**Ancient Chinese Poem Generator**

Shengyu Chen 1568011188 [shengyuc@usc.edu](mailto:shengyuc@usc.edu)

Shuai Zhou 4264203725 [zhou990@usc.edu](mailto:zhou990@usc.edu)

Yizhao He 4256071445 [yizhaohe@usc.edu](mailto:yizhaohe@usc.edu)

Zichen Yang 8236922562 [zichenya@usc.edu](mailto:zichenya@usc.edu)

# **ABSTRACT**

Ancient Chinese poetry generation is a challenging and interesting task in natural language processing. In this paper, a multi-layer Recurrent Neural Network(RNN) with Long Short-Term Memory (LSTM) model is trained and tone pattern rule is applied to the model to generate ancient Chinese poem. Experiment results show that our model outperforms a SMT-based baseline model through manual evaluation methods.

# **INTRODUCTION**

Ancient Chinese poetry, as one of the most famous cultural heritage, has been applied widely to express author’s feelings, such as homesickness, friendship, etc. The golden age of ancient Chinese poetry was in Tang dynasty, which was the most prosperous dynasty in Chinese history.

There are some fixed rules for ancient Chinese poetry written during Tang dynasty. Basically, there are either four or eight sentences in a poem and five or seven Chinese characters in each sentence. In this paper, we will focus on the most popular poetry style, which consists of four sentences with five characters in each sentence. Of course, in order to sound comfortable when reading the poem, there are restrictions on rhyme, like Ping (the level tone) or Ze (the downward tone). Based on these requirements, we collected 16,000 poems written during Tang dynasty as our dataset.

With the development and rising popularity of the deep learning, poetry generation have been a hot topic. There are a lot of valuable resources and methods based on previous research[1-4]. These methods usually generate the first line by selecting one line from the dataset of poems according to the user’s writing intents, and the other three lines are generated based on the first line and the previous lines. Here, we use the similar way to generate our poem. However, the creativity in our project is that we integrate acrostic poem, which means besides the first sentence, we will give first character of the other three sentences to generate the remaining part of the poem. Brilliant poets sometimes use the acrostic poem to implicitly display their talents, express their view, satirize some people or phenomena, etc. Moreover, another essential feature of our project is that we strictly consider tone pattern for each character to keep the beauty of rhyme for whole poem.

The rest of this paper is organized as follows. In section 2, we specifically introduce our methods including data source and how to preprocess data, establish and train LSTM model, and evaluation process. Section 3 discuss the results of our model compared to a SMT-based baseline method. At last, section 4 discusses our project.

# **METHOD**

* 1. Materials

The source corpus of ancient Chinese poetry is collected from Jackie Gao's GitHub repository[5]. The whole source data consists approximately 55,000 Tang poems of 10,000 famous poets in total. For each poem, the basic information of author, paragraph and tone pattern is well structured in the corpus.

* 1. Procedure

Basically, to train our model, we first preprocess and clean the training data. Then a multi-layer LSTM is trained to generate Tang style Chinese poems. We apply tone pattern constraints on the generating procedure to make poems more human-written alike.

* + 1. *Data preprocessing and cleaning*

We store each poem as a dictionary (python data type) in which three keys represent paragraph, tone and title respectively. For tone representation, ‘p’ stands for the level tone (平) and ‘z’ stands for the downward tone (仄). Since we only need 5-character poems for training, we filter out all the poems whose format is not classified as ancient Chinese 5-character poems. For simplicity, we convert traditional Chinese to simplified Chinese, remove all the commas and periods and replace them with ‘$’ to represent end of sentence(<*EOS*>). To obtain high-quality data, we need to remove incorrect records and unnecessary symbols in the data set. In order to save as much efforts as possible and at the same time obtain larger training corpus, we come up with a filtering strategy. The strategy is to remove any poem whose contents are considered as incompletion or format is not correct and modify the poem which contains extra unrecognized characters. Before applying the strategy, we obtain around 28,000 poems from the source corpus. After applying, only 16,000 effective poems with correct tone and complete paragraph are selected for training.

* + 1. *Ancient Chinese poem generator*

Thanks to LSTM’s ability of modelling long-term dependencies [6], we use a rule-based multi-layer LSTM to generate the following sentences of the poem. The structure of this generator is shown in Fig.1. This neural network is basically a sequence-to-character model, which models probability of a character given the preceding *n-*character sequence:

We use one-hot encoding to represent each Chinese character so that the model can be a generative model which outputs the probabilities of all the candidates. This makes it easier to apply tone pattern constraints on the poem, which will be discussed later.

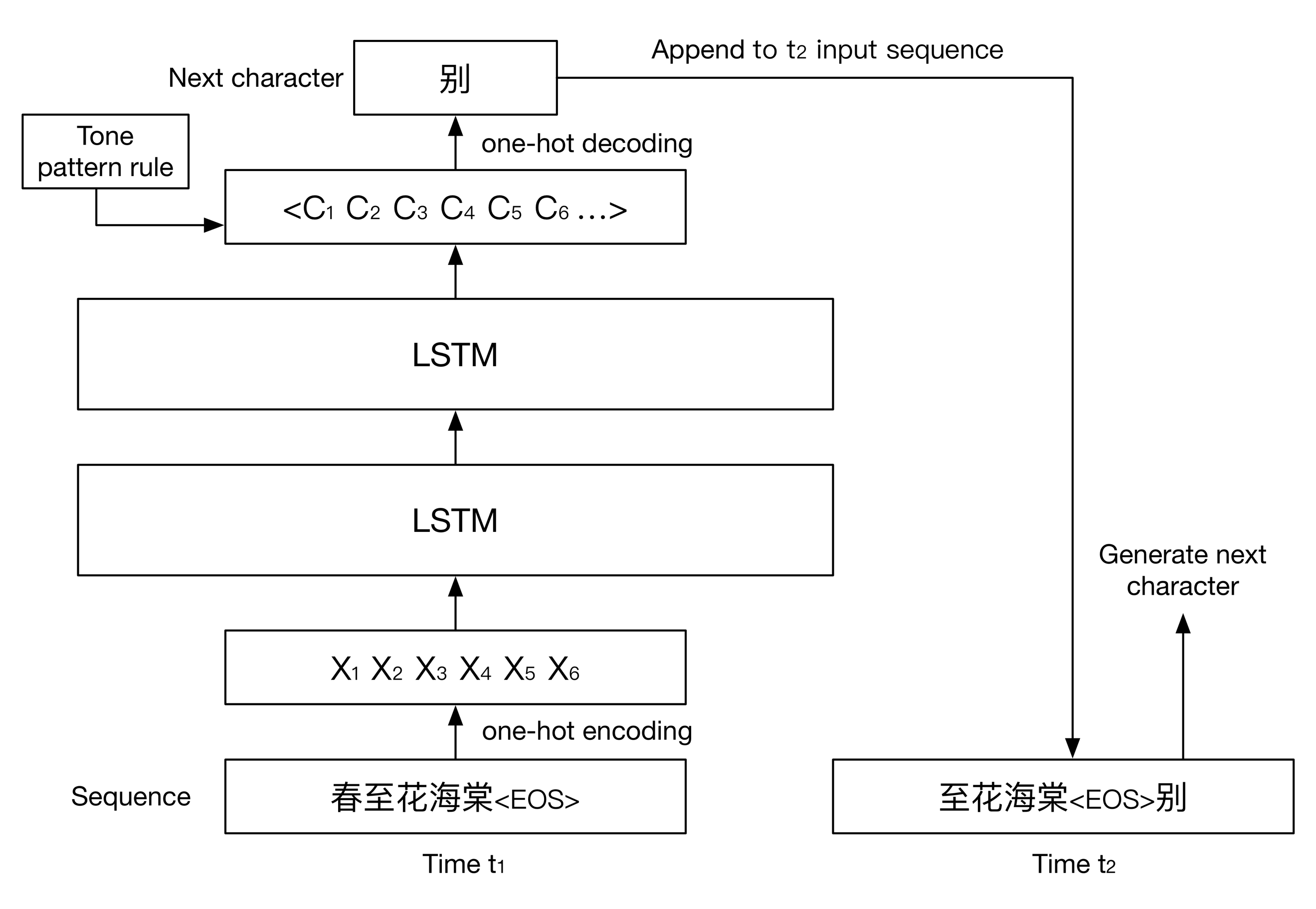


Fig. 1 Structure of following sentences generator with two layers LSTM

(Input sequence means “Spring comes, begonia blossoms”. Output character means “Depart”)

*Training*

All processed data containing about 16,000 5-character ancient Chinese poems written in Tang dynasty is used on training the model. We tried different number of layers and memory units to get the best result. Other configurations and hyper parameters are fixed: we use *softmax* as the activation function; loss is calculated through cross entropy; *Adam*[7] is used to optimize the learning rate; dropout is 0.2; sequence length is 6 (for better estimating each 5-character sentence since we also encode *<EOS>* as a character); batch size is 128.

*Generating*

To generate the following sentences given the first sentence which is also a 6-character sequence ended with *<EOS>*, we simply input the first sentence and use the most likely character as the next character and recursively generate the whole poem:

To make the poem more human-written alike, there are some rules we need to consider. First, we want the model learn to generate poem in rhyme, so we encode *<EOS>* to represent the end of sentence. However, *<EOS>* should never be output in the middle of each sentence, and must be output at the end. The other thing is we want the poems to be in correct tone patterns. There are four different tone pattern templates for 5-character ancient Chinese poem. Under this constraint, the generator will choose the most likely character that also follow the tone pattern rule.

* + 1. *Poem generator user interface*

In order to provide a way for normal users to access our poem generator and visualize poem generation results, we craft an elegant user interface.  Workflow of our poem generator are as follows: Firstly, gather user input, including initial characters for each sentence (optional) and the first sentence.  When the user clicks the submit button, a GET request will be sent to our server. A Python script will be invoked to load the model and vocabulary dictionary to generate the poem.  After receiving the JSON object that wrapping the result, our UI will display the poem sentence by sentence, word by word.

Regarding the technology stack, Express.js is used as the backend. The frontend part uses Semantic-UI for the layout and styling. Also, Vue.js is used for handling the frontend logic.

For live demo, you can visit our website: <http://poem.knotesapp.com/>



Fig. 2 User interface

* 1. Evaluation

*2.3.1 Evaluation Metrics*

Since accurate evaluation of automatically generated poetry is very difficult and the overlap-based automatic evaluation methods such as BLEU have been proven by Chia-Wei Liu[8] to be unrelated to human evaluation, we decide to perform human survey. We invite ten Chinese people who have comprehensive and authoritative knowledge of Chinese poetry. The experts rate the poems generated by our model using 1-10 scale with the higher score the closer to human written poems based on four evaluation standards in following general grading criterions.

General grading criterions:

(a) Poeticness: rhyme and tone requirements are followed

(b) Fluency: ideas succinctly stated, proper mechanics

(c) Meaningfulness: have meaning, function or purpose

(d) Coherence: structural, phonological, and semantic requirements are satisfied

*2.3.2 Baseline method*

We used a SMT-based model, a Chinese poetry generation method based on Statistical Machine Translation [9] as baseline and employed the same pre-processing method for this model. For SMT-based model, a poem is generated iteratively by “translating” the previous line into the next line.

*2.3.3* *Experiments*

In the first experiment, the first step is to select ten five-character Chinese quatrains from Tang dynasty which are not in our training corpus. Then generate ten five-character poems using our proposed method and baseline method with the first sentence from the ten selected poems as input.

In the second experiment, we do the same selection for test poems. Then generate the poems given the first sentence and first Chinese character of each sentence from the selected poems. This experiment is to test our acrostic feature.

In both experiments, we tried three different kind of parameter configurations of our proposed model. They vary in the number of memory units and number of LSTM layers.

For each survey, ten human evaluators rate ten sets of poems. Each set of poems contains 5 poems including one poem generated by SMT- based method, three poems generated by our proposed model trained with different parameters and also the original human written poem. The human evaluators have no clue about which model the poem is generated by or whether the poem is written by human poet. The final score will be the average score for all grading criterions. To eliminate bias, it's necessary to remove outliers that will significantly affect the final score.

1. **RESULTS**

In the following part, we use MLSTM-mmm-n to denote the *n*-layer LSTM method with *mmm* memory units. From the experiments results shown in Table 1 and Table 2, our proposed method MLSTM-500-3 which is trained with 500 memory units and 3 layers of LSTM performs the best among all the MLSTM methods. The reason of MLSTM-700-3 performs poor than the MLSTM-500-3 version may be too many memory units cause the network hard to converge in the training.

According to the results shown in Table 1, MLSTM-500-3 outperforms significantly beyond SMT-based method on poeticness and coherence. This demonstrates that our model precisely captures the tone pattern, the beauty of poetry. Moreover, the score indicates that our model is capable of generating more meaningful sentences than SMT-based method, because our model captures long term dependencies.

In the second experiment, we also include the first character of the selected poem as input. From the results shown in Table 2, we can tell that the poem generated by baseline method and our model are rated with averagely higher scores than experiment 1. This is because the input provides more information for the model, so that the output is more related to the original poem.

**Table 1**. Human evaluation results for experiment 1, poem generating given first sentence

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Models** | **Poeticness** | **Fluency** | **Meaningfulness** | **Coherence** | **Average** |
| SMT | 5.25 | 5.40 | 4.22 | 5.30 | 5.04 |
| MLSTM-500-2 | 5.44 | 5.60 | 4.40 | 5.50 | 5.24 |
| MLSTM-500-3 | 5.56 | 5.70 | 4.80 | 5.67 | 5.43 |
| MLSTM-700-3 | 5.33 | 5.50 | 4.33 | 5.40 | 5.14 |
| Human Poem | 8.70 | 8.22 | 8.30 | 8.55 | 8.44 |

**Table 2**. Human evaluation results for experiment 2, poem generating given first sentence and initial character of each sentence

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Models** | **Poeticness** | **Fluency** | **Meaningfulness** | **Coherence** | **Average** |
| SMT | 6.11 | 5.44 | 4.50 | 5.88 | 5.48 |
| MLSTM-500-2 | 6.40 | 5.70 | 4.80 | 6.30 | 5.80 |
| MLSTM-500-3 | 6.64 | 5.90 | 5.00 | 6.40 | 5.99 |
| MLSTM-700-3 | 6.20 | 5.50 | 4.90 | 6.00 | 5.65 |
| Human Poem | 9.20 | 8.80 | 8.44 | 8.33 | 8.69 |

Although the poems generated by our model are very fluent and coherent, we must admit that it’s still far away from human poet. Since our model generates each character based on preceding sequence, the meaning and key idea of next character is purely depending on the previous ones. If we solely look at one sentence of a generated poem, it maps the relation of previous and following lines. As we combine all sentences together, we find that it lacks expression of deep emotion, and insights of inner life of Chinese poets.

1. **DISCUSSION**

In this paper, we propose new method based on a multi-layer LSTM model to generate ancient Chinese poetry. Tone pattern rule is applied to the model to generate rhyme poems. In the experiments, we compare our model with a SMT-based method through human evaluations. The results show that our model outperforms the baseline method and our model is capable of generating Tang style poems with certain poeticness, fluency, meaningfulness and coherence.

There are still some issues we need to consider to improve our model. First, due to the limitation of our time and computation ability, the parameters of the model may be configured more precisely and more appropriate to the size of the training data so that the model will generate better poems. This may need several attempts with different parameter setting. Second, our model needs first sentence of the poem as the input to generate the rest. It is more convenient for the user to generate the whole poem with giving only a few key words. To implement this may need a separate model to generate the first sentence according to the input key words.

**REFERENCES**

[1] Zhang, Xingxing, and Mirella Lapata. "Chinese Poetry Generation with Recurrent Neural Networks." *EMNLP*. (2014).

[2] Yi, Xiaoyuan, Ruoyu Li, and Maosong Sun. "Generating chinese classical poems with rnn encoder-decoder." *arXiv preprint arXiv*:1604.01537 (2016).

[3] Wang, Qixin, et al. "Chinese song iambics generation with neural attention-based model." *arXiv preprint arXiv*:1604.06274 (2016).

[4] Wang, Zhe, et al. "Chinese poetry generation with planning based neural network." *arXiv preprint arXiv*:1610.09889 (2016).

[5] JackeyGao. "JackeyGao/chinese-poetry." *GitHub. N.p*., https://github.com/jackeyGao/chinese-poetry. Web. 27 Apr. 2017.

[6] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8: 1735-1780 (1997).

[7] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv*:1412.6980 (2014).

[8] Liu, Chia-Wei, et al. "How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation." *arXiv preprint arXiv*:1603.08023 (2016).

[9] He, Jing, Ming Zhou, and Long Jiang. "Generating Chinese Classical Poems with Statistical Machine Translation Models." *AAAI*. (2012).

**DIVISION OF LABOR**

**WORD COUNT**