Chapter 1

Text Representation

1.1 Introduction

Technically, a text is a set of symbols saved in a digital format. Some of these formats are intended only for the distribution of documents such as Portable Document Format (PDF) or PS (PostScript Format), while others are determined by environments with well-defined rules such as the Microsoft Word DOC format, the Open Office SXW format, or the LATEX format. 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.

In order to apply learning algorithms on text, it is necessary to create a numerical representation of the text andassign 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 will present the different methods of text representation, specifying their advantages and disadvantages, and then mention the weighting techniques most used in Text Classification as well as the techniques used to reduce dimensionality.

1.1.1 Feature Selection

A feature can be defined as any element that can be used as an attribute in classification algorithms. As the feature selection is the first step in the categorization process, it is essentially important to choose which features to use for represent the documents. In fact, this choice will have a major impact on the rest of the process.

Bag-of-Words Representation The representation "bag of words" is the simplest and most intuitive representation. It consists in representing each document by a vector whose component corresponds to the number of occurrences of a word in the document. It has been used in many works to represent textual documents such as [90, 85]. 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. R.Gilly

considers [51] 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. Add to this that 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.

Representation by sentences Bag-of-words representation excludes any notion of order and relationship between the words of a text, several searches have attempted to use sentences as features instead of words [46, 152]. 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 [93], this representation is penalized by the large number of possible combinations which leads to low and too random frequencies. One solution proposed in [17] 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.

Representation by lemmas or lexical roots The representation by lemmas or lexical roots is an extension of the representation "bag of words" which consists of replacing each word by its canonical form. Thus, it is a question of grouping the different forms that a word (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 following two advantages:

- The reduction of the dimensionality of the space of representation. Indeed, in the representation "word bag", each form of a word is given a dimension; while with lemma representation the different forms 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.

- The increase of the occurrences of the features. Indeed, it is better to consider a single play feature having seven occurrences than to consider the seven features playing, player, players, playable, playing, playing, cheek, etc. with one occurrence of each.

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 [137], has been developed for thirteen different languages: German, English, French, Italian, Dutch, Spanish, Bulgarian, Russian, Greek, French Portuguese, Chinese, Swahili and Old French. Racinization uses a knowledge base of syntactic rules and grammatical languages to transform words into their roots. One of the most well-known stemming algorithms for the English language is Porter's algorithm [115]. Lemmatization is more complicated to implement since it depends on the grammatical labellers. In addition, she is more sensitive to misspellings than stemming.

Representation by N-grams In contrast to the previously described methods of representation, the representation by N-grams excludes all notions of language in the representation of a textual document. It consists of cutting the text to be represented in several sequences of n consecutive characters. Thus, it is a question of moving on the text by step of a character and to select the n characters with each movement. For example, the division of the text "the search for information" will give as results the following 3-grams: la-, ar, -re, rec, ech, che, her, erc, rch, che, he-, ed, - d, of,, in, inf, nfo, for, orm, rma, mat, ati, tio, ion. The notion of n-grams was introduced by Shannon [139] in 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 [75]:

- Independence with respect to the language of the document.
- Does not require any prior segmentation of the document. - Less sensitive to misspellings.
- Allows you to identify the language of a document.

Representation by concepts All the methods described previously are statistical methods based on the number of occurrences of the features, excluding any notion of semantic relation between these different features. This notion of semantics being neglected in statistical methods, ambiguity remains a major handicap. Taking into consideration the example of Figure 3.1 illustrating four documents from the Reuters-21578 corpus belonging to the same category "corporate acquisition" do not share any common words (except for empty words), the statistical methods will consider the four documents totally independent. 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 is found mapped in several concepts. The conceptual representation improves the representation of a document by allowing it to be enriched by features semantically close to the original features. 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

Once all the features representing our textual documents are fixed, this step consists of weighting the features according to their relative importance in the documents. A good weighting of a feature in a document must take into account the local aspect, the overall aspect as well as the standardization aspect. Generally the weighting of a feature i in a document j implies the computation of:

$$w_{ij} = L_{ij} G_i N_i$$

where:

- L_{ij} is the local weighting of feature i in document j. This weighting is usually based on the number of occurrences of the feature in the document.
- G_i is the global weighting of feature i in the document collection. This weighting disadvantages the most common features in the collection. N_j is the normalization factor of the weight of the document j. This factor is aim to eliminate the influence of the size of documents.

Depending on the choices on these three aspects, several techniques were used to weight the features.

TF (Term Frequency) This measure 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 [126, 141].

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

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

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

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

IDF (Inverse Document Frequency) This weighting 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.

TFIDF 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 This measure makes it possible to overcome 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:

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

Dimensionality reduction

Like any other field dealing with natural language, the categorization of texts as a problem the large dimension of the space of representation. Indeed, the number of features for a corpus of reasonable size can be tens of thousands. This great dimensionality negatively influences the subsequent categorization process by generating two problems:

- a high cost of treatment. Indeed, the more the number of dimensions is high, the more the calculation volume is important;

- Impossibility to build reliable rules from low feature frequencies. Indeed, the more the number of dimensions is high, the more the occurrences of the features become weak. The techniques used for dimension reduction come from the theory of information and linear algebra. Sebastaini [44] classifies these techniques in two ways: i) depending on whether they act locally or globally, and ii) depending on the nature of the results of the selection (is it a selection of terms or a term extraction)?

  Local dimension reduction It is a question of proposing, for each category c a new set of term T ii with i | Ti | . Thus, each category ci has its own set of terms and each document dj will be represented by a set of different vectors dj according to the category.

  Overall size reduction In this case, the new set of terms T is chosen according to all categories. Thus, each document dj will be represented by a single vector whatever the category.

  Selection of terms The dimension reduction techniques by selection consist in filtering the set of features by proposing a subset of features considered relevant in relation to the other features. These techniques include:

1. MI (Mutual Information): The MI technique measures the mutual dependence between a word t_k and a category c_i. His formula is as follows [168]:

$$MI\left(t_k, c_i\right) = \log \frac{A\left(t_k, c_i\right) N}{\left(A\left(t_k, c_i\right) + C\left(t_k, c_i\right)\right)\left(A\left(t_k, c_i\right) + B\left(t_k, c_i\right)\right)}$$

- N is the number of documents in the learning corpus.
- $A(t_k, c_i)$ is the number of documents containing the term t_k and belonging to category c_i.
- $B(t_k, c_i)$ is the number of documents containing the term t_k and not belonging not in category c_i.
- $C(t_k, c_i)$ is the number of documents that do not contain the term t_k and belong to holding in category c_i.

This technique favors sparsely populated categories compared to other categories which leads to performance degradation. This problem has been solved by another alternative proposed in [9] whose formula is as follows:

$$MI\left(t_k, c_i\right) = p\left(t_k | c_i\right) \log \frac{A\left(t_k, c_i\right) N}{N\left(c_i\right) N\left(t_k\right)}$$

1. Information Gain (IG): The IG method measures the amount of information obtained for predicting the category knowing the presence or absence of a term in a document [168]. His formula is:

$$IG\left(t_k\right) = \sum_{c \in \{c_i, \bar{c}_i\}} \sum_{w \in \{w_k, \overline{w_k}\}} p(c|w) \times \log \frac{p(c|w)}{p(w) \times p(c)}$$

Several research studies have evaluated the different selection techniques. A comparison between the IG and MI techniques was performed in [168] on the Ohsumed and Reuters-21578 corpora. This comparison showed that the best performances were obtained with IG techniques.

Extraction of terms Clustering [142, 129, 7] is another alternative for reducing 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 for the extraction of terms and that proposed by S. Deerwester and S. Dumais in [30]. This method called LSA is based on a singular value decomposition of the document x term matrix. The purpose of this decomposition is to change the representation by keeping only the k axes of strongest singular values. The reduction methods based on the extraction of terms are very expensive in terms of calculation time during learning. Likewise, with each new document, it is necessary to redo the whole grouping process.

Conclusion

Since text representation is the first step in the text categorization process. It has attracted a lot of attention from researchers in this field. In fact, a poor representation of the documents will negatively influence the rest of the process. The result of this step is the construction of a matrix features x documents whose intersection between line i and column j represents the weight of the ith feature in the jiemedocuments. In this chapter, we have discussed

the steps to follow to achieve a representation of texts namely: Choice of features: The representation "bag of words" is undoubtedly the most used representation in the field of the text classification. However, the researches are more and more accentuated towards the semantic representations.

- Weighting of features: The majority of the weights used in the categorization of texts are inherited from the domain of the IR, and more particularly from the vector model.

- Dimensionality reduction: The number of dimensions is a very important factor for classifiers. Indeed, classifiers such as SVMs can handle a large number of dimensions while other classifiers including neural networks are not.

Chapter 2

Categorization of English Structures Using CEFR

Abstract

Determining the difficulty of grammatical structures is essential in language learning and teaching. A teacher constantly needs to provide their students with lessons and quizzes that match their level of proficiency. Common European Framework Reference (CERF) categorizes language learning skills into six level according to difficulty (A1, A2, B1, B2, C1, C2) where A1 refers to easiest, and C2 refers to the hardest. To this end, we propose a method to automatically detect the difficulty level of a given text using a supervised and semi-supervised machine learning classifier. Our method achieves .81 F1 score.

2.1 Introduction

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 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. and This is your mission should you accept it. The latter is considered more difficult to an English learner as the word should is less commonly appear in this grammatical structures.

With the rapid development of Wide World Web and social media, countless number of digital texts and video materials can be harnessed for language learning and teaching[@cite authentic material]. 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. Unlike lexical difficulty that can be treated as simple search problem, grammatical difficulty is not a trivial task. CEFR provides 1200 criteria[@englishprofile.com] for the difficutly of grammatical structures

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

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 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[75] and K-nearest neighbor[66] 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 [67] incorporated dictionary-based methods to construct a conceptual similarity with KNN classifier, while [60] 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 [64,65]. 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 [49]. 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 [100,101,102,104]. While Naive Bayes do not often outperform discriminative classifiers like Support-Vector Machine, it has the advantage of functioning well with small training data. [83] and [84] shows 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 [85].

The SVM classification method and logistic regression have shown outstanding results when used in text classification tasks [99, 111,110,70] 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.

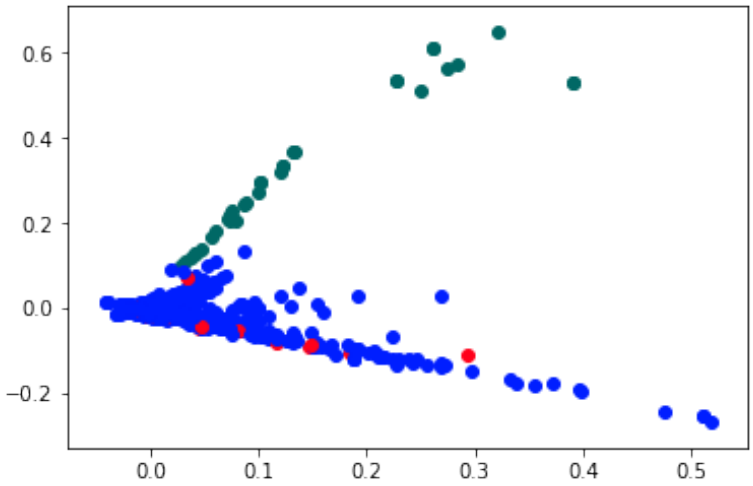

Figure 2.1: 2d projection of the three classes A, B and C using K-means Clustering and PCA

- **Example**: The teachers are very nice and friendly.
- **Level**: A1

As we can see, the dataset provides some guideance of what characterizes each example. Using these rules, one can easily create a set of regular expression rules and solve the problem symbolically. However, using a statistical approach to detect the difficulty class without hand-crafting rule is very challenging as there are not enough examples for each case within each category. Thus, in order to get good results, we are using a semi-supervised data augmentation technique similar to Yarawsky's bootstrapping approach (Yarawsky, 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 $z(\mathbf{x}) = \sum_{j=1}^{K} \exp(\mathbf{w}_j^T \mathbf{x} + b_j)$.

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 (Dummy) 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

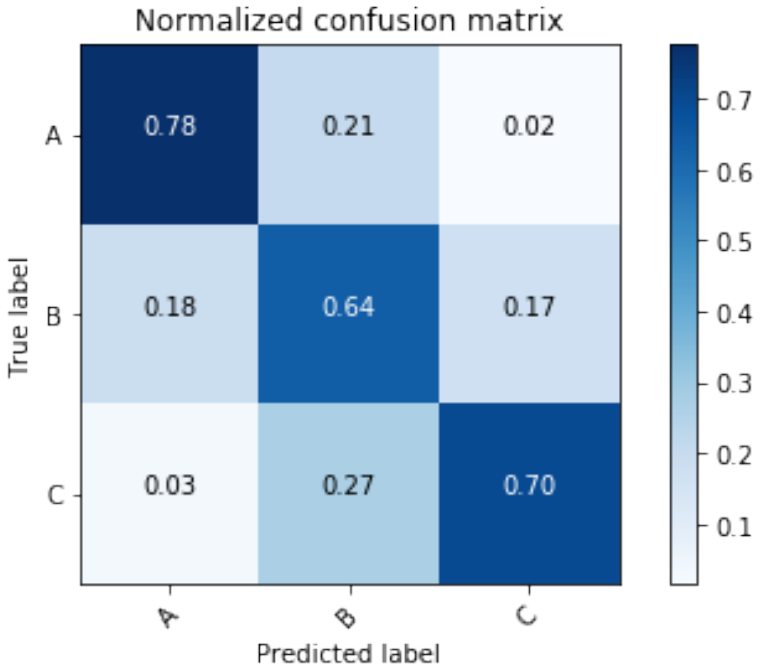

Figure 2.2: Confusion matrix illustrating the performance of LR-BoW-Word model with class weight to deal with data imbalance

our bag-of-words 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.

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 (%)

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 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, 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 (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 paper, 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 in accordance with Common European Framework Reference (CEFR) guidelines for language learning and teaching. Our model achieves an overall F1 score of 0.69, and 0.79 after augmenting it with example sentences from unlabeled corpus.

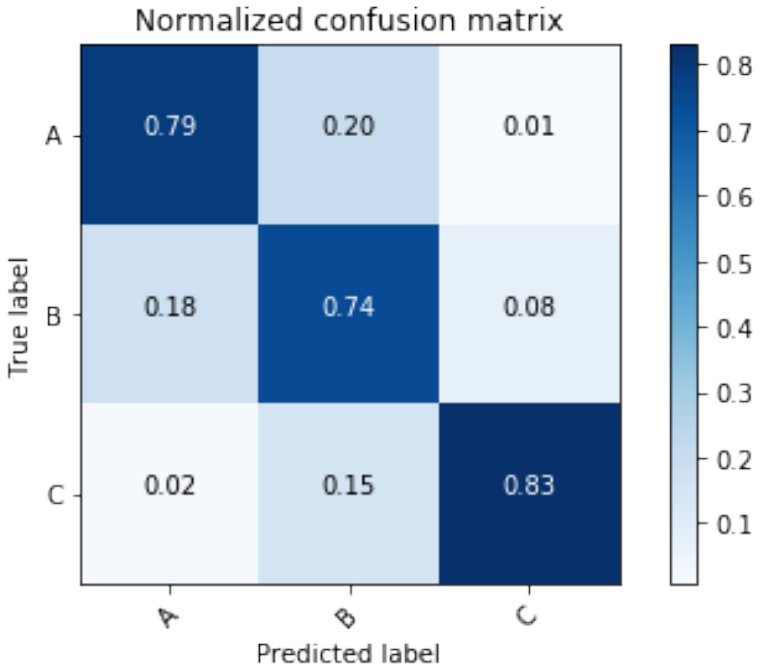

Figure 2.3: Confusion matrix illustrating the performance of LR-BoW-Word model after augmentation

2.6 Acknowledgment

This study has made use of the English Grammar Profile. This resource is based on extensive research using the Cambridge Learner Corpus and is part of the English Profile programme, which aims to provide evidence about language use that helps to produce better language teaching materials. See http://www.englishprofile.org for more information.

Chapter 3

Automatic Reading Comprehension

Abstract

Machine Comprehension has received great attention from the NLP community not only because solving it will bring us closer to understand humans better, but it also has significant applications in education and automated dialogue platforms. Recently the very task has become even more popular due to the release of SQUAD Competition (Rajpurkar et al., 2018). In this task, a machine learning model is expected to find an answer to a question from a passage, or abstain from answering if an answer is nonexistent. We propose a simple two-stage classification approach for this task. The first stage is training a logistic regression classifier to pick the sentence that is most likely has the answer to the question, or return null if an answer does not exist. After that, we train another logistic regression classifier to choose the right answer span from the best candidate sentence. Our results show this multi-stage approach achieves 0.71 score on the first stage, and 0.%% on the second stage.

3.1 Introduction

Machine Comprehension (MC) is the task of automatically answering comprehension questions based on information derived from short texts. While this task is relatively easy task for the humans to do, it is a hard task for the machine because it requires not only linguistic information, but also cognitive, world-knowledge skills as well. MC 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.

For this task, we will be using the second release of Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 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, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

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, meaning 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. Therefore, 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 need another classifier to find the constituent best represent the right answer span within the setence 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 this 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 RNN 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

vector 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 et al (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 (GloVe)@ and a pre-trained neural language modal ELMo @ 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.

While these models achieve astonishing results, they are needlessly complex in terms of architecture design and implementation, and very resource-intensive. This complexity results due the lack of linguistic assumptions that can arguably constrain and direct the problem. The baseline for the task is a simple logistic regression with a set of features, and I would argue that adding more feature could achieve good results at a fracture of cost and complexity.

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).

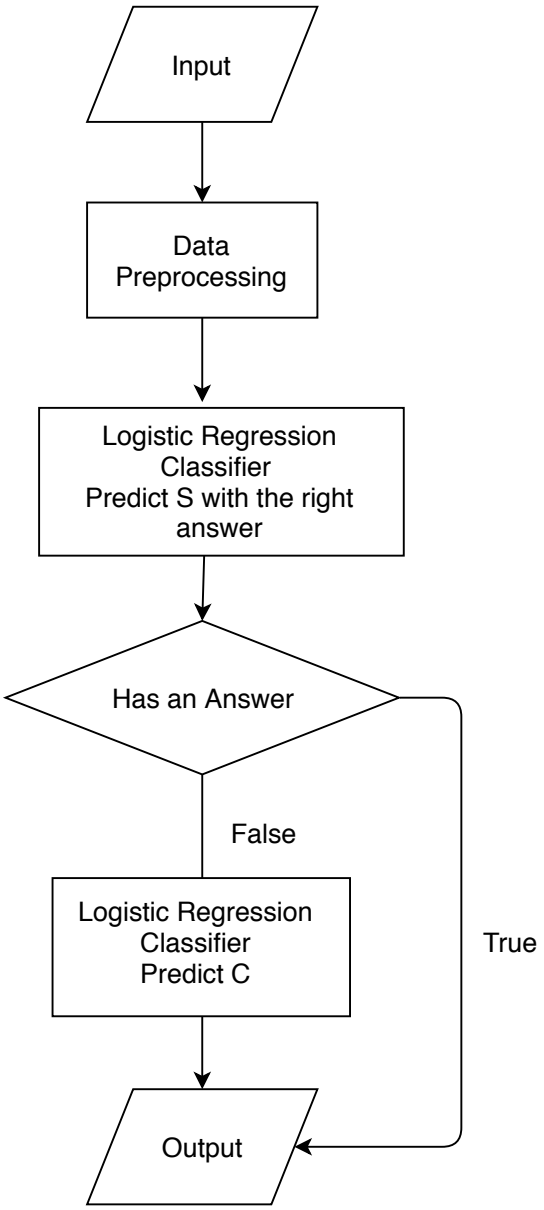

Figure 3.1: Flowchart illustrating the two-stage classification approach

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 i) shares more words with the question. ii) has high cosine similarity with the question. iii) 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 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.

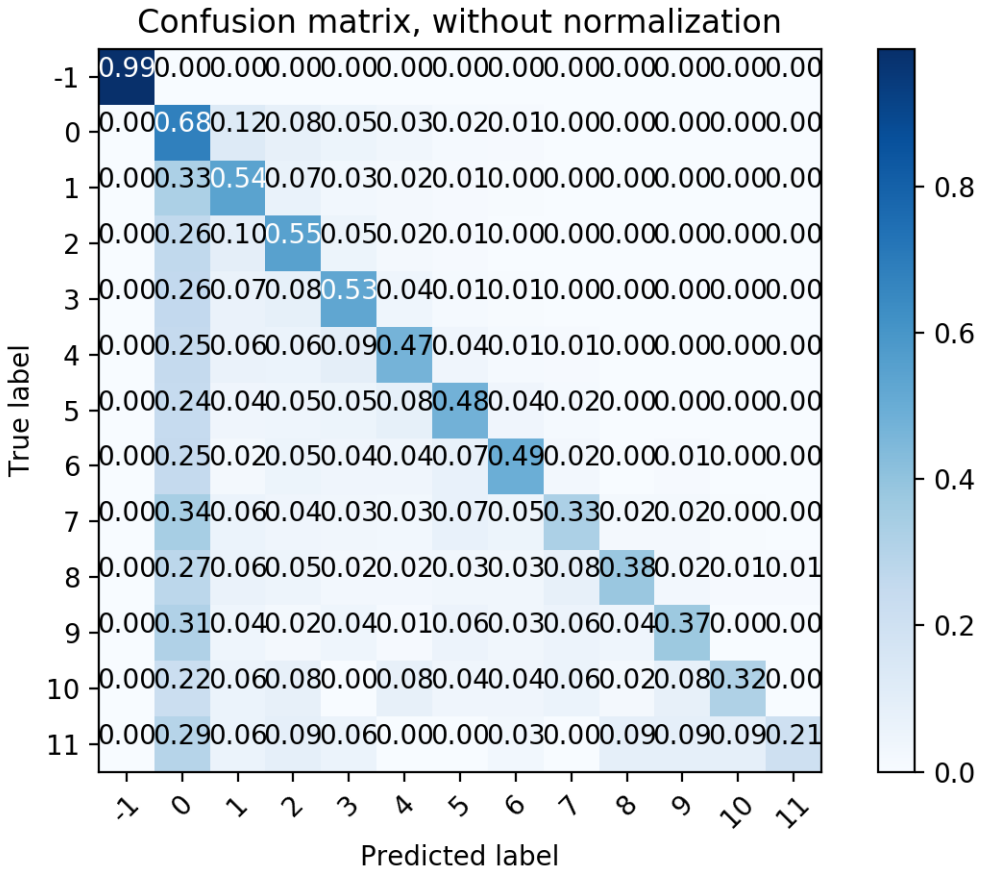

Figure 3.2: Confusion matrix shows which classes were predicted correctly

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, 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 the constituents. This results in around 45K data point. Next, we train a logistic regression classifier with

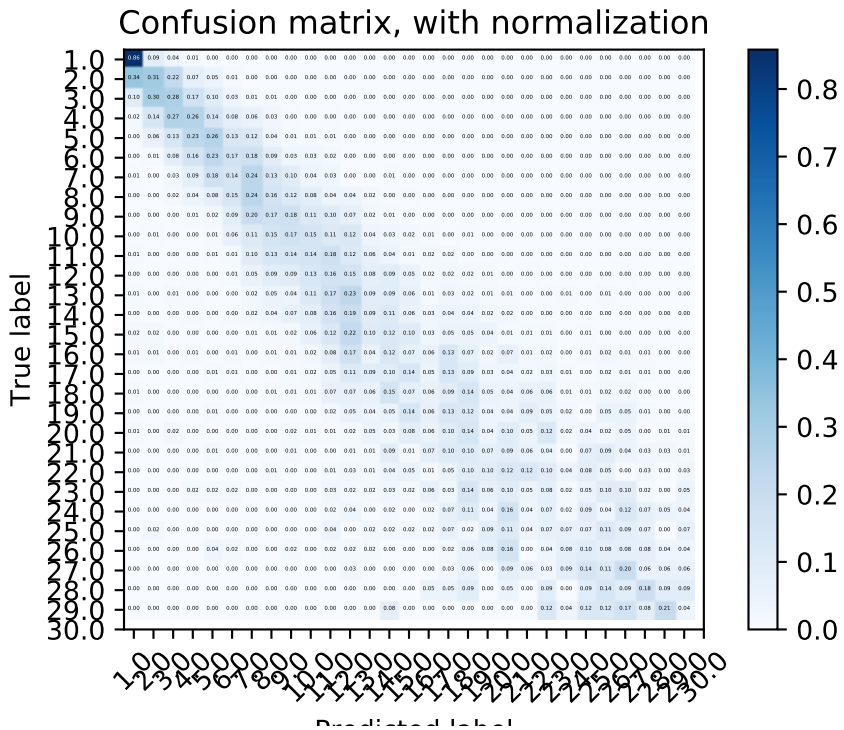

Figure 3.3: Confusion Matrix illustrating the first iteration of Stage2 LR.

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 im-

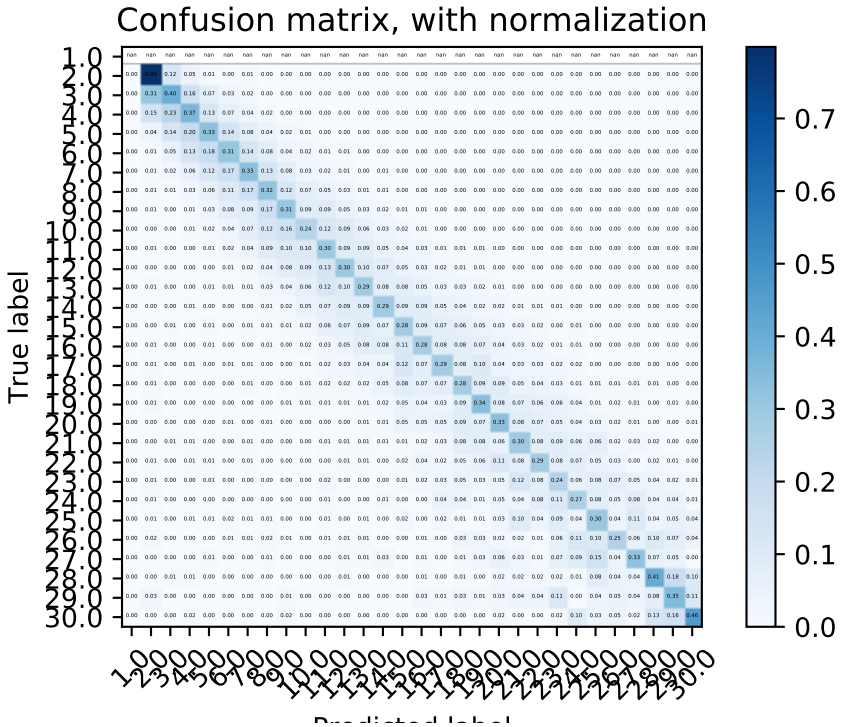

Figure 3.4: Confusion Matrix illustrating the Second iteration of Stage2 LR.

prove 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.