UNIVERSITY OF ARIZONA

Application of Logistic Regression in Automatic Reading Comprehension and Detection of Grammatical Structures Based on Difficulty

Author:
Ragheb AL-GHEZI

Supervisor:
Dr. Robert HENDERSON

A report submitted in fulfillment of the requirements for the degree of Masters of Science

in the

Human Language Technology
Department of Linguistics

February 11, 2019

UNIVERSITY OF ARIZONA

Abstract

Faculty Name
Department of Linguistics

Masters of Science

Application of Logistic Regression in Automatic Reading Comprehension and Detection of Grammatical Structures Based on Difficulty

by Ragheb AL-GHEZI

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

For/Dedicated to/To my…

Chapter 1

Text Representation

1.1 Introduction

Computers do not understand human language the way we do; they are limited by their innate nature of numerical computation. So in order to make use of decades worth of lingusitic theories in the field of human language technology, tremendious human efforts must be extended to represent lingusitic phenomona in a way computers are able to deal with. The goal is to unify the representation of textual documents whatever their formats by transforming them into a set of terms (features) that can be easily used by learning algorithms. Features can be thought of as any rule a lingusit devises to represent particular characteristic of langauge. For instance, the presence of the substring 'win $1000 by clicking here' in an email may indicate for its spamminess. Feature engineering and design is an active research area in natural language processing, and its methods spans between manual and automatic creation.

In order to apply learning algorithms on text, it is necessary to create a numerical representation of the text and assign a weight to each feature in text. This weighting is extremely important as it will influence the results of learning algorithms. Thus, a second objective is to improve weight the features.

The number of features is a very important factor on which the performance of Text Classification depends. Indeed, several learning algorithms are unable to handle a large number of features. For this purpose, it is necessary to reduce the dimensionality of the representation space by keeping only the best features. Reduction methods are used to select or extract the most important features. In this chapter, we present most common automatic methods of text representation, specifying their advantages and disadvantages, and then we discuss the weighting techniques most used in Text Classification as well as the techniques used to reduce dimensionality.

1.2 Classification as a Machine Learning Task

Classification is a supervised Machine learning task aims to select a correct class for a given input, or more generally a task of " assigning objects from a universe to two or more classes or categories " (Manning, Manning, and Schütze, 1999). In natural language processing, the tasks of tagging, word sense disambiguation and spam filtering are all examples of supervised machine learning classification. A classifier is an algorithm that quantitatively models the relationship between a set of inputs and their associated output such that it generalizes well to new input data. In the task of spam

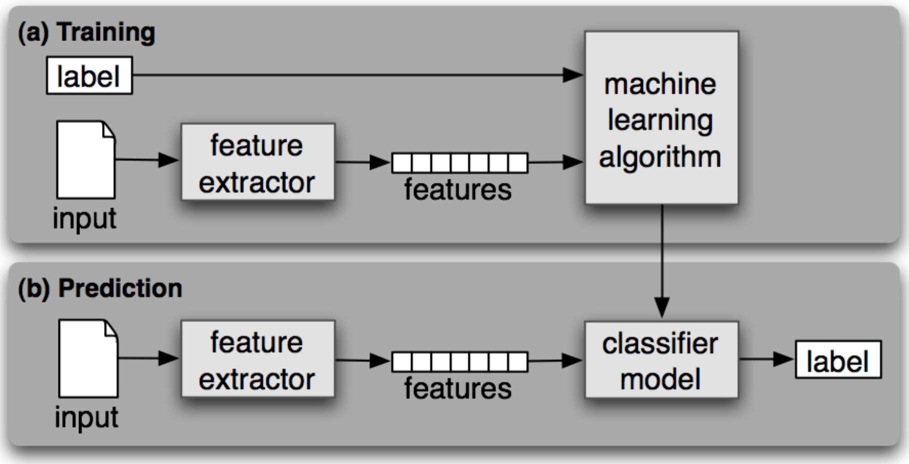


FIGURE 1.1: A figure shows the machine learning workflow

filtering, for instance, input data are set of sentences with binary labels (SPAM, Not SPAM).

CHECKMEOUT: There are two main phases to supervised classification: Training and Prediction. During training, a feature extractor is used to convert each input value to a feature set. These feature sets, which capture the basic information about each input that should be used to classify it, are discussed in the next section. Pairs of feature sets and labels are fed into the machine learning algorithm to generate a model. In prediction phase, the same feature extractor is used to convert unseen inputs to feature sets. These feature sets are then fed into the model, which generates predicted labels.

1.3 Logistic Regression

Logistic regression is one of the most common machine learning classifier in natural language processing tasks. It is also considered the building blocks of neural network models. Logistic regression belongs to a group of probabilistic classifiers called discriminative models which, unlike its generative counterparts like Naive Bayes classifiers, is to try to learn to distinguish the classes. More formally, given a document d and a class c , logistic regressions attempts to compute the conditional probability $P(c|d)$. According to (Jurafsky and Martin, 2014), classification using logistic regression, like any other supervised classification algorithm, has four main components:

- Feature Representation: a method of numerically represent language data (characters, words and sentences, etc) input x as feature vector $[x_1, x_2, x_3, ...x_n]$ where i represents an observation in the input data.
- Classification function: computes an estimation to class $\hat{y}$ via $p(y|x)$. In the case of binary logistic regression, the sigmoid function is used, while a softmax is used for multinomial classification.
- An objective function involving minimizing error on training examples (cross entropy)

- An optimizer to help find the minimum of objective function (stochastic gradient descent).

Logistic regression also has two phases: training and testing. In training phase, the goal is to find the optimal weights that best account for mapping input features x_i to the labelled output y_i of a subset of the data. In testing, we measure the generalizability of weights learned in the training phase on the remaining set of the data.

Given a dataset $\mathcal{D} = \left\{ \left(\boldsymbol{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{N}$, the weights are estimated by maximum conditional likelihood,

$$\log \mathrm{p}\left(\boldsymbol{y}^{(1:N)} | \boldsymbol{x}^{(1:N)}; \boldsymbol{\theta} \right) = \sum_{i=1}^{N} \log \mathrm{p}\left(y^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{\theta} \right) \tag{1.1}$$

$$\sum_{i=1}^{N} \boldsymbol{\theta} \cdot \boldsymbol{f}\left(\boldsymbol{x}^{(i)}, y^{(i)} \right) - \log \sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{f}\left(\boldsymbol{x}^{(i)}, y' \right) \right) \tag{1.2}$$

1.4 Feature Representation

A feature is defined as any element that can be used as an attribute in classification algorithms. Choosing what features to use to represent a text or a document not only has a major impact on the entire classification process, but it is also arguably the most difficult phase in any natural language processing task. The reasons go to several fundamental questions on how to represent language computationally. For example, on what level should we represent a text: *character-wise*, *word-wise* or *sentence-wise*, or even more specifically morphologically, etymologically or phonologically? Do we treat a word e.g. *try* and all its morphological variations e.g. *tries*, *trying*, *tried* similarly or differently? How to represent and model the semantics of human language computationally? Questions like these and many others are the core of natural language processing; therefore, we briefly explore some of the most common text representation methods.

1.4.1 Frequency-based Methods

Bag-of-Words Representation

The "bag of words" representation is the simplest and most intuitive representation. It represents each document by a vector whose component corresponds to the number of occurrences of a word in the document. Words have the advantage of having an explicit meaning. Nevertheless, the implementation of this representation raises several difficulties. The first difficulty is that of the delimitation of words in a text. Indeed, we still can not define what is a word. A word as being a sequence of characters belonging to a dictionary, or formally, as being a sequence of characters separated by spaces or punctuation characters. This definition is not valid for all languages. Indeed, languages such as Chinese or Japanese do not separate their words by spaces. In addition, some separators can be part of certain words (for example: today, 127.0.0.1, potato, etc.). Another difficulty concerns the management of compound words (for example: rainbow, potato, etc.) and acronyms (like: IBM, CAF, CAN, etc.). Consideration of these cases requires quite complex language treatments. This representation of texts excludes any grammatical analysis and any notion of order between words and therefore semantically distant texts can have the same representation. For example, the sentences `the wolf eats the goat` and `the goat eats the wolf` is given the same representation despite being semantically different.

Bag-of-words representation excludes any notion of order and relationship between the words of a text, several studies have attempted to use sentences as features instead of words (Fuhr and Buckley, 1991) (Tzeras and Hartmann, 1993). The use of the sentences makes it possible to solve the problem of ambiguity generated by the use of the representation "bag of words". For example, the word `mouse` has several possible meanings while optical mouse and domestic mouse has no ambiguity. Although the sentences have the advantage of better keeping the semantics in relation to the words, their use as features did not lead to the expected results. According to Lewis (Lewis, 1992), this representation is penalized by the large number of possible combinations which leads to low and too random frequencies. One solution proposed in (Caropreso, Matwin, and Sebastiani, 2001) was to consider a sentence as a set of contiguous (but not necessarily ordered) words that appear together but do not necessarily respect grammatical rules.

A modification to the "bag of words" is representation by lemmas or lexical roots which replaces each word by its canonical form so that words with different forms (singular, plural, masculine, feminine, present, past, future, etc.) can have in a single form called a canonical form. Grouping the different forms of a word offers the advantage of reducing the dimensionality of the learning space. Indeed, in the bag-of-words representation, each form of a word is given a dimension; while with lemma representation the different forms of one word will be merged into one dimension. For example, words such as play, player, players, play, play, play, cheek, etc. will be replaced by a single describer ie the root played or the lemma play.

Lemmatization and stemming are the two techniques used to find the canonical form of a word. Lemmatization uses a knowledge base containing the different inflected forms corresponding to the different possible lemmas. Thus, the inflected forms of a noun will be replaced by the singular masculine form while the different inflected forms of a verb will be replaced by the infinitive form. Lemmatization requires the use of a dictionary of inflected forms of language as well as a grammar labeler. An efficient algorithm, named TreeTagger (Schmid, 1994), has been developed for thirteen different languages: German, English, French, Italian, Dutch, Spanish, Bulgarian, Russian, Greek, French Portuguese, Chinese, Swahili and Old French. Stemming uses a knowledge base of syntactic and morphological rules and to transform words into their roots. One of the most well-known stemming algorithms for the English language is Porter's algorithm (Porter, 1980). Lemmatization is more complicated to implement since it depends on the grammatical labellers. In addition, it is more sensitive to misspellings than stemming.

LISTING 1.1: Python example

```python
def stem(word):
    Suffix_list = ["ing","ly","ed","ious",\
                                "ies","ive","es","s","ment"]
    for suffix in Suffix_list:
        if word.endswith(suffix):
            return word[:-len(suffix)]
    return word
```

A slightly better approach to represent text that makes use of context is the representation by n-grams. It consists of breaking the text into moving sequences of n consecutive tokens. For example, the trigram segmentation of the word "language" gives the following 3-grams: lan, ang, ngu, gua, uag, age. The notion of n-grams was introduced by (Shannon, 1948); he was interested in predicting the appearance of certain characters according to the other characters. Since then, n-grams have been used in many areas such as speech recognition, documentary research, and so on. The use of n-grams offers the following advantages: 1) language-independent 2) Requiring no prior segmentation of the document. 3) Less sensitive to misspellings. 4) effective method to automatic language identification.

Depending on the task, we may sometimes need to aggregate words with certain linguistic relation in concepts. For example, we replace co-hyponyms *dog* and *wolf* with their hypernym *canine*; or *vehicle* and *car* with *automobile*. Concepts, defined as units of knowledge, can be used as features for the purpose of solving the ambiguity problem as well as the problem of synonymy. Indeed, each concept represents a unique meaning that can be expressed by several synonymous words. Similarly, a word with many meanings (sense) is found mapped to several concepts. Thus, a document containing the word *vehicle* may be indexed by other words such as car or automobile. The transition from a word representation to a concept representation requires the use of semantic resources external to the content of documents such as: semantic networks, thesauri and ontologies. As a result, the performance of such a representation crucially depends on the semantic richness of the resources used in terms of the number of concepts and relationships between these concepts.

Weighting of the features

Co-occurrence methods such as bag-of-words has major drawbacks, one of which is terms with higher frequency are more important than those with less frequency. To address this problem, several normalization methods were developed to remove the bias towards more frequent terms. One of these method is Term Frequency Inverse Document Frequency (TFIDF).

TFIDF has two main parts. Term Frequency is proportional to the frequency of the term in the document (local weighting). Thus, the longer the term is common in the document, the more important it is. It can be used as is or in several variations (Singhal, Mitra, and Buckley, 1997) (Sable and Church, 2001).

$$tf_{ij} = f\left(t_i, d_j\right)$$

$$tf_{ij} = 1 + \log\left(f\left(t_i, d_j\right)\right)$$

$$tf_{ij} = 0.5 + 0.5\frac{f\left(t_i, d_j\right)}{max_{t_i \in d_j} f\left(t_i, d_j\right)}$$

where $f(t_i, d_j)$ is the term frequency of document j

Inverse Document Frequency measures the importance of a term throughout the collection (overall weighting). A term that often appears in the document base should not have the same impact as a less frequent term. Indeed, the terms that appear in the majority of documents do not have any discriminating power to distinguish documents from each other and must therefore have low weightings. The IDF weighting is inversely proportional to the number of documents containing the term to be weighted. Thus, the more the term appears in several documents the less it is discriminating and is assigned a low weighting. The IDF weighting is generally expressed as follows:

$$idf\left(t_i\right) = \log\left(\frac{N}{df\left(t_i\right)}\right)$$

where $df(t_i)$ is the term frequency of feature i and N is the number of documents in the corpus.

The TFIDF weighting combines the two weightings TF and IDF in order to provide a better approximation of the importance of a term in a document. According to this weighting, for a term to be important in a document, it must appear frequently in the document and rarely in other documents. This weighting is given by the product of the local weighting of the term in the document by its overall weighting in all the documents of the corpus.

$$tfidf\left(t_i, d_j\right) = tf_{ij} \times \log\left(\frac{N}{df\left(t_i\right)}\right)$$

TFC is a modification to TFIDF that overcomes the major drawback of the TFIDF measure, namely that the length of the documents is not taken into consideration by adding a normalization factor. The TFC weighting of a term i in a document j is calculated as follows:

$$TFC_{ij} = \frac{TFIDF\left(t_i, d_j\right)}{\sqrt{\sum_{k=1}^{T} TFIDF\left(t_k, d_j\right)^2}}$$

1.4.2 Dimensionality reduction

The text representation method explored so far, despite its usefulness in a variety of tasks in natural language processing, share sparsity as one problem in common. The number of features generated using these methods can easily exceed the tens of thousands, and not only does negatively influence the categorization process, it is also very computationally expensive in terms of hardware resources. In addition, the higher the dimensions of the features are, the weaker the features become. This problem is known as the *curse of dimensionality* (Bellman, 2015). To remedy this issue, techniques, borrowed from the field of information theory and linear algebra, have been developed to reduce the feature space dimensionality without losing a lot of information.

Clustering (Baker and McCallum, 1998) (Sable and Church, 2001) (Slonim and Tishby, 2001) can be used to reduce dimensionality. It consists of representing the documents in a new representation space other than the original one. Each dimension of the new representation space groups terms that share the same meaning. Thus, the documents will no longer be represented by terms but rather by groupings of terms representing semantic concepts. This new space of representation offers the advantage of managing the synonymy since the synonymous terms will appear in the same groupings. Likewise, the fact that a term can be included in several groupings also makes it possible to manage the polysemy. Another very interesting method to reduce dimensionality, proposed by (Deerwester et al., 1990) is Latent Semantic Allocation (LSA). It uses singular value decomposition of the document x term matrix to change the representation by keeping only the k axes of strongest singular values. However, LSA is very expensive in terms of calculation time during learning as it relies on a matrix factorization method called Singular Value Decomposition (SVD) which runs computationally in cubic size. Likewise, with each new document, it is necessary to redo the whole grouping process.

1.4.3 Distributional Method

An alternative approach for text representation and natural language of processing is the distributional similarity. The basic idea of the distribution of similarity is to represent the meaning of a word using words which co-occurs in the same context. "You shall know a word by the company it keeps" (Firth, 1957). For example, the words ’dog’ and ’cat’ have the same meaning in the following two sentences: I have a cat, I have a dog. While this idea might not be adequate to capture or account for the complexity or the sophistication of human language, it has succeeded tremendously in a variety of lexicalized NLP tasks. In fact, it has become the de facto representation method of text. The need for such method does not only come from the need for more accurate numerical representation, but its density compared to other methods makes it less computationally expensive. Word2Vec (Mikolov et al., 2013) and GloVe (Pennington, Socher, and Manning, 2014) are two famous algorithms for creating word embedding.

The intuition of word2vec is to train a classifier on a binary prediction task: “Is word w likely to show up near X?”, where X is the word to which we want to find embeddings. Then, we use the weights learned by the classifier as embeddings. Skip-gram algorithm (Mikolov et al., 2013) , first, initializes embeddings randomly. Then, it iteratively updates the embeddings of each word w to be equal to the words they occur with in the same context. Finally, it injects k number non-neighbor words as negative examples.

1.5 Classification and Objective Function

1.5.1 Classification Functions

What we have discussed so far gives us a method to numerically translate language from a form that is uniquely comprehensible to humans into a representation that a computer can understand, and can somehow capture the characteristics of human language. However, in order for the logistic regression to to learn to classify, a classification function should be utilized. Depending on whether the classification task is binary or multinomial, there are two main functions used: sigmoid and softmax. Sigmoid function, used for binary classification, outputs 1 if an observation input (feature vector) is a member of a particular class e.g. (spam), and 0 otherwise. In other words, it calculates $P(y = 1|x)$ for spam text and $P(y = 0|x)$ for non-spam text. The mathematical form of the Sigmoid function or (logistic function, hence comes the name of the classifier) is as follow:

$$Sig(x) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

where x represent the input vector, w is the learned weights and b is the bias term or the intercept of the linear equation. One mathematical property of the Sigmoid function is that it maps any input between 0 and 1, and to make it as probability we need to make the sum of $P(y = 1)$ and $P(y = 0)$ equals to 1.

$$P(y = 1) = \sigma(w \cdot x + b)$$
$$P(y = 0) = 1 - \sigma(w \cdot x + b)$$

Now to make decision, the classifier declares an observation input as *Yes* (if it is a member of a class spam) if the probability is greater than certain threshold say 0.5, and declares an observation input as *No* otherwise.

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

In the case of multinomial classification, the logic remains the same expect for the classification function where Softmax is used rather than Sigmoid. Softmax

$$p(y = i|x) = \frac{e^{w_i \cdot x + b_i}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_j}}$$

works on normalizing the prediction of an observation by the values of all other classes in order to produce valid probability distribution.

1.5.2 Objective Functions

During the learning process, we need to answer the question of *how correct is the estimated output $\hat{y}$ of classifier from the true output y?*, and thus our objective *function* is to minimize the difference between the estimated output and the true one such that the weights learned during this minimization process are, hopefully, generalizable enough to correctly label observations unseen in the training phase. So we can imagine an objective function to be a difference between the $\hat{y}$ and y i.e. $|\hat{y} - y|$, but for mathematical convenience we need a non-convex objective function (or also commonly referred to as loss function) that can be easily optimized. Thus, we use maximum likelihood estimation.

Binary classification can be seen as Bernoulli distribution since the outcome is either 0 or 1. More formally, the probability of predicting class y given input x is $p(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$. Minimizing the probability is the same as maximizing the negative log likelihood:

$$L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

Finally, by plugging in the value of $\hat{y}$ we get:

$$L_{CE}(w,b) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \sigma\left(w \cdot x^{(i)} + b\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(w \cdot x^{(i)} + b\right)\right)$$

, where m is the number of training examples.

On the other hand, multinomial classification uses the same process except that it uses a different classification function *softmax* and hence has different (multinomial) distribution. So the loss function for a single example x is the sum of the logs of the K output classes:

$$L_{CE}(\hat{y}, y) = -\sum_{k=1}^{K} 1\{y = k\} \log \frac{e^{w_k \cdot x + b_k}}{\sum_{j=1}^{K} e^{w_j \cdot x + b_j}}$$

1.6 Optimization

The objective functions derived in the previous section is optimized using numerical methods. Several algorithms are used for this purpose such as Stochastic Gradient Descent, that uses first derivative (gradient) information to find a minimum of a function; Newton-Raphson algorithm uses Second derivative information to find a minimum of a function. Discussing the details of how these algorithms work and the mathematical derivation of gradients is beyond the scope of this work. Interested readers are referred to (Jurafsky and Martin, 2014) for more details on the derivation of Stochastic Gradient Descent. In addition, current machine learning frameworks like Google's Tensorflow and Facebook's Pytorch offer automatic differentiation techniques to find an optimum of a function.

1.7 Evaluation Metrics

1.7.1 Confusion Matrix

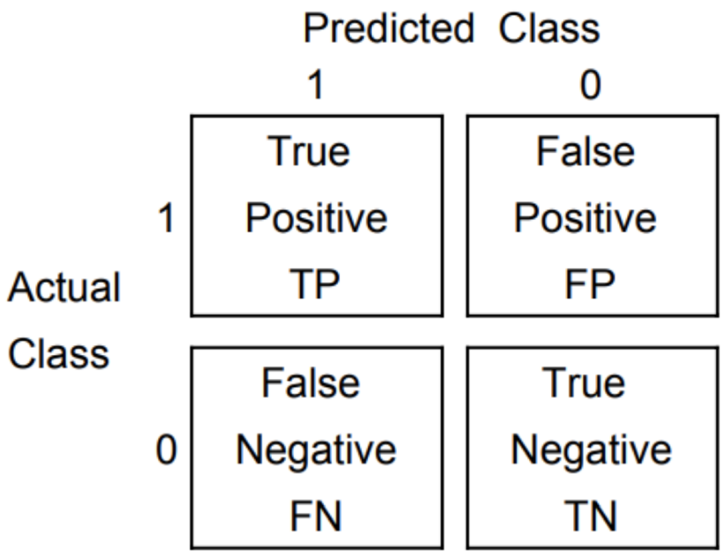
A confusion matrix is a method to visualize the results of a classification algorithm. For the binary classification, the algorithm can be used to predict whether a test sample is either 0, or 1. As a way to measure how well the algorithm performs, we can count four different metrics, here 1 defined as positive and 0 defined as negative:

1. True positive (TP), the algorithm classifies 1 where the correct class is 1.
2. False positive (FP), the algorithm classifies 1 where the correct class is 0.
3. True negative (TN), the algorithm classifies 0 where the correct class is 0.
4. False negative (FN), the algorithm classifies 0 where the correct class is 1.

1.7.2 Precision, Recall and F-Score

Precision and Recall are two popular measurement to evaluate the performance of supervised classification methods. Precision is defined as:

FIGURE 1.2: Confusion Mtrix Example



$$\text{Precision} = \frac{TruePostive}{TruePostive + FalsePositive}$$

, recall defined as:

$$\text{Recall} = \frac{TruePostive}{TruePostive + FalseNegative}$$

and F-score is defined as the harmonic mean of both precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

A classifier with high precision means it classifies almost no inputs as positive unless they are positive. A classifier high recall, on the other hand, would mean that the it misses almost no positive values.

1.8 Conclusion

In this chapter, we briefly introduced some fundamental topics of Machine Learning theory required for natural language processing supervised classification tasks. We discussed two commonly used numerical methods to represent text computationally and their variants, Frequency-based methods: Bag-of word models (BoW) and Term Frequency - Inverse Document Frequency (tfidf), and Distributional similarity methods.In addition, we reviewed some techniques to reduce the dimensionality of feature space to reduce the complexity of the training and save computational resources. Finally, we covered how logistic regression as a probabilistic algorithm can be used for binary or multinomial classification along with the evaluation metrics used to gauge its performance. In the next two chapters, we will examine the use of multinomial logisitic regression classifier on two novel tasks in natural language processing: grammatical difficulty categorization of English sentences and automatic reading comprehension. We also explore how logisitic regression can be used in a semi-supervised manner to work on tasks with scarce data.

Chapter 2

The Use of Semi-supervised Learning in Classification of English Grammatical Structures

2.1 Introduction

In this chapter, we examine the use of multinomal logisitic regression classifier in a semi-supervised manner to automatically categorize English sentences based on difficulty to second langauge learners using Common European Framework Reference. We also compare some of text representation techniques introduced in the previous chapter and determine the best one to use in this task. Finally, we evaluate the overall performance of the classifier before and the semi-supervised data augmentation.

Common European Framework Reference (CERF) is a standard for describing language achievement on a six-point scale, from A1 for beginners, up to C2 for proficient user ability on the four primary skills reading, writing, listening, speaking, and the two secondary skills: vocabulary and grammar. The latter two skills are considered the backbone for the four primary skills in terms of difficulty. For example, frequently uncommon lexical item like *sagacious* makes the sentence in which it appears more difficult to an English learner than a sentence with a more frequent lexical item like *wise*. Similarly, grammatical structures also play a vital role in the overall difficulty of a sentence. For instance, the word *should* conveys different meaning in *I should study for my final exams.* than in *This is your mission should you accept it.*. The latter is considered more difficult to an English learner because the use of *should* as a conditional is less frequent than its use as a modal auxiliary expressing necessity.

Wide World Web and social media are becoming indispensible reosources for second language learners. However, deciding the appropriateness of materials to learners is very labor-intensive task for English teachers. Instructors spent hours sifting through materials online to determine their difficulty and whether or not they are appropriate for their learners. To this end, we leverage the power of machine learning to build a tool that can automatically determine the difficulty of a given text according to CEFR guidelines. CEFR provides 1200 criteria (*English Profile - EGP Online*) for the difficutly of grammatical structures 2.1.

We treat this task as a multinomial classification problem. In our first attempt, we train a logistic regression classifier optimized by Newton-Raphson method on a dataset of around 3000 examples provided by [englishprofile.com], described in details in section 2. We get F1 score of 0.67. In our second attempt, we use the trained classifier from

TABLE 2.1: An excerpt of English sentences and their corresponding CEFR difficulty levels

Sentence	Level
I go there every year with my friends	A1
I think swimming is good for my body.	A2
Tomorrow I'm expecting a delivery of our latest catalogues.	B1
Do not hesitate to contact me should you need further information.	B2
Living in Greece, I have had a chance to realise how much tourism can affect one's life.	C1
There were no photographs of him in Ann 's mother's albums.	C2

our first attempt to predict the difficulty of sentences in an unlabeled corpus to fetch more training. After that, we merge the newly fetched data with the original dataset (obtained from Englishprofile.com) and re-train another multinomial logistic regression classifier from scratch to get F1 score of 0.79.

2.2 Related Works

Sentence classification is the most common task in natural language processing. Considerable number of studies conducted on lexicalized tasks like sentiment analysis, spam filtering, news categorization, etc. Rarely do we see work on classification sentences based on their syntactic structures. Thus, to the best of our knowledge no work has ever tackled the task of classifying the grammatical difficulty of English sentences according to CEFR guidelines. Therefore, we will briefly survey techniques used to solve general sentence classification problems.

Proximity-based Algorithms such as Rocchio's algorithm (Rocchio, 1971) and K-nearest neighbor (Tam, Santoso, and Setiono, 2002) build vector for each class using a training set of document by measuring the similarity, such as Euclidean distance or cosine similarity, between the documents. Some studies (Bang, Yang, and Yang, 2006) incorporated dictionary-based methods to construct a conceptual similarity with KNN classifier, while (Chang and Poon, 2009) combine two KNN classifier with a Naive Bayes one using TF-IDF over phrases to categorize large collections of emails. While proximity-based methods perform well on document classification, yet using them on large training set is not feasible as computing similarities across documents is computationally and resource-wise expensive. Another disadvantage is that noise and irrelevant data can severely degrade the performance of the classification.

Another family of classifier that work well in text categories tasks are decision trees. known for their rule-based approach, Decision trees are favored for their high interpretability (Apté, Damerau, and Weiss, 1994). Classification using this method is done through automatic creating of "if-then" rules. Their use in tasks like spam filtering is very common even with the advent of deep neural network methods (Wu, 2009). Another common powerful classifier is Naive Bayes that based on Baye's rule. It perform surprisingly well for many real world classification applications under some specific conditions (“A Comparison of Event Models for Naive Bayes Text Classication”) (Rish, Hellerstein, and Thathachar, 2001). While Naive Bayes do not often outperform discriminative classifiers like Support-Vector Machine, it has the advantage of functioning well with small training data. (Kim et al., 2006) and (Isa et al., 2008) show good results by selecting Naive Bayes with SVM for text classification and clustering the documents. Using a Poisson Naive Bayes for text classification model also yield very good results (Isa et al., 2008).

The SVM classification method and logistic regression have shown outstanding results when used in text classification tasks (Yang and Liu, 1999) (Brücher, Knolmayer, and Mittermayer, 2002) perhaps because of their capacity of handling high-dimensional sparse data well. However,they can be relatively complex in training and require high time and resource consumption.

2.3 Method

2.3.1 Dataset: English Grammar Profile

The dataset used for this study, provided by English Grammar Profile, exhibits typical grammar profile for each level. It consists of 3615 examples and 1215 rules, divided as follows: class A has 1194 supporting examples; class B has 1775 examples; and class C has 646 supporting examples. We merged each the six levels into three supercategories in order to provide more data within each category.

- **Rule**: Can use 'and' to join a limited range of common adjectives.
- **Example**: The teachers are very nice and friendly.
- **Level**: A1

As we can see, the dataset provides some guideance of what characterizes each example. These rules are prepared to be used by human teachers, and are not very computationally friendly. On the other hand, the small number of examples in the dataset is not enough to build a reliable probablisitic classifier to learn to distingush the various properties of the classes. Therefore, in order to improve the performance, we are using a semi-supervised data augmentation technique similar to Yarawsky's bootstrapping approach (Yarowsky, 1995).

2.3.2 Multinomial Logistic Regression

Suppose that $\mathbf{x}$ is the vectorized sentence in either of its two forms: BoW or TfIDF, $y \in Y$ is the class label, $\mathbf{w}_k$ and b_k are network parameters associated with class y. Then the probability of $\mathbf{x}$ belonging to the class y can be defined by the Softmax function:

$$p(y|\mathbf{x}) = \frac{1}{z(\mathbf{x})} \exp(\mathbf{w}_y^T \mathbf{x} + b_y)\,, \tag{2.1}$$

where $z(\mathbf{x}) = \sum_{j=1}^{K} \exp(\mathbf{w}_j^T \mathbf{x} + b_j)$.

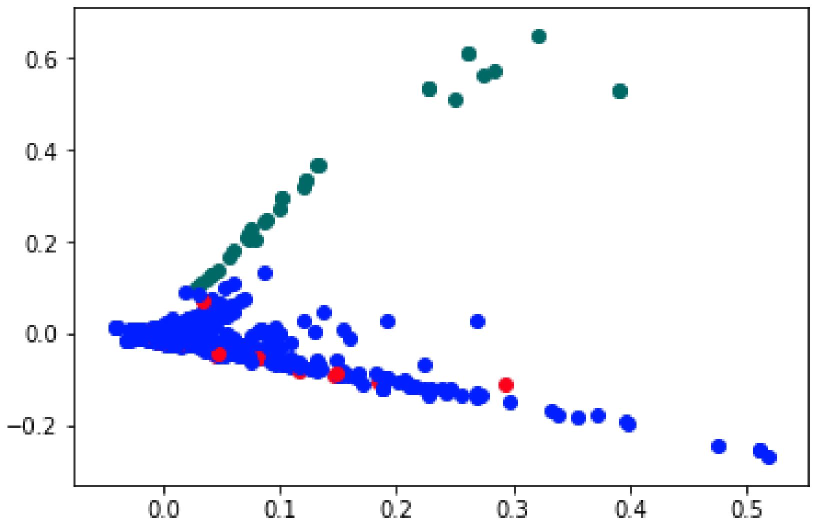


FIGURE 2.1: 2d projection of the three classes A, B and C using K-means Clustering and PCA

Let the log-likelihood function be

$$L(\beta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \beta)$$

.

$$= \sum_{i=1}^{N} \left[\overline{\beta}_{g_i}^{T} x_i - \log \left(1 + \sum_{l=1}^{K-1} e^{\overline{\beta}_l^{T} x_i} \right) \right]$$

To apply Newton-Raphson method, we need the second derivative of the log likelihood function

$$\frac{\partial^2 L(\beta)}{\partial \beta_{kj} \partial \beta_{mn}} = -\sum_{i=1}^{N} x_{ij} x_{in} p_k(x_i; \beta) \left[l(k = m) - p_m(x_i; \beta) \right]$$

The formula for updating β_{new} for multiclass is:

$$\beta^{\text{new}} = \beta^{\text{old}} + \left(\tilde{\mathbf{X}}^T \mathbf{W} \tilde{\mathbf{X}} \right)^{-1} \tilde{\mathbf{X}}^T (\mathbf{y} - \mathbf{p})$$

where y is the concatenated indicator vector of dimension $N \times (K-1)$, p is the concatenated vector of fitted probabilities of dimension $N \times (K-1)$, $\tilde{X}$ is an $N(K-1) \times (p+1)(K-1)$ matrix; and Matrix W is an $N(K-1) \times N(K-1)$ square matrix.

2.3.3 Feature Design

For this task we are comparing two common methods of feature generation in natural language processing and information retrieval: bag-of-word model and term-frequency inverse-document frequency (TFIDF)

Bag of Word Model

a method to represent the occurrence of words within one sentence. We add different variation by counting unigram, bigram and trigram frequencies. In addition, because this model ignores the order or the location of the word in the sentence, it might not be that helpful to our task as it is not lexicalized and the grammar is core of it. Therefore, we replace some lexical words such as nouns, adjactive with their parts-of-speech tags in order to enforce the presence the of the grammatical category, and avoid the influence of the word itself. For example, the sentence this is your mission should you accept it" becomes this is NP should S-PRN VRB O-PRN (CLAUSE)".

TF-IDF

This common information retrieval technique differs from the regular bag-of-words model in that length of the sentence and the frequency of words across sentences does not influence the score of a sentence as it is normalized by other sentences in the dataset. TF·IDF is calculated based on term frequency and document frequency. Term frequency is the number of times a term t occurs in a document d , denoted by TF (t, d) . Document frequency measures the number of documents in which a term t occurs, denoted by DF(t) . TF·IDF is typically calculated as:

$$TF \cdot IDF(t,d) = TF(t,d) \cdot \log \frac{|D|}{DF(t)}$$

, where $|D|$ is the total number of documents.

2.4 Experimental Results

In this section, we evaluate several variations of our method against random model as our baseline model. In the following experiments, all methods are done using Python 3.5 and Scikit Learn Library, tested on MacBook Pro laptop with Core i5 processor and 8 GB of RAM.

2.4.1 First Phase

In the first phase of our experiment, we train a logistic regression classifier, optimized by Newton-Raphson method , on English Grammar Profile dataset. We also introduce a feature design method to mask word categories with their part-of-speech tags in order to preserve grammatical information and avoid the lexical influence of the word. We use a combination of feature design methods such as:

- **BoW**: Apply unigram, bigram and trigram bag-of-words model with both word tokens, and masked tokens.
- **Tf-idf**: Apply unigram, bigram and trigram tf-idf model with both word tokens, and masked tokens.

From table 2, we compare the performance of 4 variations of our method against the baseline, which is a random model. Surprisingly, we notice that our bag-of-words

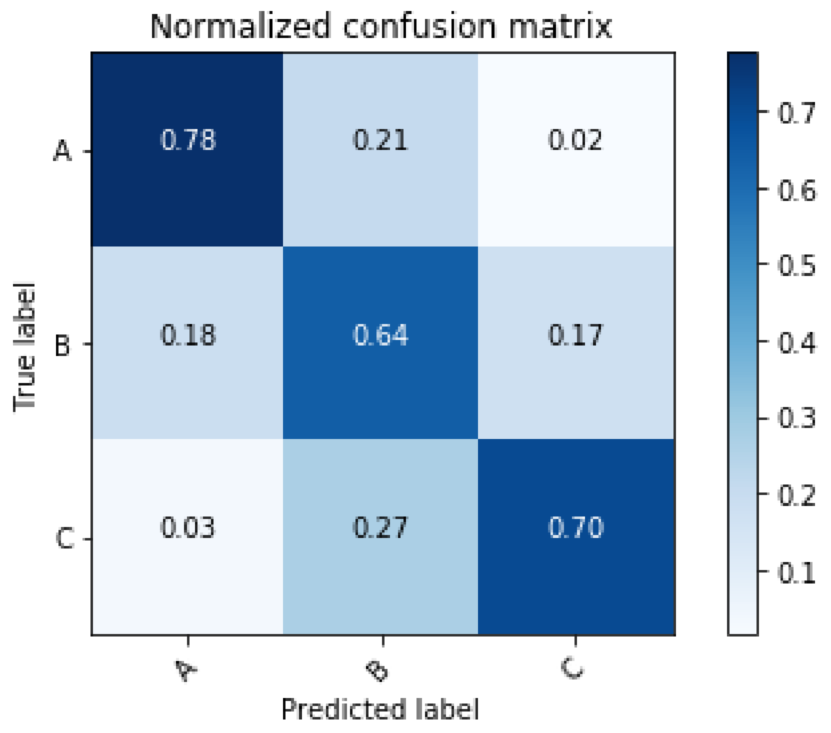


FIGURE 2.2: Confusion matrix illustrating the performance of LR-BoW-Word model with class weight to deal with data imbalance

TABLE 2.2: Comparison of Precision-Recall - Phase One

Model	Precision (%)	Recall (%)	F1 (%)
Random-Baseline	39 (%)	39 (%)	39 (%)
BoW-LR-Words	70 (%)	**69** (%)	**69** (%)
BoW-LR-Masked	66 (%)	62 (%)	63 (%)
Tfidf-LR-Words	**75** (%)	66 (%)	60 (%)
Tfidf-LR-Masked	66 (%)	64 (%)	61 (%)

model with word tokens (BoW-LR-Words) outperform the one with masked sequence in terms of precision and recall respectively. It also outperforms (tfidf-LR-words) model in terms of recall only, while the latter has higher recall. From the confusion matrix (fig. 2), we see that (BoW-LR-Words) model predicts most of testing data correctly even though the number of training data is not equal among the classes, especially for class C. Therefore, we needed to assign certain weights to each class during the training in order to avoid this imbalance in the data.

2.4.2 Second Phase

The results shown in Table 2 does in fact indicate that the linear model has learned the associations between sentences and their corresponding grammatical class reasonably well. However, both of our feature design techniques, namely bag-of-words and tfidf have their disadvantages. The first assigns similarity based on occurrence, size and

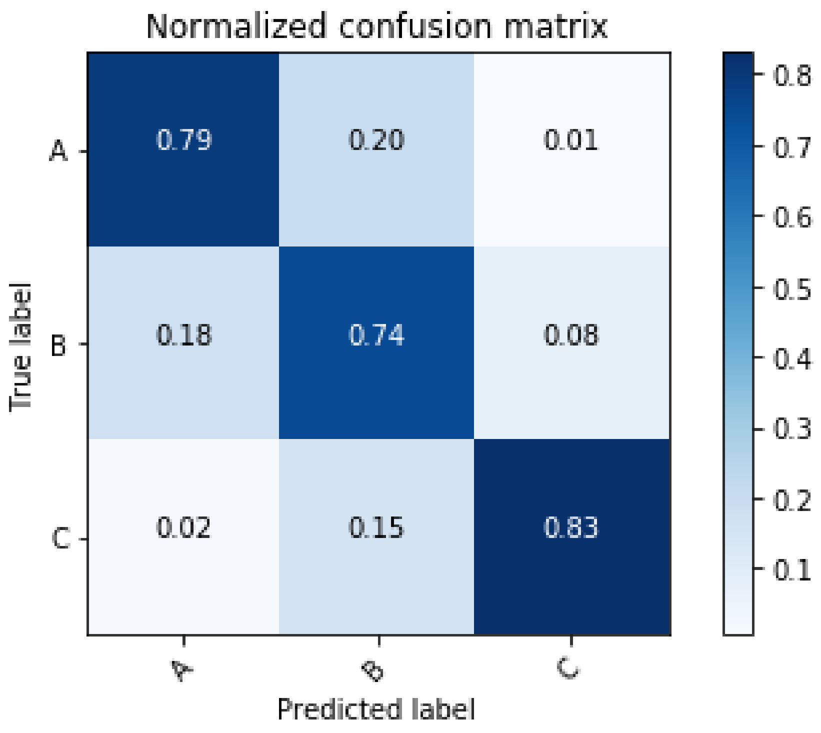


FIGURE 2.3: Confusion matrix illustrating the performance of LR-BoW-Word model after augmentation

mutual words. For example, BoW model learns that longer sentences tend to be the most difficult, while shorter one are less difficult. Similarly, while tfidf treats the drawbacks of BoW model, it still ignores the usefulness of what-so-called *Stop Words* due to their commonality. In other words, it is very hard to trust these results with this amount of training data. Therefore, we propose a non-synthetic data augmentation method to provide more training examples, and thereby improve the overall performance of the model.

Using a text-only version of the brown corpus (Francis and Kucera, 1979) , we use our BoW-LR-Words model to predict the grammatical difficulty labels of its sentences. Then, we collect the sentences predicted with high certainty (above 0.9 probability) to be fed to the original dataset. It is very important to set a higher probability threshold as the new data examples will serve as seed points for our prediction in the future, and we do not want the classifier to *drift* from the original trajectory. This technique is reminiscent of Yarawsky's Bootstrapping (Yarowsky, 1995)(semi-supervised) algorithm, but with one difference is that unlike in Yarawsky's method, we apply this step only once.

Out of 38400 sentences in the unlabeled corpus, only 1428 sentences have been detected with certainty above 90%. We also removed any sentence that is longer than 30 words in order to lessen the undesirable effect of BoW technique. We get apply our best model from phase one to the augmented dataset of 5043 examples under same training conditions to get a precision of **0.80**, recall of **0.79**, and F1 score of **0.79**.

2.5 Conclusion

In this chapter, we have presented a classification method based on simple multinomial logistic regression and bag-of-words model augmented with semi-supervised (bootstrapping) method to classify English sentences based on their level difficulty, A=Beginner, B=intermediate, C=Advanced according to CEFR. We have also compared several common feature design techniques to encode the sentences along with a proposed masking BoW method that replaces content lexical items with their grammatical category. Finally, the model achieves an overall F1 score of 0.69, and 0.79 after augmenting it with example sentences from unlabeled corpus.

Chapter 3

Two-stage Classifcation Method for Automatic Reading Comprehension

3.1 Introduction

Machine Comprehension or Automatic Reading Comprehension is the task of automatically answering comprehension questions based on information derived from short texts. While this task is relatively easy for the humans to do, it is hard for the machine because it requires not only linguistic information, but also cognitive, world-knowledge skills as well. Machine Comprehension has been of great interest to NLP community because of its practical applications in education, especially in the field of automated assessment and intelligent tutoring systems. Working on similar tasks will bring us a step closer to understand how human is able to perform such cognitive tasks so easily while machine is not.

In this chapter, we will be using the second release of Stanford Question Answering Dataset (SQuAD) (Rajpurkar, Jia, and Liang, 2018). SQuAD 2.0 is a reading comprehension dataset, consisting of 150K questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable. The difference between SQuAD 2.0 and its previous release is that it has 50K unanswerable questions in addition to paragraph-supported questions. To do well on SQuAD2.0, a system must not only answer questions when possible, but also abstain from answering when no answer is supported by the paragraph.

The strategy will be used to tackle this task is mainly driven by a simple linguistic intuition. Since SQUAD 2.0 task is an question answering task, the answer span lies within one of given sentences in the paragraph, then an answer is one of the constituents of the best candidate sentence. Thus, our goal is to, first, find the sentence containing the right answer, and we do by training a logistic regression classifier with some linguistic features. Then, we train another classifier to find the constituent best represent the right answer span within the sentence predicted in the first stage. In order to account for the unanswerability of some of the questions, we add a null-answer as a dedicated class in the first classifier along with the potential sentences. This way, if an answer is not found, the inference process stops without proceeding to the next stage, which saves time and computation.

3.2 Related Work

There has been a large number of studies tackle traditional Machine Comprehension tasks, where answers are extractive and explicitly stated in the passage. The difficulty of SQUAD 2.0 task, however, lies in abstaining from answers when no answer span exist in the passage. Thus, we will focus on models that not only answers the questions, but also the one that predict the probability of questions being answered.

(Seo et al., 2016) introduced bi-directional attention flow (BiDAF) that uses an recurrent neural network to encode contextual information in both question and passage along with an attention mechanism to align parts of question to the sentence containing the answer and vise versa. The model offers context representation at multilevel of granularity: character-level, word-level and contextual embedding. What sets this work from others is that it does not represent the context paragraph into fixed-length vector. Instead, it dynamically computes vectors at each time step, combines it with the one from previous layer and allow *flow* through to the next layers. The model outputs confidence scores of start and end index of all potential answers. One problem with this model is it is not designed to handle unanswerable questions. (Levy et al., 2017) extends the work by assigning a probability to null-answers to account for the questions whose answers do not exist in the corresponding paragraph. It achieves 59.2% EM score and 62.1% F1 score.

(Hu et al., 2018) propose a read-then-verify system that is able to abstain from answering when a question has no answer given the passage. They introduce two auxiliary losses to help the neural reader network focus on answer extraction and no-answer detection respectively, and then utilize an answer verifier to validate the legitimacy of the predicted answer. One key contribution is answer-verification. The model incorporates multi-layer transformer decoder to recognize the textual entailment that support the answer in and passage. This model achieves an ExactMatch EM score of 71.6% and 74.23% F1 on SQuAD 2.0.

(Wang, Yan, and Wu, 2018) proposes a new hierarchical attention network that mimics the human process of answering reading comprehension tests. It gradually focuses attention on part of the passage containing the answer to the question. The modal comprises of three layers. a) **encoder layer**: builds representations to both the question and the passage using a concatenation of word-embedding representation (Pennington, Socher, and Manning, 2014) and a pre-trained neural language modal (Peters et al., 2018) b) **Attention layer**: its function is to capture the relationship between the question and the passage at multiple levels using self-attention mechanisms. c) **Matching layer**: given refined representation for both question and passage, a bi-linear matching layer detects the best answer span for the question. This method achieves the state-of-the-art results as of September 2018. Their single model achieves 79.2% EM and 86.6% F1 score, while their ensemble model achieves 82.4% EM and 88.6% F1 Score.

Current models that have shown significant improvement on Machine Comprehension tasks and other similar ones owe their success to a recent neural architecture called transformer networks (Vaswani et al., 2017). It has become the *de facto* in recent sequential learning tasks, eschewing recurrence. This architecture creates global dependencies between input and output using only attention mechanisms. An even more successful model is Bidirectional Encoder Representations from Transformers BERT (Devlin et al., 2018). It is a task-independent pretrained language model that uses a deep transformer network to create rich and context-specific vector representation. Using a method called Masked Language Model, the goal is to randomly mask tokens from the input and predict the vocabulary id of the masked word based only on its context. BERT has shown significant improvement on several tasks, one of which is SQUAD.

While these models achieve astonishing results, they are complex in terms of architecture design and implementation, and very resource-intensive. This complexity results due to lack of linguistic assumptions that can arguably constrain and direct the problem.

3.3 Baseline Implementation

The new version of SQuAD 2.0 task adds a new constraint to competing question-answering models. In addition to identifying the extractive answer spans, a question-answering model should abstain from answering if the passage does not contain an answer. To this end, we implement a simple multinomial logistic regression classifier to address this task. At this level, the classification task is to predict the sentence, in the paragraph, containing the right answer, or declaring that the question is unanswerable. Next, we apply a constituency parser over the sentence predicted from the first stage to get its constituents among which lies the correct answer span (see Figure 1).

3.3.1 Feature Design

To find the sentence containing the answer, a classifier must determine the sentence that is most similar to the question, and by similar we mean that a good candidate sentence 1) shares more words with the question. 2) has high cosine similarity with the question. 3) shares syntactic similarity with the question. Thus, three main features have been selected to this end:

- **Cosine Similarity**: for every sentence in the paragraph as well as the question a word vector representation is created via InferSent (Conneau et al, 2017), which is a pre-trained sentence embeddings method that provides semantic representations for English sentences. InferSent is an encoder based on a bi-directional LSTM architecture with max pooling, trained on the Stanford Natural Language Inference (SNLI) dataset. Cosine distance score is calculated for each sentence-question pair.

- **Word Overlap**: this calculates the Jaccard score between each sentence-question pair. Jaccard index is a method of computing the explicit similarity between two sets as follows:
$$J(Q,S) = \frac{|Q \cap S|}{|Q \cup S|}$$
where Q and S are sets of words in question and sentence respectively.

- **POS Overlap**: This feature computes the Jaccard score over the part-of-speech-tag representation of sentences. In other words, instead of the word tokens, it checks

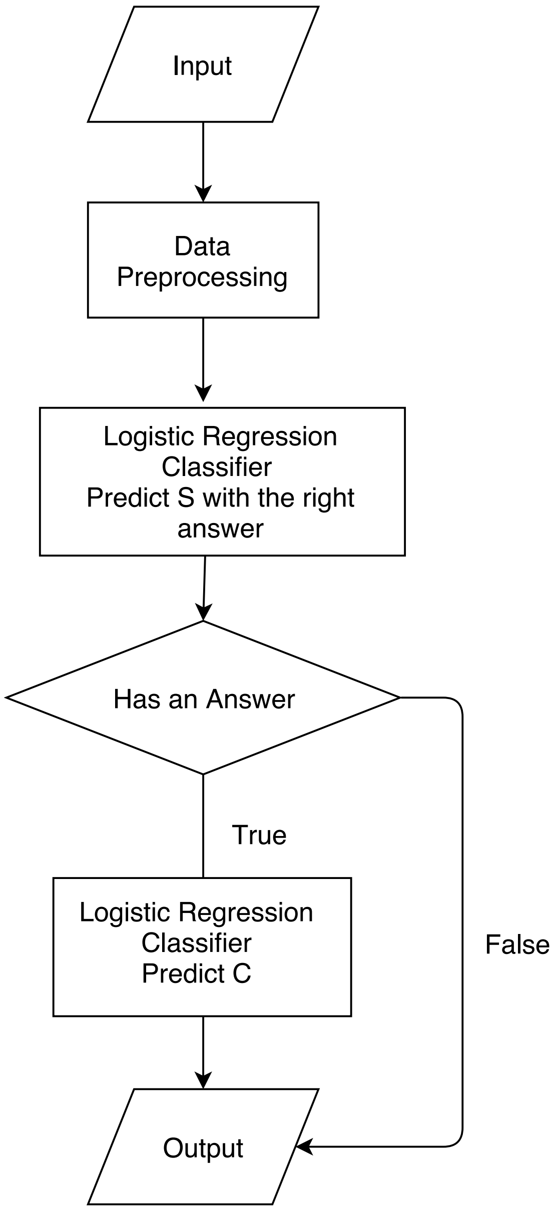
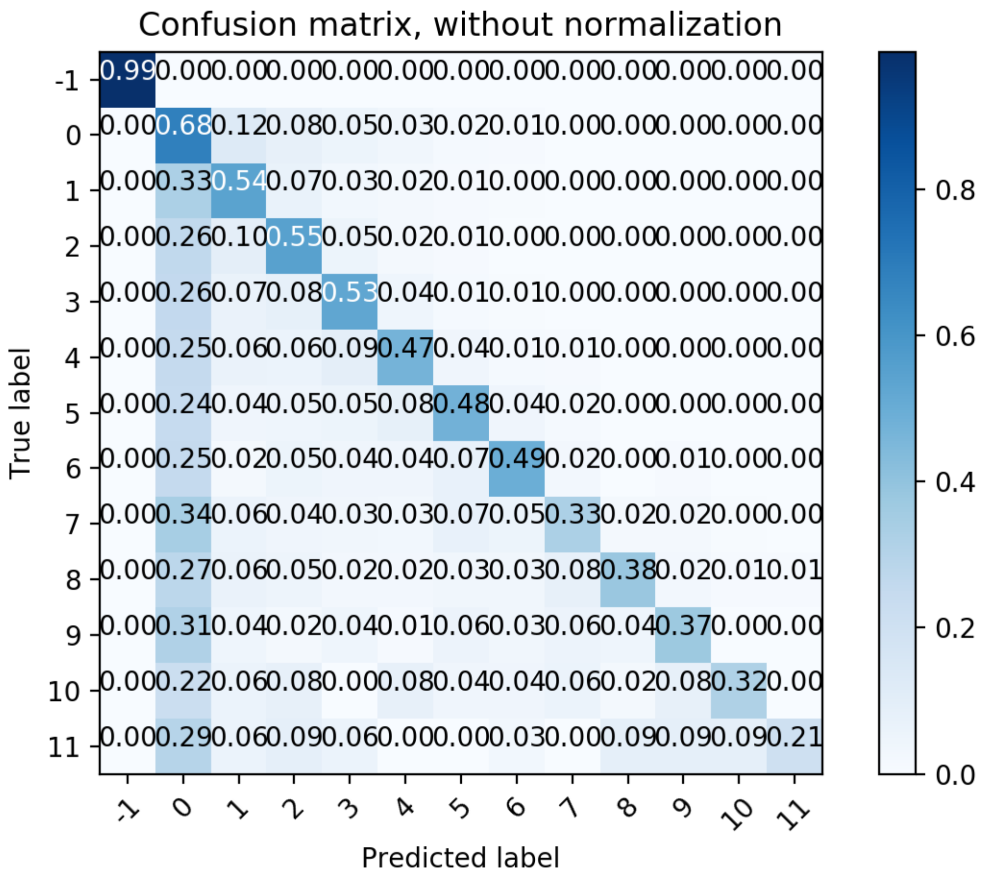

Input
Data Preprocessing
Logistic Regression Classifier Predict S with the right answer
Has an Answer
True
False
Logistic Regression Classifier Predict C
Output

FIGURE 3.1: Flowchart illustrating the two-stage classification approach



True label \ Predicted label	-1	0	1	2	3	4	5	6	7	8	9	10	11
-1	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0	0.00	0.68	0.12	0.08	0.05	0.03	0.02	0.01	0.00	0.00	0.00	0.00	0.00
1	0.00	0.33	0.54	0.07	0.03	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.26	0.10	0.55	0.05	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.26	0.07	0.08	0.53	0.04	0.01	0.01	0.00	0.00	0.00	0.00	0.00
4	0.00	0.25	0.06	0.06	0.09	0.47	0.04	0.01	0.01	0.00	0.00	0.00	0.00
5	0.00	0.24	0.04	0.05	0.05	0.08	0.48	0.04	0.02	0.00	0.00	0.00	0.00
6	0.00	0.25	0.02	0.05	0.04	0.04	0.07	0.49	0.02	0.00	0.01	0.00	0.00
7	0.00	0.34	0.06	0.04	0.03	0.03	0.07	0.05	0.33	0.02	0.02	0.00	0.00
8	0.00	0.27	0.06	0.05	0.02	0.02	0.03	0.03	0.08	0.38	0.02	0.01	0.01
9	0.00	0.31	0.04	0.02	0.04	0.01	0.06	0.03	0.06	0.04	0.37	0.00	0.00
10	0.00	0.22	0.06	0.08	0.00	0.08	0.04	0.04	0.06	0.02	0.08	0.32	0.00
11	0.00	0.29	0.06	0.09	0.06	0.00	0.00	0.03	0.00	0.09	0.09	0.09	0.21

FIGURE 3.2: Confusion matrix shows which classes were predicted correctly

similarity over POS tokens.We use the default POS-tagger in the Spacy library of Python programming language to obtain the POS representation for the sentences and questions alike.

Using the three features above every question-sentence pair will have three scores and an additional binary feature indicating whether or not the question is answerable. This latter feature is provided by the dataset.

3.3.2 Training and Result

We train a logistic regression classifier with L2 regularization ('newton-cg' solver) using scikit-learn library of Python programming language, and we get the results shown in table 1. Numbers in class column represents the index of the sentence in paragraph containing the answer, and -1 indicates that the question has no answer in the paragraph. We also limit the number of sentence to 10. The results show that with simple features we get an F1 score of 0.71.

height**class**	**precision**	**recall**	**F1-score**
-1	1	0.99	0.99
0	0.47	0.68	0.56
1	0.63	0.54	0.58
2	0.62	0.55	0.58
3	0.62	0.53	0.57
4	0.58	0.47	0.52
5	0.57	0.48	0.52
6	0.57	0.49	0.53
7	0.46	0.33	0.38
8	0.56	0.38	0.45
9	0.46	0.37	0.41
10	0.48	0.32	0.39
11	0.35	0.21	0.26
avg / total	0.72	0.71	0.71

TABLE 3.1: This table shows the results of running a multinomial regularized logistic regression. Class column represents the index of sentences within the paragraph, and -1 represent the unanswerable question case. Unpredicted classes are removed.

3.4 Stage Two: Predicting the Answer Span

3.4.1 Iteration 1

To select the most plausible answer span from the candidate sentence, we design a number of features:

- **Constituents**: Using a constituency parser (Kitaev and Klein, 2018), we obtain all constituents in a candidate sentence. These constituents will be the classes from which we pick the right span.
- **Contextual Overlap**: Constituents sharing context with the original question are potential candidates to be the correct answers. So we measure the cosine similarity between each constituent and the question:

$$similarity = \frac{\sum_{i=1}^{n} C_{|w|} Q_i}{\sqrt{\sum_{i=1}^{n} C_{|w|}^2}\sqrt{\sum_{i=1}^{n} Q_i^2}}$$

  where w is the number of slide window around the candidate constituent. For our purposes features of size 2 and 3 are used.
- **Constituent Label**: Constituency parse tree label of the span combined with wh-word.

Out of 85K training example, the answer spans of nearly half of them are not within the constituents. However, answers can be part the constituents. For example, an answer span might be *L'official* and the nearest constituent is *L'official Magazine*. So for our first attempt, we remove all data points whose answers are not explicitly found within

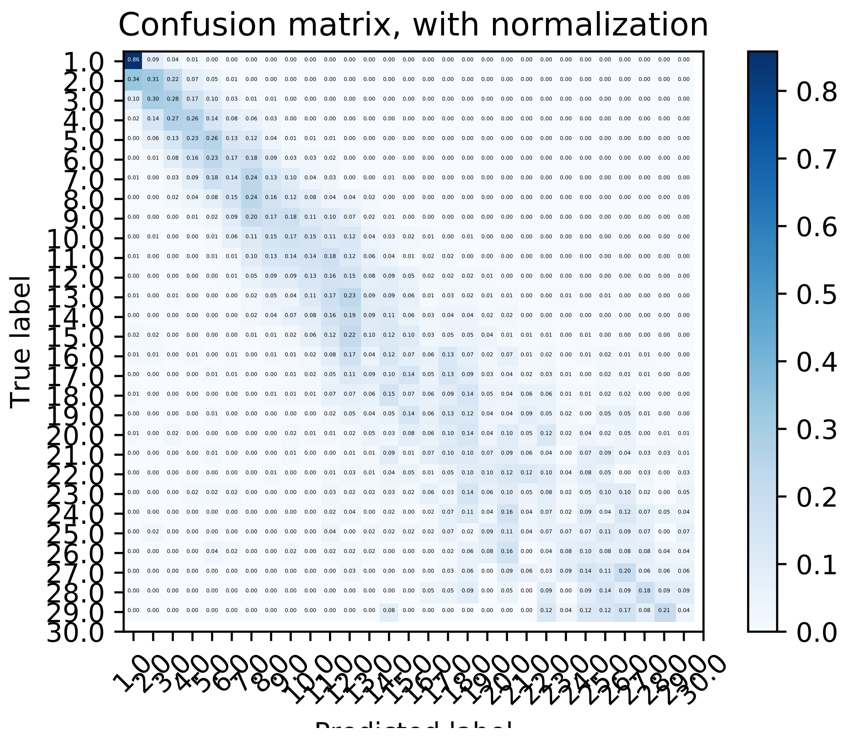


FIGURE 3.3: Confusion Matrix illustrating the first iteration of Stage2 LR.

the constituents. This results in around 45K data point. Next, we train a logistic regression classifier with L2 regularization and 100 epochs, optimized by newton method 'newton-cg' on a Macbook Pro laptop with core i5 processor and 8 gb of RAM. Python modules used are Scikit-learn and Spacy. The latter is used to drive the attentive-neural constituency parser (Kitaev, 2018). The result is 0.25 F1 score.

3.4.2 Error Analysis: Iteration 1

By looking at the confusion matrix the first thing we notice is data imbalance. Classes at the beginning have more supporting points, and therefore has less error than the others. However, class 2 has been identified as class 1 34%. For example, the answer of *At what age did Beyonce meet LaTavia Robertson?* is predicted as *At age eight* while the correct answer is *age eight*. Another example, the answer for *How much was public expenditure on the island in 2001-2002?* is predicted to be *Public expenditure* while the true answer is *£10 million*. In this case, the phrase public expenditure appears in the question, which is a stronger candidate given the features designed. Similarly, class 12 is predicted as class eleven 16%. For example, the answer to the question *Who supervised the design and implementation of the iPod user interface?* is *Steve Jobs*, but it is predicted as *Of Steve Jobs*. Another example, answer to *What roles were women recruited for in the 1950s?* is *in medicine, communication*, but it is predicted as *logistics, and administration*. Given the features we have designed it is very difficult for the classifier to recognize this settle difference between the classes.

3.4.3 Iteration 2

In the first attempt we have sacrificed valuable data because the answer span was not within the constituents of the candidate sentence. This time, we use all 85K but we will face the same problem introduced in the previous section. The answer span can be a member of more than one constituent in the same sentence, and this will impose a very hard constraint on the performance because the inference can be close but not exact. Also because this problem requires more feature engineering efforts, we decided to leave it for future studies. Instead, we will set the target to the first constituent of which the span is a member. However, we will add three more features:

- **Distributional Distance**: We measure the distributional cosine similarity between the sum of all words in the contextual window and the question using Glove (Pennington, 2012).
- **Matching Word Frequencies**: Sum of the TF-IDF of the words that occur in both the question and the sentence containing the candidate answer.
- **Lengths**: Number of words to the left and to the right of the span.

For classification we compare the performance of a logistic regression classifier with similar configuration as the first stage to 3-layer feed-forward neural network of 128, 64 and 32 nodes respectively. The logistic regression achieves 0.35 F1 while the neural network does 0.42 F1 using Adam optimizer and 100 epochs. Looking at the confusion matrix it is very clear that the new features (tf-idf and distributional cosine similarity) improve the performance. However, they could not recognize the settle difference among the phrase.

3.5 Conclusion

We introduced a two-step classification method for automatic reading comprehension via SQUAD 2.0 dataset. Our stage1 classifier managed to find whether or not a question is answerable within a given passage and find the sentence containing the right answer with F1 score of 0.71. Our stage2 classifier manages to detect the exact span with F1 score of 0.35 even though the predicted answer is not distant from the exact answer. In order to improve the performance of our approach, future studies should investigate the usefulness of features generated from Named Entity Recognition, Semantic Role Labeling and Dependency Parsing processes, which are expected to be potential solutions to the problems we faced in this work.

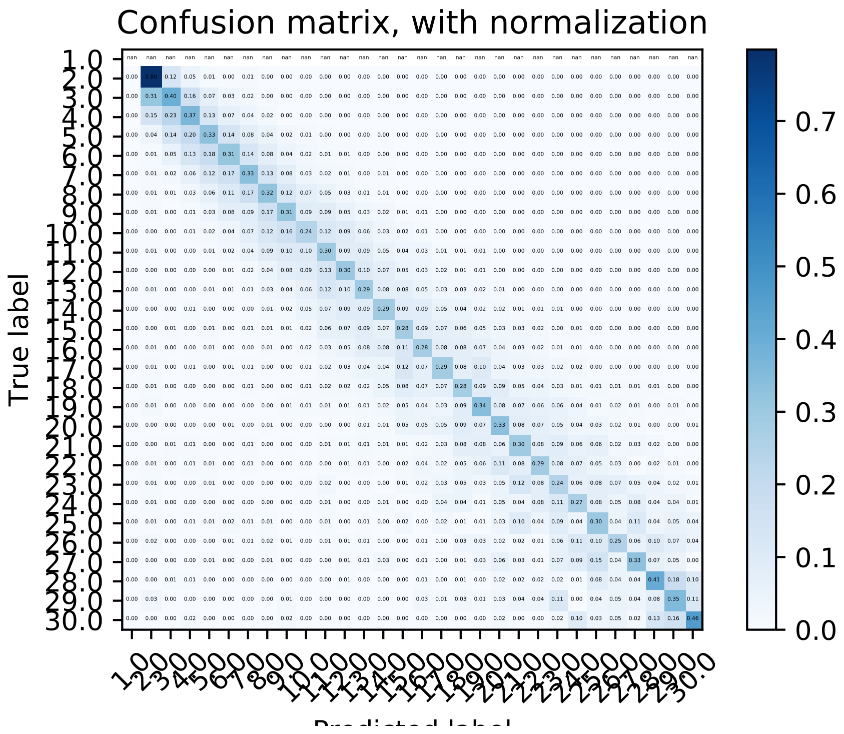


FIGURE 3.4: Confusion Matrix illustrating the Second iteration of Stage2 LR.

Bibliography

Apté, Chidanand, Fred Damerau, and Sholom M. Weiss (1994). "Towards Language Independent Automated Learning of Text Categorization Models". en. In: *SIGIR '94*. Ed. by Bruce W. Croft and C. J. van Rijsbergen. Springer London, pp. 23–30. ISBN: 978-1-4471-2099-5.

Baker, L Douglas and Andrew Kachites McCallum (1998). "Distributional clustering of words for text classification". In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 96–103.

Bang, Sun Lee, Jae Dong Yang, and Hyung Jeong Yang (Mar. 2006). "Hierarchical document categorization with k-NN and concept-based thesauri". In: *Information Processing & Management* 42.2, pp. 387–406. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2005.04.003. URL: http://www.sciencedirect.com/science/article/pii/S0306457305000579 (visited on 12/09/2018).

Bellman, Richard E (2015). *Adaptive control processes: a guided tour*. Vol. 2045. Princeton university press.

Brücher, Heide, Gerhard Knolmayer, and Marc-André Mittermayer (2002). "Document classification methods for organizing explicit knowledge". In:

Caropreso, Maria Fernanda, Stan Matwin, and Fabrizio Sebastiani (2001). "A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization". In: *Text databases and document management: Theory and practice* 5478, pp. 78–102.

Chang, Matthew and Chung Keung Poon (June 2009). "Using phrases as features in email classification". In: *Journal of Systems and Software* 82.6, pp. 1036–1045. ISSN: 0164-1212. DOI: 10.1016/j.jss.2009.01.013. URL: http://www.sciencedirect.com/science/article/pii/S0164121209000089 (visited on 12/09/2018).

Deerwester, Scott et al. (1990). "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6, pp. 391–407.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

English Profile - EGP Online. URL: http://www.englishprofile.org/english-grammar-profile/egp-online (visited on 01/29/2019).

Firth, John R (1957). "A synopsis of linguistic theory, 1930-1955". In: *Studies in linguistic analysis*.

Francis, W. Nelson and Henry Kucera (1979). *The Brown Corpus: A Standard Corpus of Present-Day Edited American English*. Brown University Liguistics Department.

Fuhr, Norbert and Chris Buckley (1991). "A probabilistic learning approach for document indexing". In: *ACM Transactions on Information Systems (TOIS)* 9.3, pp. 223–248.

Hu, Minghao et al. (2018). "Read + Verify: Machine Reading Comprehension with Unanswerable Questions". In: *CoRR* abs/1808.05759. arXiv: 1808.05759. URL: http://arxiv.org/abs/1808.05759.

Isa, D. et al. (Sept. 2008). "Text Document Preprocessing with the Bayes Formula for Classification Using the Support Vector Machine". In: *IEEE Transactions on Knowledge and Data Engineering* 20.9, pp. 1264–1272. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.76.

Jurafsky, Dan and James H Martin (2014). *Speech and language processing*. Vol. 3. Pearson London.

Kim, Sang-Bum et al. (Nov. 2006). "Some Effective Techniques for Naive Bayes Text Classification". In: *IEEE Transactions on Knowledge and Data Engineering* 18.11, pp. 1457–1466. ISSN: 1041-4347. DOI: 10.1109/TKDE.2006.180.

Kitaev, Nikita and Dan Klein (2018). "Constituency Parsing with a Self-Attentive Encoder". In: *arXiv preprint arXiv:1805.01052*.

Levy, Omer et al. (2017). "Zero-Shot Relation Extraction via Reading Comprehension". In: *CoRR* abs/1706.04115. arXiv: 1706.04115. URL: http://arxiv.org/abs/1706.04115.

Lewis, David Dolan (1992). "Representation and learning in information retrieval". PhD thesis. University of Massachusetts at Amherst.

Manning, Christopher D, Christopher D Manning, and Hinrich Schütze (1999). *Foundations of statistical natural language processing*. MIT press.

McCallum, Andrew and Kamal Nigam. "A Comparison of Event Models for Naive Bayes Text Classication". en. In: (), p. 8.

Mikolov, Tomas et al. (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Peters, Matthew E. et al. (2018). "Deep contextualized word representations". In: *Proc. of NAACL*.

Porter, Martin F (1980). "An algorithm for suffix stripping". In: *Program* 14.3, pp. 130–137.

Rajpurkar, Pranav, Robin Jia, and Percy Liang (2018). "Know What You Don't Know: Unanswerable Questions for SQuAD". In: *arXiv preprint arXiv:1806.03822*.

Rish, Irina, Joseph Hellerstein, and Jayram Thathachar (2001). *An analysis of data characteristics that affect naive Bayes performance*. Tech. rep.

Rocchio, Joseph John (1971). "Relevance feedback in information retrieval". In: *The SMART retrieval system: experiments in automatic document processing*, pp. 313–323.

Sable, Carl and Kenneth W Church (2001). "Using bins to empirically estimate term weights for text categorization". In: *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*.

Schmid, Helmut (1994). *Probabilistic part-of-speech tagging using decision trees. In proceedings of international conference on new methods in language processing,(pp. 1-9), Access date: 09.11. 2012.*

Seo, Min Joon et al. (2016). "Bidirectional Attention Flow for Machine Comprehension". In: *CoRR* abs/1611.01603. arXiv: 1611.01603. URL: http://arxiv.org/abs/1611.01603.

Shannon, Claude Elwood (1948). "A mathematical theory of communication". In: *Bell system technical journal* 27.3, pp. 379–423.

Singhal, Amit, Mandar Mitra, and Chris Buckley (1997). "Learning routing queries in a query zone". In: *ACM SIGIR Forum*. Vol. 31. SI. ACM, pp. 25–32.

Slonim, Noam and Naftali Tishby (2001). "The power of word clusters for text classification". In: *23rd European Colloquium on Information Retrieval Research*. Vol. 1, p. 200.

Tam, V., A. Santoso, and R. Setiono (Aug. 2002). "A comparative study of centroid-based, neighborhood-based and statistical approaches for effective document categorization". In: *Object recognition supported by user interaction for service robots*. Vol. 4, 235–238 vol.4. DOI: 10.1109/ICPR.2002.1047440.

Tzeras, Kostas and Stephan Hartmann (1993). "Automatic indexing based on Bayesian inference networks". In: *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 22–35.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *CoRR* abs/1706.03762. arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

Wang, Wei, Ming Yan, and Chen Wu (2018). "Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering". In: *CoRR* abs/1811.11934. arXiv: 1811.11934. URL: http://arxiv.org/abs/1811.11934.

Wu, Chih-Hung (Apr. 2009). "Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks". In: *Expert Systems with Applications* 36.3, Part 1, pp. 4321–4330. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2008.03.002. URL: http://www.sciencedirect.com/science/article/pii/S0957417408001772 (visited on 12/09/2018).

Yang, Yiming and Xin Liu (1999). "A Re-Examination of Text Categorization Methods". In: *SIGIR*.

Yarowsky, David (June 1995). "Unsupervised word sense disambiguation rivaling supervised methods". In: *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, Massachusetts, USA: Association for Computational Linguistics, pp. 189–196. DOI: 10.3115/981658.981684. URL: http://www.aclweb.org/anthology/P95-1026 (visited on 12/09/2018).