Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

T.U.I.A

Aprendizaje Automático II

Trabajo Práctico 2

Redes Recurrentes

2024

Autor/es:

Grupo Nº	
Nombre y Apellido	Nº de Legajo
Mateo Gravi Fiorino	g-5845/9
Gauto Lucas	

Corrigió	Calificación

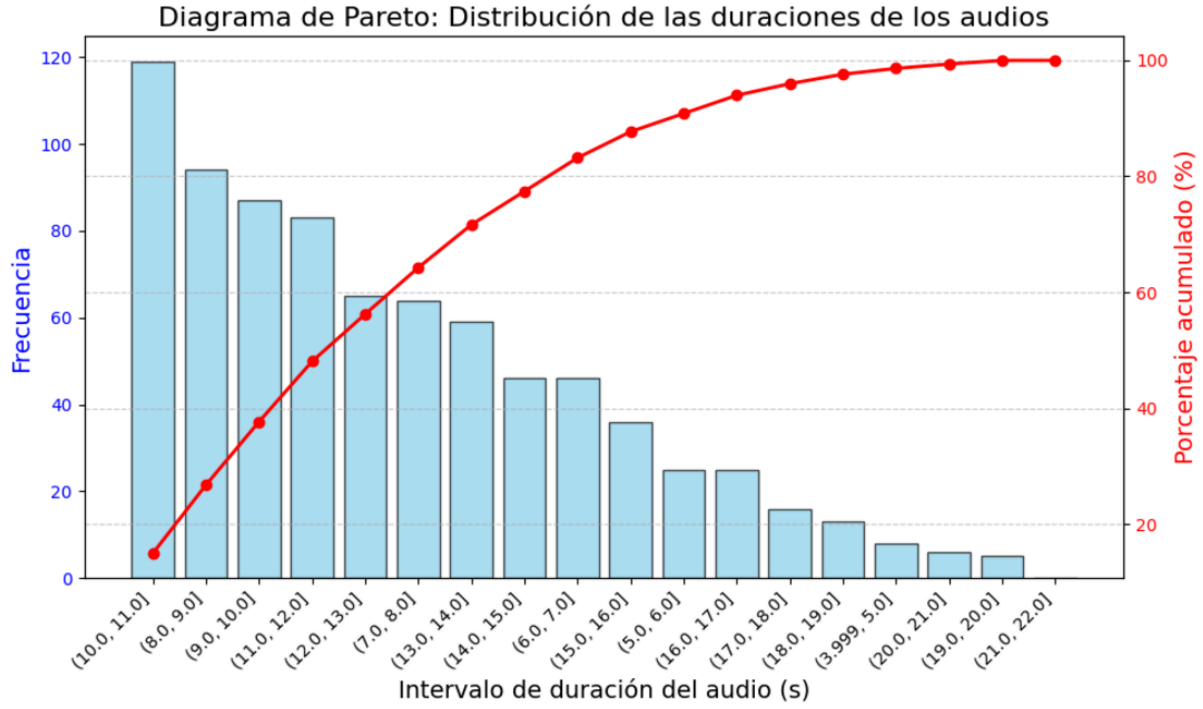
Problema 1

Se requería entrenar dos redes neuronales (una recurrente y una convolucional) para la detección de idiomas en una de cuatro clases: español, francés, inglés y japonés. Para ello se utilizaron los audios del dataset **xtreme_s**, concretamente un subconjunto de cada idioma.

Análisis y Pre-procesamiento

Duración de los audios

Con el análisis de los audios descubrimos que existen duraciones muy diferenciadas en ellos, algunos teniendo una duración de cinco segundos o menos y otros llegando a los veintidós segundos. Como nuestro enfoque fue "llenar" los audios cortos con silencio para que todos tengan una longitud similar, decidimos quitar del dataset aquellos audios con una duración demasiado alta o demasiado baja, quedándonos solamente con audios de longitudes de seis segundos a quince segundos (donde se concentraban la mayoría de los datos).



En el diagrama de Pareto podemos ver cómo más del 80% de los datos se encuentran en este intervalo.

Nuestro enfoque se basó en el utilizado en clases: convertir las ondas de audio en espectrogramas con el fin de capturar las distintas variaciones a lo largo del tiempo (útil para la red recurrente) y una matriz bidimensional a la cual extraerle características (útil para la red convolucional).

Tras realizar algunas pruebas, descubrimos que con longitudes de audio menores la red tendría más ejemplos y, además, aumentaría su velocidad de entrenamiento. Por ello, partimos los audios en longitudes más pequeñas.

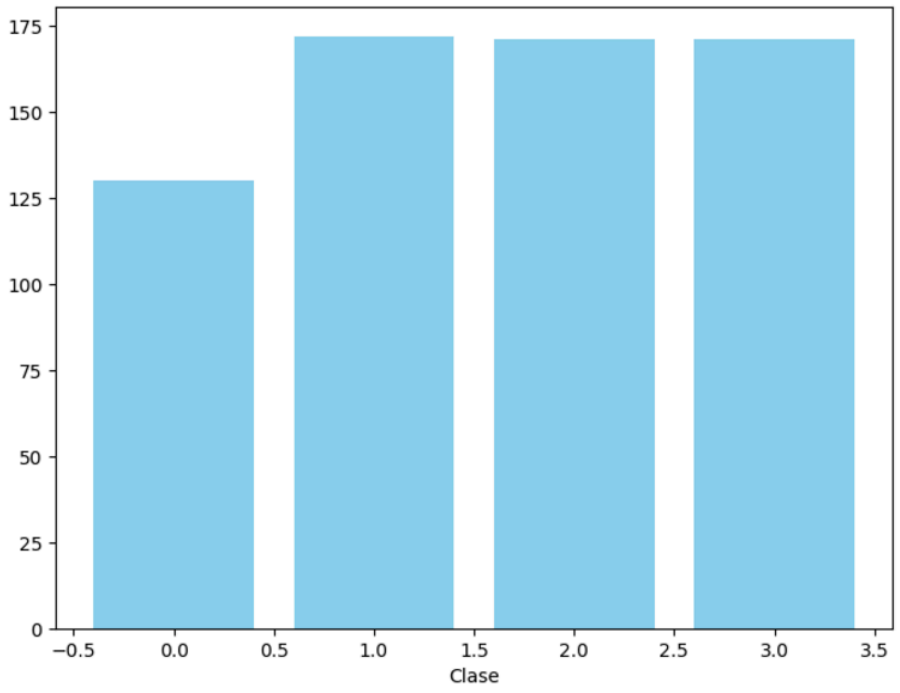
Ruido

Luego de realizar algunas pruebas de entrenamiento de redes recurrentes, descubrimos que el rendimiento era mucho menor al esperado, con un accuracy que no supera el 40%. Por ello, decidimos volver al análisis de los audios.

Empíricamente encontramos algunos audios con una cantidad demasiado alta de ruido, y consideramos que era mejor conservar únicamente audios con bajo ruido que permita la identificación del idioma, así que decidimos eliminar los audios con demasiado ruido.

Para ello utilizamos una herramienta de la librería de manipulación de audio de python *librosa*, concretamente la herramienta calcula el contraste espectral, el cual podríamos decir que es un indicador de la cantidad de ruido que hay en una onda. De esta manera, filtramos los audios que observaban un contraste espectral promedio mayor a 12 (debajo de este umbral los audios eran prácticamente puro ruido) y nos quedamos con aquellos de menor ruido.

Como resultado de este filtrado, nuestro dataset se desbalanceó ligeramente. Sin embargo, consideramos que este desbalance no era pronunciado y decidimos seguir. Posteriormente comprobaremos al realizar pruebas que la clase 0, Español, tiene dificultades para ser predicha correctamente, lo que atribuimos a esta particularidad del dataset adaptado.



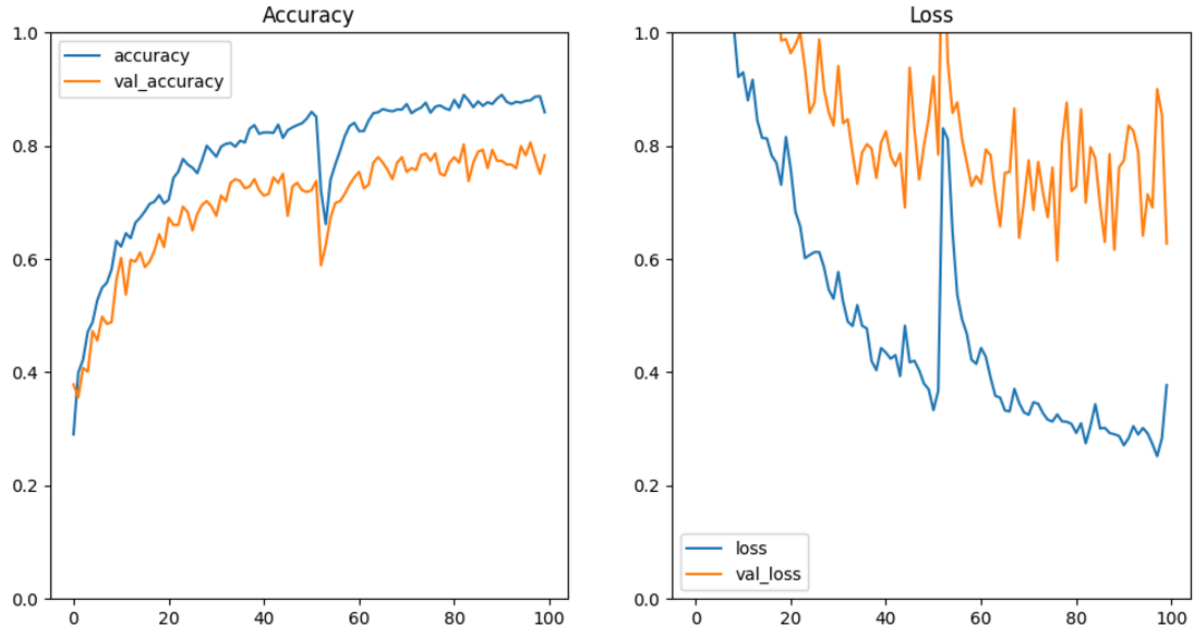
Red Recurrente

La arquitectura de nuestra mejor red recurrente es la siguiente:

Layer (type)	Output Shape	Param #
conv1d_2 (Conv1D)	(None, 498, 64)	24,832
max_pooling1d_2 (MaxPooling1D)	(None, 166, 64)	0
dropout_3 (Dropout)	(None, 166, 64)	0
conv1d_3 (Conv1D)	(None, 164, 128)	24,704
max_pooling1d_3 (MaxPooling1D)	(None, 54, 128)	0
dropout_4 (Dropout)	(None, 54, 128)	0
lstm_1 (LSTM)	(None, 64)	49,408
dropout_5 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 4)	132

Total params: 101,156 (395.14 KB)
Trainable params: 101,156 (395.14 KB)
Non-trainable params: 0 (0.00 B)

Y sus resultados fueron:



En test, el accuracy fue del **80%**.

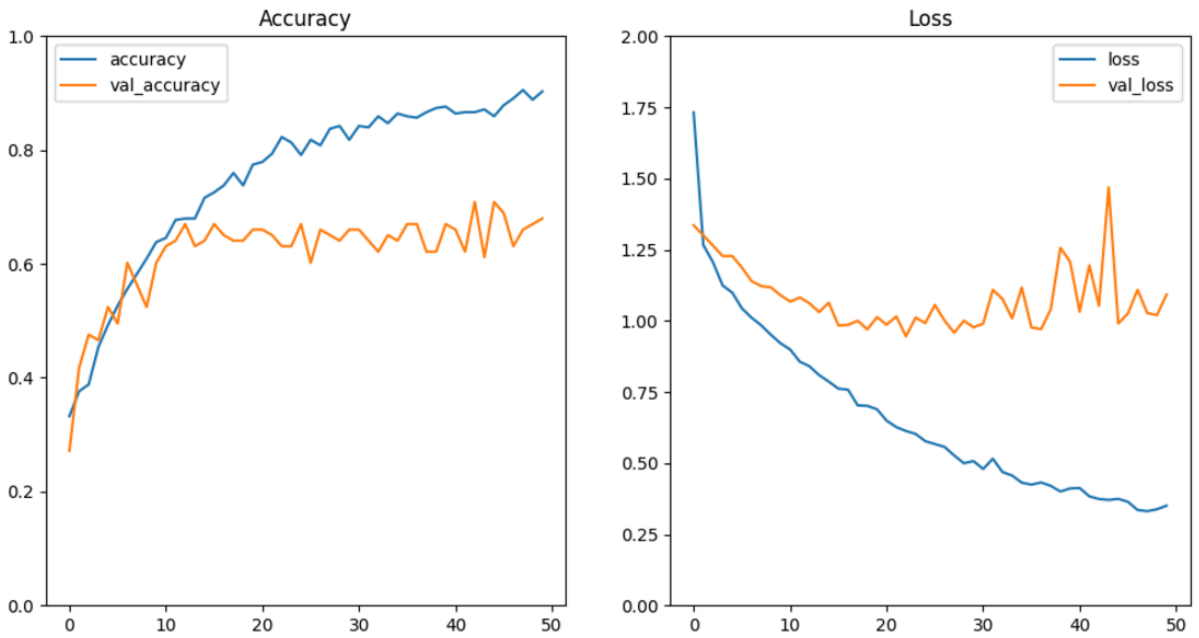
Red Convolucional

En el caso de la red convolucional, tuvimos un rendimiento inferior. La arquitectura utilizada fue:

Layer (type)	Output Shape	Param #
resizing (Resizing)	(None, 128, 128, 1)	0
normalization (Normalization)	(None, 128, 128, 1)	3
conv2d (Conv2D)	(None, 126, 126, 64)	640
max_pooling2d (MaxPooling2D)	(None, 42, 42, 64)	0
dropout_6 (Dropout)	(None, 42, 42, 64)	0
conv2d_1 (Conv2D)	(None, 40, 40, 128)	73,856
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 128)	0
dropout_7 (Dropout)	(None, 13, 13, 128)	0
flatten (Flatten)	(None, 21632)	0
dense_4 (Dense)	(None, 32)	692,256
dense_5 (Dense)	(None, 4)	132

Total params: 766,887 (2.93 MB)
Trainable params: 766,884 (2.93 MB)
Non-trainable params: 3 (16.00 B)

Y sus resultados:



Y el accuracy alcanzado en test fue del **64%**.

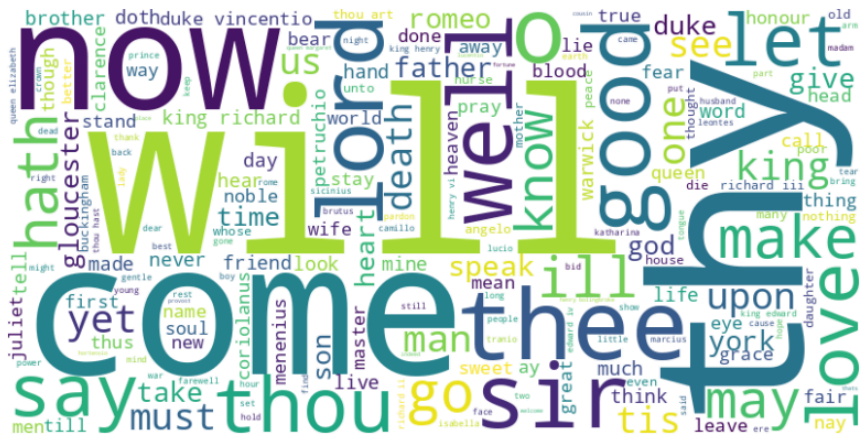
Conclusiones del problema 1

Observamos en ambas redes un sobreajuste a los datos de entrenamiento, nuestros modelos no logran generalizar correctamente (especialmente en el modelo convolucional) y, luego de realizar varias pruebas, consideramos que la razón podría deberse a la cantidad de datos con los que fueron entrenadas (como ya se mencionó, un subconjunto mínimo del dataset real).

Problema 2

En el siguiente problema, se presenta un conjunto de datos correspondientes a escritos de Shakespeare. El objetivo del problema es crear un modelo capaz de generar texto con dialecto de época y escritura en verso y prosa.
Utilizando el dataset construido, el objetivo es construir modelos de generación de texto utilizando redes neuronales que puedan generar texto con dialecto de época y escritura en verso y prosa.

Luego de preparar el entorno, leer los datos y hacer un pequeño análisis exploratorio,



Pasamos a la vectorización del texto para su posterior uso en el entrenamiento de un modelo caracter a caracter.Se crean los batches de entrenamiento y el dataset.

Construcción del modelo

Definimos el modelo como una subclase de **keras.model**. Este modelo tiene tres capas:

1. La **capa de entrada**, una lookup entrenable que asignará cada ID de caracter a un vector con dimensiones preestablecidas.
2. Una **capa recurrente** de tamaño preestablecido.
3. La **capa de salida** que genera un logit para cada caracter de vocabulario.

Funcionamiento

Para cada caracter el modelo calcula su embedding, corre la capa GRU y aplica la capa densa para generar los logits produciendo las probabilidades del siguiente caracter.

Entrenamiento

La funcion de pérdida estándar `tf.keras.losses.sparse_categorical_crossentropy` funciona en este caso porque se aplica en la última dimensión de las predicciones.
Cargamos los checkpoints del modelo para ir guardando el mismo durante el entrenamiento.

Al finalizar el entrenamiento obtenemos:

Epoch 50/50
172/172 [==============================] - 11s 54ms/step - **loss: 0.4157**

Generación de texto

Al generar texto con una clase preestablecida, vemos resultados con sentido, oraciones formadas, ‘,’ y signos de exclamación. Parece una buena primera impresión.:

Sheakspeares notive but boundly lists
And pluck'd up with procereings, that no more comes
To have some part end or solemn hands.

CLIFFORD:
My lady's dead! how he is in!
Here comes the fault to you. Lord Angelo's womb,
That rage, for quarrel of a soldier arm;
Hold-morrow, as I am now chaste, and Tranio.

TRANIO:
Master, fool! what, in this viewly die?

DUKE OF YORK:
Please you, sir, so hath a haste hours here let God his grace.

A pack had more blazing incornatement
Of my heart's deputy, shall kingly give thee like requian.

KING RICHARD II:
Now put it, I say; thou shouldst be man,
Like out again with rude and lent mischance
In the shepherd's nose, and let the coronation.
Thy father, by God-grandon! Who's there? O woful, we have spoke,
a bawd of health, and give him chair, and shrift
His horse in blood, but in my speeches
Were neared to: for a thousand crowns!
I prophys are made to be his dishonour'd,--
Whom Warwick was my father's fault, if this Marcius
Where were unearded! make not man he is.

Distintas temperaturas

Con la temperatura podemos medir la flexibilidad y la robustez. Vemos que para temperaturas bajas genera un texto más repetitivo, robusto. Y a medida que aumenta la temperatura vemos más *creatividad* y flexibilidad a la hora de generar texto

Generando texto con temperatura: 0.5

Sheakspeares notice, how his happier kind
That breed by the immodable of your brother's part,
And, by the stranger pluck in hope, which will in time
Break ope the devil and deliver you
Of some short of my steed, whose tender lady
Hath set the birds of patience and in one army
Beggars, and gave them for the fau

Generando texto con temperatura: 1.0

Sheakspeares notice of his hands and that to the
out: mine honest cheek the man, that thou art old,
which now you seem'd ladies with a horse-frought of

instered.

Second Citizen:
What will mourn fight, and yet my brother's blood,
He's base the fearful King of France:
Let us be blessed new together, but thou
ca

Generando texto con temperatura: 1.5

Sheakspeares notice of Venician grave?

DUCHESS OF YORK:
Nay, if thy time would follow me.

GLOUCESTER:
Come you, pray, let us hear these years
a most hell; and such in justice, yield,
My soil thou art, an envious that I have.

RICHARD:
You're a brave fellow; I Harry Bolingbroke his trimb in a
nughtee of our c

Modelo palabra a palabra

En esencia es muy parecido. Se realizan algunos ajustes, cómo la **vectorización del texto por palabra** y no por caracter y la clase de Generación de texto de igual forma.

Creamos el dataset de igual forma y entrenamos el modelo.

Entrenamiento

Al finalizar el entrenamiento obtenemos una **loss** del **0.18**

Epoch 45/45

126/126 [==============================] - 17s 125ms/step - loss: 0.1819

El modelo resultó con un total de 58M de parámetros aproximadamente.

Al probar la generación de texto con este modelo obtuvimos resultados raros y extraños:

Generando texto con temperatura: 0.5

To be or not to
betototototototototototototototototototototototototototototototototototototototototototototototototototototototototot
otototototototototototototototototototototototototototototototototototototototototototototototototototototototo

Generando texto con temperatura: 1.0

To be or not to
bechildrentheeshesototototototototototototototototototototototototototototototototototototototototototototo
tototototototototototototototototototototototototototototototototototototototototototototototototototototototototototot
oto

Generando texto con temperatura: 1.5

To be or not to bethatmysea,
ThanhouseofHarry,conversationobey:
ThytownquicklyitfullIherslainusehisroyaldaughter,
Andguiltheryournumbers,father'sbosoms!Gloucester?
Now,outusehaddarestamongstho!grassyourthee,me,
AndasblessinghadyouyouheavybreathedasLordheaven!theeeverlittleshock
AthousethankashenotyouyouandyouearsyouyouityousoonEDWARDbe;
Forwillaswillwillofwillbelievewillwillwillfromwillmostare,theetheethatwillforwillwillwillwillwilldail
ylayyouthumb,youwillwillcupthee

Debido a estos resultados, decidimos reconstruir el modelo y aplicar una capa de LSTM. Un modelo de red neuronal recurrente (RNN) con capas LSTM para la predicción de texto a partir de secuencias de palabras.

1. División de Entrada y Salida:

- **Función split_input_target_word(chunk)**:
  Esta función toma una secuencia de palabras (un "chunk") y la divide en dos partes:
  - **Entrada**: La secuencia original, excluyendo el último elemento (chunk[:-1]).
  - **Salida**: La secuencia desplazada, excluyendo el primer elemento (chunk[1:]). Esto es necesario para el entrenamiento de modelos de lenguaje, donde el modelo debe predecir la siguiente palabra (salida) dada una secuencia anterior (entrada).

2. Preparación del Dataset:

- **Mapeo de Datos**:
  Utilizando la función split_input_target_word, el dataset de palabras (sequences_word) se mapea correctamente para que cada secuencia de entrada y salida esté alineada.
- **Configuración de Batches**:
  El dataset se "mezcla" (shuffle) para evitar sesgos durante el entrenamiento y luego se divide en batches de tamaño 64 (BATCH_SIZE). La opción drop_remainder=True asegura que solo se utilicen batches completos.

3. Definición del Modelo:

- **Modelo Secuencial con LSTM**:
  El modelo está compuesto por una serie de capas que incluyen:
  - **Capa de Embedding**: Convierte los índices de palabras en vectores de baja dimensión.
  - **Capas LSTM**: Se utilizan dos capas LSTM para capturar dependencias temporales en las secuencias de palabras. Se ha utilizado la opción return_sequences=True para que cada capa LSTM devuelva una secuencia completa de salidas, lo cual es útil cuando se tiene una salida secuencial (como en la predicción de palabras).
  - **Capa densa**: Con una activación softmax, esta capa genera una distribución de probabilidades sobre el vocabulario para cada paso de la secuencia.

4. Compilación del Modelo:

- **Función de Pérdida**: Se utiliza sparse_categorical_crossentropy como función de pérdida, que es adecuada cuando las etiquetas son índices de clase.
- **Optimizador**: Se usa el optimizador adam, que es popular por su eficiencia y adaptabilidad en problemas de aprendizaje profundo.
- **Métricas**: La métrica de accuracy permite monitorear la precisión del modelo durante el entrenamiento.

5. Entrenamiento del Modelo:

- El modelo se entrena durante 50 épocas (epochs=50) utilizando el dataset preparado.
- Durante el entrenamiento, se observa que la **pérdida** (loss) disminuye y la **precisión** (accuracy) aumenta, lo que indica que el modelo está aprendiendo a predecir las secuencias de palabras correctamente.

6. Resultados del Entrenamiento:

- **Evolución de la Pérdida y Precisión**:
  Durante las últimas épocas, el modelo muestra una mejora consistente, con la pérdida cayendo de 0.5508 a 0.4293 y la precisión aumentando del 90.15% al 92.62%, lo que sugiere una mejora en la capacidad del modelo para predecir las secuencias de palabras de forma precisa.

Luego, probamos la generación de texto con diversas temperaturas y observamos una gran mejora en la generación de texto.

Temperatura 1.0

To be or not to be opinion!
Alack, pent-up fourteen;
That degree,
Or by;
Having which
That yet:
But lovely shame.

CAPULET:
Go recreant,
To teeth, weariest mayest past.

KING word:
and heaviness
And old.

Third school'd
In elder, guess, city,
In blood
That subject, see'st wrath time fiery-red give;
And time?
If permit
A must
know wind.
These news.

GRUMIO:
First, I
know penitent
king, decked mocker! land:
The hold
In disposed
Ere name's There's favours
That bawd.

LUCIO:
Well, cook?
How we
shall three-man-song-men extremes.

KING LAURENCE:
Romeo company
Than

Temperatura 0.5

To be or not to be whit: eternal reasonable,
To repulse die,
And triumph comfort! IV:
We cull'd yourself?

PETRUCHIO:
Tedious litter part;
Lend court
Whiles death--
Shall leave
to justs good?

KING pray.
Whereto voluptuously ere't yourself storm,
While best.

MERCUTIO:
Help great.
DUKE say.
First, raying, What! died, although Clarence,
With pleasures; command.

ARIEL:
To conserves answer.

LEONTES:
O kiss.

PETRUCHIO:
Why, must: knew'st forth,
Of plunge organs
Of so-forth:' watch?
Sirrah, tool! done,
His days,
I incensed, rage, becomes endure

Your general,

Here

Temperatura 1.5

To be or not to be tears?
Evermore branches
We Plantagenet
Led territories,
And escape She thwack
repent trodden powers;
perform'd.

A old
man will?

POLIXENES:
I madman's understand.
Thine yourself's Senator:
Behold alike.

For

First all
the slain transported wept
majesty!

ANTONIO:
Long pray;
Or Thyself 'only' temper;
purgation.
Produce crown?

QUEEN here.

ROMEO:
I feast?
Or come?

Both:
To, road.
Upon's inforced Cominius
yourself?

PETRUCHIO:
Tedious commits lived,
Save tape, Oxfordshire play,
Alone, death.

DUKE successfully.
My officer:
Correction footman them.
Well, why,
thou recompense:
My growth
The either