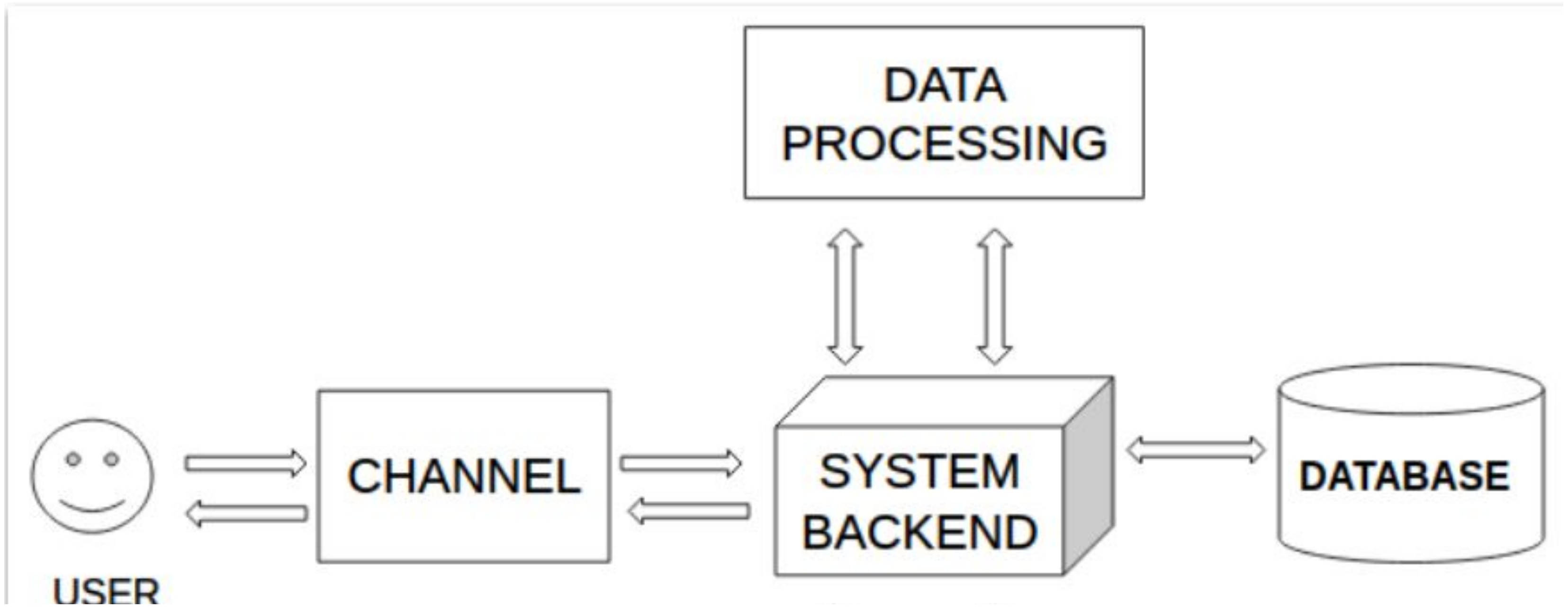
NEED FOR PROJECT

- Lack of information to students who reside outside the city regarding the college
- Lack of proper communication system to clear doubts and queries
- Need to read through the entire website content to get the required information
- Lack of knowledge regarding in campus programs and things.

PROBLEM STATEMENT

To create a virtual assistant to guide students who wish to take admission in the college.

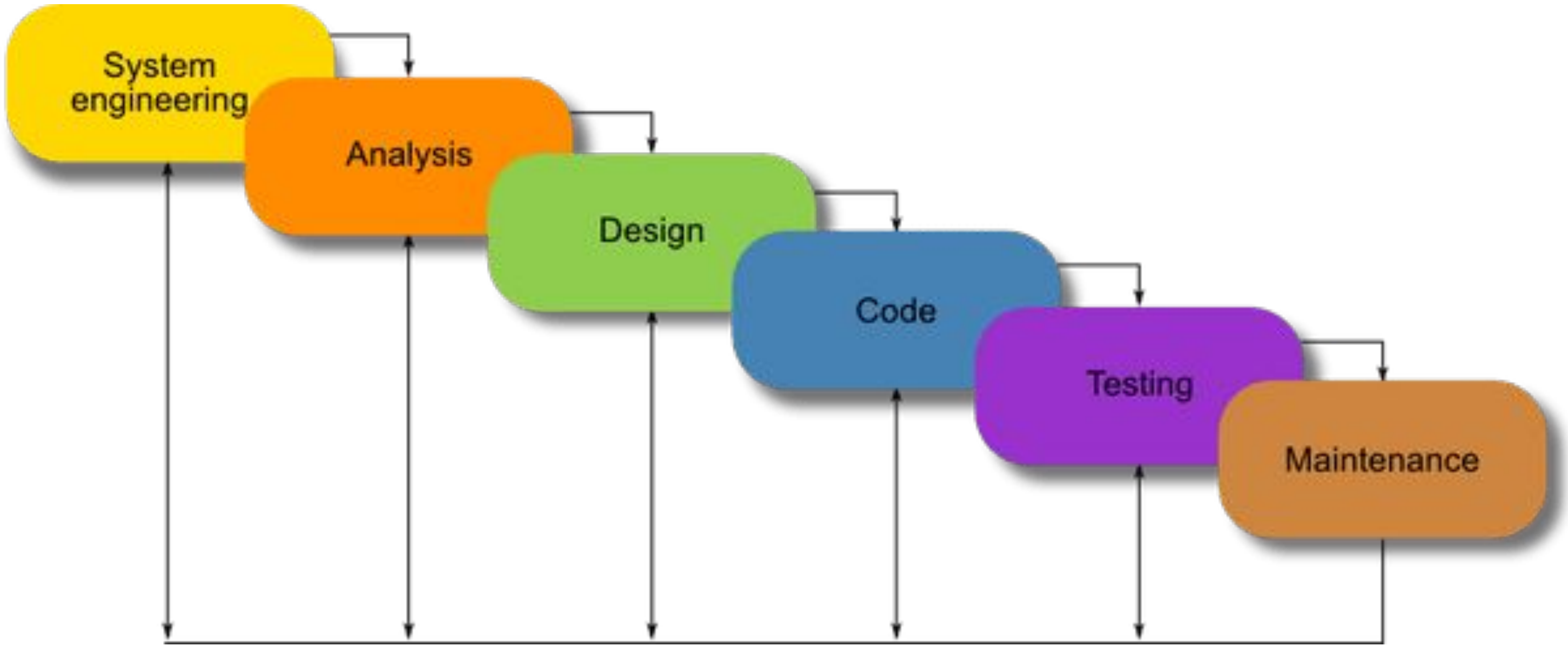
ARCHITECTURE



METHODOLOGIES

Waterfall Model

- Each phase of the model has specific deliverables and a review process
- Phases are completed one at a time
- Works well for projects where requirements are very well understood
- Clearly defined stages
- Easy documentation
- Technology can be well understood


System engineering
Analysis
Design
Code
Testing
Maintenance

DEVELOPMENT CYCLE

<u>Requirement Analysis Phase</u>

1) Survey carried out with in which students from different departments of different years were asked about the kind of questions they had in mind before joining the college.
2) Manipal University project was roughly studied.
3) For data collection, a Google forms link was distributed amongst all years asking them to fill the possible query questions.

<u>Design Phase</u>

1) Architecture of the bot was constructed and the required components decided upon
2) Required resources/softwares chosen as per the requirements from the previous phase
3) Data model representation chosen

<u>Implementation Phase</u>

1) Tasks assigned to both group members and strategy of work devised
2) Application implementation carried out in parts

SENTENCE SIMILARITY ALGORITHM

Semantic similarity between two sentences is carried out as below,

- Each sentence is divided into a list of tokens
- The divided sentence can implement bigram set
- After getting the divided sentence, it is counted
- Choose the pattern with the highest sentence similarity measure

$$\frac{\varsigma(s_1 \in s_2) \cup \varsigma(s_2 \in s_1)}{\varsigma(s_1) \cup \varsigma(s_2)} \tag{8}$$

The symbol of ς is count of the sentence or string that is symbolized as s. The used method to compute the semantic similarity between two sentences could be written as bellow,

a. Each sentence is divided into a list of tokens
b. The divided sentence can implement bigram set [11]
c. After getting the divided sentence, it is counted and applied the equation (8)

For example this method can be written as bellow,

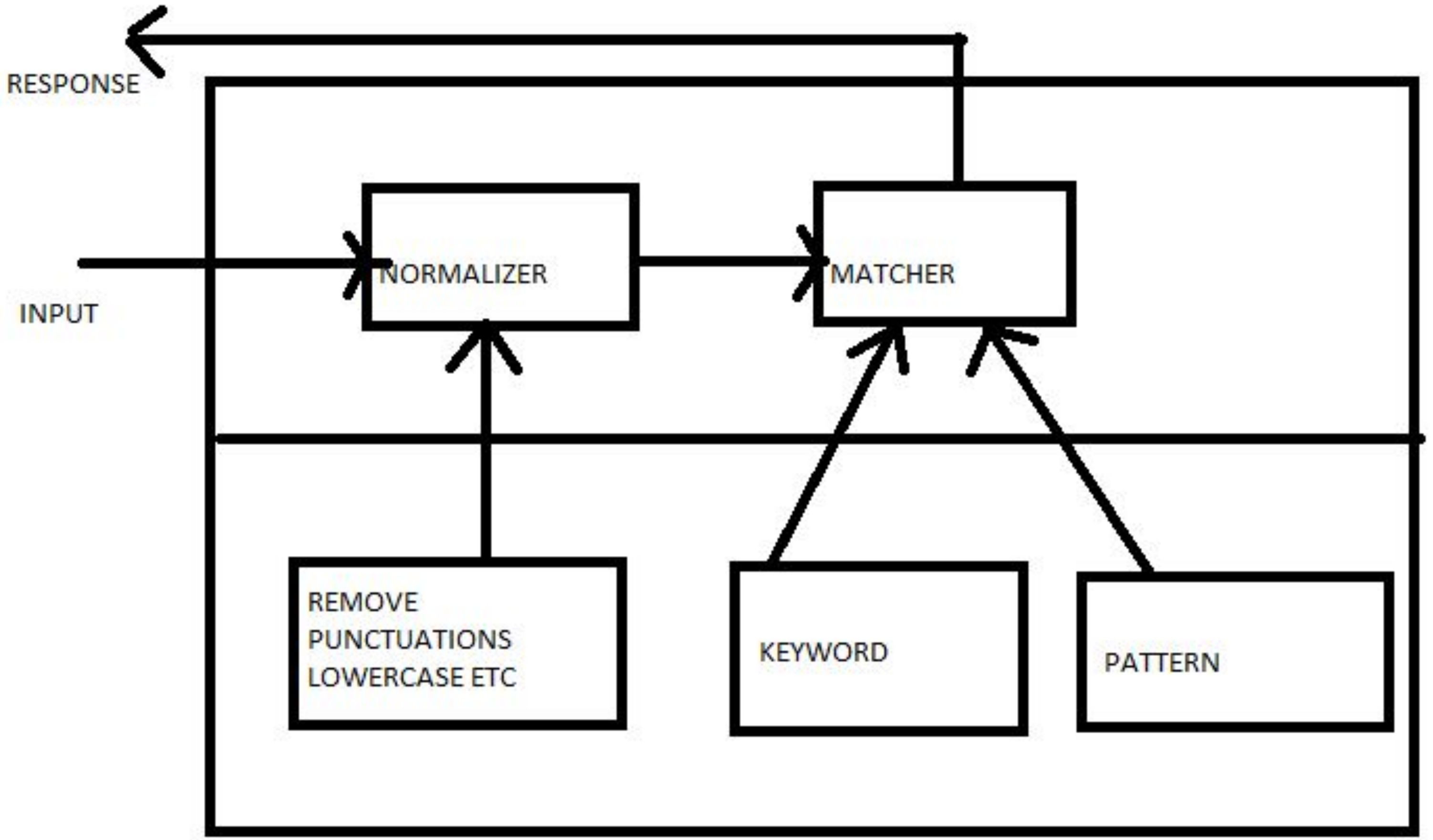
s_1 = “burung”
s_2 = “burrungg”

so s1 and s2 can be divided into bigram set as

s_1 = {“bu”, “ur”, “ru”, “un”, “ng”} ≈ 5
s_2 = {“bu”, “ur”, “rr”, “ru”, “un”, “ng”, “gg”} ≈ 7
$s_1 \in s_2$ = {“bu”, “ur”, “ru”, “un”, “ng”} ≈ 5
$s_2 \in s_1$ = {“bu”, “ur”, “ru”, “un”, “ng”} ≈ 5

Thus the similarity score for these sentences is obtained as follow,

$$\frac{5 \cup 5}{5 \cup 7} \approx \frac{10}{12} \approx 0{,}83333$$


RESPONSE
INPUT
NORMALIZER
MATCHER
REMOVE PUNCTUATIONS LOWERCASE ETC
KEYWORD
PATTERN

CHATBOT TRAINER

Chatbot works on data used to train the bot. It does the following on the data:

- Data is classified into words,classes,documents,ignore_words
- Create intents based on the data provided in json files
- Parse through the intents
- Tokenization
- Stemming
- Convert to lower case
- POS tagging
- Bag of words found

- Keywords found
- Neural network formed with desired number of units per layer

CHATBOT ENGINE

Chatbot engine performs the following on the user input :

- Stemming
- Tokenization
- Convert to all alphabets to lower case
- Forms bag of words
- Classifies the words in bag of words into classes in parts of speech
- Words not falling into class are deleted
- Response function - gives the response

We use NLTK (natural language toolkit) for 2 things:

- T**okenization**
- **Stemming**

There are various stemmers that we can use, for today we’ll use the ***Snowball stemmer.***

Each word from the given sentence is tokenized, stemmed, and converted to lowercase, this is consistent with the transforms we applied to the corpus data. In other words: *our input data is transformed consistently with our training data*.

STEMMING

- Stemming is the process of producing morphological variants of a root/base word.
- Stemming programs are commonly referred to as stemming algorithms or stemmers.
- Some more example of stemming for root word "like" include:
  - Likes
  - Liked
  - Likely
  - Liking

TOKENIZATION

- **Token** – Each “entity” that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is “tokenized” into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph.
- Eg : “***I am very happy.***” will be tokenized as ‘I’, ‘am’, ‘very’, ‘happy’, ‘.’

- Example of applying tokenization and stemming to an input sentence

  “Programers program with programing languages.”

Output :

‘Program’, ‘Program’, ‘With’, ‘Program’, ‘Lang’, ‘.’

POS TAGGING (POST)

- **Token :** Each “entity” that is a part of whatever was split up based on rules.
- Here’s a list of the tags, what they mean, and some examples:
  - *VBZ verb, 3rd person singular present - takes*
  - *VBP verb, sing. Present - take*
  - *PRP personal pronoun - I, he, she*
  - *NN noun, singular - desk*
  - *VB verb, base form - take*
  - *VBD verb, past tense - took etc*

BAG OF WORDS

- A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling,
- The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.
- A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:
    1. A vocabulary of known words.
    2. A measure of the presence of known words.
- The model is only concerned with whether known words occur in the document, not where in the document.

STOP WORDS

- In natural language processing, useless words (data), are referred to as ***stop words***.
- For this, we can remove them easily, by storing a list of words that you consider to be stop words (such as “the”, “a”, “an”, “in”).
- NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.
- Example :
  "This is a sample sentence, showing off the stop words filtration."
  Output

  ```
  'This', 'sample', 'sentence', ',', 'showing', 'stop','words', 'filtration', '.'
  ```

User Interface

- Chatbot to be embedded onto the website of the college
- Official website of SCTCE : www.sctce.ac.in
- HTML and CSS used to build the interface model
- Embed into website in such a way that as soon as the home page opens up the bot button is visible and accessible to the users
- User can ask the required query in the text box provided and press the Send button

DATASET

- Google forms created and distributed amongst students to obtain queries from them
- Queries and responses held in a single file
- JSON format used to store data
- Tags and contexts used to differentiate queries related to different subjects

TECHNOLOGY STACK

FRONT END

HTML and CSS used to build the front end

BACK END

Python 3 used to build the back end

WEB INTEGRATION

To connect front end to the back end

OBSERVATIONS

- To keep track of performance measures while training model, the custom metrices of the model were captured

Number of epochs	8 neuron layer		16 neuron layer	
	Loss	**Accuracy**	**Loss**	**Accuracy**
1000	2.51	0.650	**2.06**	**0.78**
2000	2.58	0.651	3.52	0.50
4000	3.25	0.517	2.49	0.70

- After rigorous training procedures the best suited metrices were chosen for the model and it was seen that the final model had a loss of 2.06 and an accuracy of 0.78.

OBSERVATIONS

- For different batch number of inputs considered and error threshod value, the same procedure was carried out..

Epoch values	**8 neurons**		**16 neurons**	
	Loss	**Accuracy**	**Loss**	**Accuracy**
1000	3.06	0.56	**2.7**	**0.71**
2000	2.81	0.57	3.04	0.51
4000	3.11	0.51	2.5	0.68

RESULTS

- An 80-20 split validation was carried out and it was seen that for every test set of queries given as input correct responses were provided 70% of the times.
- Overfitting of model occurs when number of neurons in hidden layers were increased to 16 and the epoch value was increased to 4000.
- Underfitting of model occured when the number of neurons in hidden layers were made 16 and the epoch value was equal to 2000.
- Sample result : User: hello

  Chatbot: Good to see you, what doubt do you have regarding SCTCE

  User: bye

  Chatbot: Thanks for visiting

CONCLUSION AND FUTURE WORK

- Chatbots are capable of accessing a broad array of information.
- Chatbots are used less in the academic arena thus it needs to be incorporated more into websites of colleges so that information can be passed on to the users without them having to go through the entire websites.
- The proposed chatbot has a general processing structure and makes use of sentence similarity algorithm.
- The present system study is made into a paper titled "Chatbots" and further work on it is being carried out to publish the paper in International Journal of Engineering and Advanced Technology (IJEAT) which is a corpus journal.
- As future work, the proposed model can also be integrated with voice serach module which will help users to search and find required information using voice commands.

REFERENCES

1. Bhavika R. Ranoliya Nidhi Raghuwanshi, Sanjay Singh Department of Information and Communication Technology Manipal University (2017) "Chatbot for University Related FAQs"   International Conference on Advances in Computing, Communications and Informatics.
2. C.B.Lee, H.N.Io (2017) "Chatbots and Conversational Agents: A Bibliometric Analysis" by, IEEE International Conference on Industrial Engineering and Engineering Management.
3. Thomas N T Amrita (2016) "An E-business Chatbot using AIML and LSA" Intl. Conference on Advances in Computing, Communications and Informatics.
4. Bayu Setiaji, Ferry Wahyu Wibowo (2016) "Chatbot Using A Knowledge in Database Human-to-Machine Conversation Modeling" International Conference on Intelligent Systems, Modelling and Simulation.
5. AM Rahman, Abdullah Al Mamun, Alma Islam (2016) "Programming challenges of Chatbot: Current and Future Prospective" IEEE International Islamic University Chittagon.
6. Luis Fernando D'Haro and Rafael E. Banchs (2017) "Learning to predict the adequacy of answers in chat oriented human agent dialogs" IEEE Region 10 Conference (TENCON), Malaysia.

THANKYOU

Example of the Bag-of-Words Model

Let's make the bag-of-words model concrete with a worked example.

Step 1: Collect Data

Below is a snippet of the first few lines of text from the book "A Tale of Two Cities" by Charles Dickens, taken from Project Gutenberg.

> *It was the best of times,*
> *it was the worst of times,*
> *it was the age of wisdom,*
> *it was the age of foolishness,*

For this small example, let's treat each line as a separate "document" and the 4 lines as our entire corpus of documents.

Step 2: Design the Vocabulary

Now we can make a list of all of the words in our model vocabulary.

The unique words here (ignoring case and punctuation) are:

- “it”
- “was”
- “the”
- “best”
- “of”
- “times”
- “worst”
- “age”
- “wisdom”
- “foolishness”

That is a vocabulary of 10 words from a corpus containing 24 words.

Step 3: Create Document Vectors

The next step is to score the words in each document.

The objective is to turn each document of free text into a vector that we can use as input or output for a machine learning model.

Because we know the vocabulary has 10 words, we can use a fixed-length document representation of 10, with one position in the vector to score each word.

The simplest scoring method is to mark the presence of words as a boolean value, 0 for absent, 1 for present.

Using the arbitrary ordering of words listed above in our vocabulary, we can step through the first document ("*It was the best of times*") and convert it into a binary vector.

The scoring of the document would look as follows:

- “it” = 1
- “was” = 1
- “the” = 1
- “best” = 1
- “of” = 1
- “times” = 1
- “worst” = 0
- “age” = 0
- “wisdom” = 0
- “foolishness” = 0

As a binary vector, this would look as follows:

```
[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
```

The other three documents would look as follows:

```
"it was the worst of times" = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
"it was the age of wisdom" = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]
"it was the age of foolishness" = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]
```

All ordering of the words is nominally discarded and we have a consistent way of extracting features from any document in our corpus, ready for use in modeling.

Managing Vocabulary

As the vocabulary size increases, so does the vector representation of documents.

In the previous example, the length of the document vector is equal to the number of known words.

You can imagine that for a very large corpus, such as thousands of books, that the length of the vector might be thousands or millions of positions. Further, each document may contain very few of the known words in the vocabulary.

This results in a vector with lots of zero scores, called a sparse vector or sparse representation.

Sparse vectors require more memory and computational resources when modeling and the vast number of positions or dimensions can make the modeling process very challenging for traditional algorithms.

As such, there is pressure to decrease the size of the vocabulary when using a bag-of-words model.

There are simple text cleaning techniques that can be used as a first step, such as:

- Ignoring case
- Ignoring punctuation
- Ignoring frequent words that don't contain much information, called stop words, like "a," "of," etc.
- Fixing misspelled words.
- Reducing words to their stem (e.g. "play" from "playing") using stemming algorithms.

A more sophisticated approach is to create a vocabulary of grouped words. This both changes the scope of the vocabulary and allows the bag-of-words to capture a little bit more meaning from the document.

In this approach, each word or token is called a "gram". Creating a vocabulary of two-word pairs is, in turn, called a bigram model. Again, only the bigrams that appear in the corpus are modeled, not all possible bigrams.

For example, the bigrams in the first line of text in the previous section: “It was the best of times” are as follows:

- “it was”
- “was the”
- “the best”
- “best of”
- “of times”

A vocabulary then tracks triplets of words is called a trigram model and the general approach is called the n-gram model, where n refers to the number of grouped words.

Often a simple bigram approach is better than a 1-gram bag-of-words model for tasks like documentation classification.