BLM0334 Algoritma Analizi ve Tasarımı Projesi

2023-2024 Güz Yarıyılı

LSTM ALGORİTMASI İLE METİN OLUŞTURMA

Not: Uygulama kısmımız 10 mb sınırını aştığından dolayı interete yükleyip link olarak atıyoruz

Drive link: https://drive.google.com/file/d/1S_69oGXrVvN2iXZYQtNdNMM9rTVz7ZZZ/view?usp=sharing

Github Link: https://github.com/wassapman/LSTM

Kullanılacak Algoritma: Doğal Dil İşleme Algoritması (LSTM)
Problem: Manuel metin oluşturmanın zaman alıcı ve yoğun bir süreç olması. Kullanıcılar tarafından yapılan manuel içerik üretimi, büyük çaba ve zaman gerektirirken, hızlı ve etkili bir içerik üretme ihtiyacı giderek artmaktadır.

Genel Proje Açıklaması

Projemiz Doğal Dil İşleme algoritması olan LSTM algoritmasını kullanarak, makine öğrenmesi sayesinde rastgele Türkçe kelime ve cümle üretebilmektedir. Projemizde LSTM algoritması modelinin iç yapısı detaylı şekilde açıklanıp kullanılmaktadır ve hazır model ile manuel hazırlanmış model çalıştırılıp doğruluk, kayıp ve karmaşıklık gibi değerleri karşılaştırılmaktadır. 2 adet veri seti kullanılmıştır. İlk veri seti Oğuz Atay'ın Tutunamayanlar kitabından alınmış olup toplam 963 cümle bulunmaktadır. İkinci veri setinde 5749 cümle bulunmaktadır.

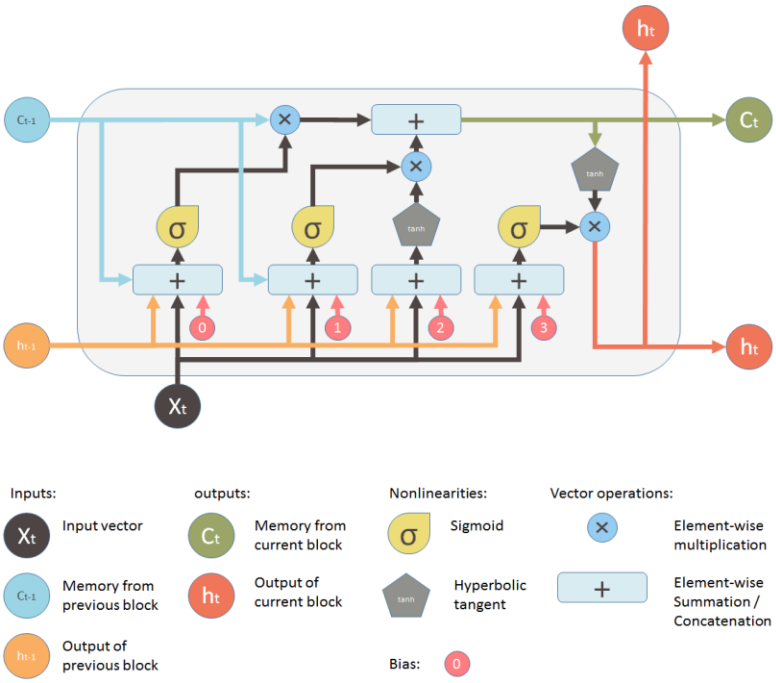
Proje Literatürü

Tekrarlayan Sinir Ağları (RNN) ile Sıradan Yapay Sinir Ağları (ANN) arasındaki temel fark, RNN'nin dizi verilerini daha etkili bir şekilde işleyebilme yeteneğidir. Dilin doğasında, bir cümle içindeki kelimeler arasında belirli bir bağlam ve korelasyon bulunabilir. RNN, bu tür bağlamları anlamak ve işlemek için özel olarak tasarlanmıştır, bu da doğal dil işleme görevlerinde etkili olmasını sağlar. Gradyanın rolü, sinir ağının ağırlık değerlerini güncellemektir. Ancak, ağırlıkların çok küçük olması durumunda gradyan kaybolur ve öğrenme durur. Tam tersi durumda, ağırlıkların çok büyük olması gradyan patlamasına yol açar. RNN'nin bu gradyan sorunları, uzun vadeli bağlantıları koruma ve kısa vadeli bağlantılardan daha etkili öğrenme yapabilen özel tasarlanmış modeller olan Uzun Kısa Süreli Bellek Ağları (LSTM) ve Kapı Tekrarlayan Birim Ağları (GRU) ile çözülmüştür. [1]

LSTM (Uzun Kısa Vadeli Bellek), bir tekrarlayan sinir ağı (RNN) mimarisidir ve normal sinir ağı mimarilerinden farklı olarak değerleri belirsiz aralıklar boyunca hatırlayabilme özelliğine sahiptir. LSTM, bilinmeyen süreli zaman gecikmeleri olan zaman serilerini sınıflandırmak, işlemek ve tahmin etmek için uygundur. LSTM'nin diğer RNN'ler, gizli Markov modelleri ve sıra öğrenme yöntemlerine göre avantajları arasında boşluk uzunluğuna karşı duyarsızlık bulunur. RNN'den farklı olarak LSTM, hücre durumu adı verilen uzun vadeli belleğe sahiptir. LSTM hücresi, unutma kapısı (Forget Gate) ve giriş modülasyon kapısı (Input Gate) tarafından kontrol edilen hücre durumunu kullanarak bilgiyi saklar ve unutur. Hücre durumu, önceki durumu unutma kapısıyla günceller ve giriş kapıları tarafından yeni bilgiler eklenir. Sigmoid ve tanh aktivasyon fonksiyonları, hangi bilginin hücre durumunda saklanacağını ve unutulacağını belirler. LSTM'nin çıkış kapısı, hangi bilginin bir sonraki gizli duruma gideceğini belirler. Bu sayede LSTM, uzun vadeli bağımlılıkları başarıyla işleyebilir.[2]

LSTM, dört sinir ağı ve bir dizi hafıza bloğundan oluşan hücrelerin bağlı olduğu bir zincir yapısına sahiptir. Geleneksel bir LSTM birimi, bir hücre, bir giriş kapısı (Input Gate), bir çıkış kapısı (Output Gate) ve bir unutma kapısı (Forget Gate) içerir. Bilginin hücreye girişi ve çıkışı, üç kapı tarafından kontrol edilir ve hücre, değerleri belirsiz zaman aralıkları boyunca hatırlar. Giriş Kapısı: Hücre durumunu güncellemek için kullanılır. Önceki gizli durum ve mevcut girişi bir sigmoid fonksiyonundan geçirir. Bu, hangi değerlerin güncelleneceğine karar verir (0 önemsiz, 1 önemli). Ayrıca, gizli durumu ve girişi tanh fonksiyonuna sokarak ağın düzenlenmesine yardımcı olacak şekilde değerleri sıkıştırır. tanh çıkışıyla sigmoid çıkışını çarparak hangi bilgilerin önemli olduğunu belirler. Cell State: İlk olarak, hücre durumu (Cell State) unutma vektörü (Forget Vector) ile noktasal çarpıma tabi tutulur. Bu, hücre durumundaki değerlerin 0'a yakın değerlerle çarpılması

durumunda değerlerin düşmesine neden olabilir. Ardından, yeni güncellenmiş hücre durumunu Tanh fonksiyonundan geçirilir. Tanh çıkışı sigmoid çıkışı ile çarpılıp, bu da gizli durumun taşıması gereken bilgiyi belirler. Çıkış, gizli durumdur. Yeni hücre durumu ve yeni gizli durum bir sonraki zaman adımına taşınır. [3]



Şekil1. LSTM Algoritması Hücre İşleyişi [4]

Metin oluşturma, doğal dil işleme, hesaplamalı dilbilim ve yapay zekâ teknolojilerini bir araya getirerek belirli iletişimsel ihtiyaçları karşılamak üzere tasarlanmış bir alt disiplindir. Bu alandaki temel görevler, metinden metne üretim (T2T), veriden metne üretim (D2T) ve çok modaliteli metne üretim (M2T) olarak sınıflandırılabilir. T2T görevleri, mevcut metni alarak yeni bir tutarlı metin oluşturmayı amaçlar. D2T görevleri, sayısal veya yapılandırılmış verileri kullanarak metin oluşturur, örneğin, sayısal verilerden rapor hazırlama veya anlam temsillerinden metin oluşturma. M2T görevleri ise çok modlu girdilerdeki anlamı doğal dil metinlerine aktarır, örneğin, resimlerden altyazı oluşturma veya videoları metinle özetleme. Bu metin üretim kategorileri, çeşitli uygulama alanlarında iletişimsel gereksinimleri karşılamak ve otomatikleştirmek için geniş bir potansiyele sahiptir. [5]

Metin üretme stratejisi, ilk olarak geniş bir metin veri kümesi üzerinde bir dil modelini eğitme adımını içerir. Daha sonra, model, önceki karakterleri giriş olarak alarak bir sonraki karakteri veya karakterleri üretebilme yeteneğine sahip olur. Örneğin, "*Cat likes mil*" gibi bir dizide, bir sonraki karakterin 'k' olması tahmin edilebilir. LSTM modeli kullanılan bir eğitimde, sonraki karakterin olasılığını tahmin etmek için çıkış katmanında Softmax aktivasyon fonksiyonu kullanılır. Metin üretiminin esas sihri, bir sonraki karakterin oluşturulmasında kullanılan olasılık skorunun rastgele bir örneklenme stratejisi ile belirlenmesindedir. Örneğin, softmax çıkışından alınan en yüksek olasılık skoruna dayalı bir yaklaşım kullanıldığında, oluşturulan metin orijinal metne oldukça benzeyebilir. Ancak, rastgelelik eklemek için bir örneklem stratejisi kullanılarak bu durumun önüne geçilebilir. Bu strateji, bir sıcaklık değeri aracılığıyla rastgelelik seviyesini kontrol etmeye olanak tanır. [6]

Algoritma Analizi

LSTM algoritması, derin öğrenme temeli üzerine kurulu olduğu için tam olarak karmaşıklığını belirtmek zordur. Algoritmanın karmaşıklığı modelin derlenme ve eğitme aşamalarında kullanılan özel optimizasyonlara ve veri boyutuna bağlı olarak değişebilir. Big-O notasyonu ile genel olarak belirtmek gerekirse; T iterasyon sayısı, N girdi sayısı, M gizli katman olmak üzere Big-O notasyonu O(T x N x M) diyebiliriz

İterasyon Sayısı (T): Daha fazla iterasyon, modelin eğitim süresini artırabilir, ancak aynı zamanda daha iyi bir performans sağlayabilir. Fakat, aşırı eğitim (overfitting) riskini de artırabilir.

Girdi Sayısı (N): Daha büyük veri setleri, modelin genelleme yapma yeteneğini artırabilir, ancak eğitim süresini uzatabilir. Ayrıca, bellek gereksinimlerini ve algoritmik karmaşıklığı artırabilir.

Gizli Katman Sayısı (M): Daha fazla gizli katman, modelin daha karmaşık örüntüler öğrenme kapasitesine sahip olmasını sağlayabilir, ancak eğitim süresini ve hesaplama maliyetini artırabilir.

Ayrıca bu algoritma bilgisayardan bağımsız değildir, GPU ve CPU'ya oldukça bağlıdır ve çalışma zamanını gözle görülür şekilde etkiler. Algoritmanın derleme aşamasında kullanılan optimizasyonlar ve derleme araçları, çalışma zamanındaki performans üzerinde etkilidir. Bu uygulamada ise Adam optimizer kullanılmıştır. Adam (Adaptive Moment Estimation) Optimizer, gradyan tabanlı bir optimizasyon algoritmasıdır ve genellikle derin öğrenme modellerinde kullanılır. Bu algoritma, güncellemeleri adaptif bir şekilde ayarlamak ve modelin hiperparametrelerini (öğrenme oranı, momentum vb.) otomatik olarak ayarlamak için tasarlanmıştır.

Ayrıca algoritmanın performansını arttırmak için özellikle büyük veri setleri üzerinde çalışırken veri paralelleştirme teknikleri kullanarak eğitim süresi azaltılabilir. LSTM'nin ağırlıklarını iyi bir şekilde başlatılması, uygun kullanılan aktivasyon fonksiyonları sayesinde de algoritmanın performansı arttırılabilir.

Algoritma Tasarımı

Projemizde tek problem için 2 farklı çözüm bulunmaktadır. İlk çözüm LSTM algoritmasının iç yapısının manuel olarak yazılıp herhangi bir makine öğrenmesi kütüphanesi import edilmeden hazırlanmıştır. Diğer çözümde ise tensorflow, keras gibi derin öğrenme kütüphaneleri kullanılarak çözüme ulaşılmıştır.

İlk çözümün açıklanması (LSTM_Manuel):

Bu çözümde 3 farklı Python dosyası bulunmaktadır. lstm.py dosyasında lstm algoritmasının iç yapısı bulunmaktadır, train.py dosyasında lstm.py dosyasındaki modüller import edilip manuel olarak oluşturduğumuz model eğitilip kaydedilmektedir, predict.py doyasında ise eğitilmiş modelin pratiğe geçmektedir. Not: lstm.py dosyası tek başına çalışınca sonuç vermemektedir.

Lstm.py dosyası modüllerin açıklanması:

İlk olarak, çıkış katmanının boyutunu ve karakter kümesini başlatan initialize_output_units fonksiyonu bulunur. Bu, LSTM ağının eğitim aşamasında kullanılacak çıkış katmanının boyutunu belirler ve softmax aktivasyon fonksiyonunu içerir.

Ardından, sigmoid, hiperbolik tanjant, softmax ve tanjant türevi gibi temel aktivasyon fonksiyonları tanımlanmıştır. Bu fonksiyonlar, ağın içerdiği aktivasyon katmanlarında kullanılır. initialize_parameters fonksiyonu, LSTM hücresinin ağırlıklarını başlatır. lstm_cell fonksiyonu, bir LSTM hücresinin ileri yayılımını gerçekleştirirken, output_cell fonksiyonu çıkış hücresinin ileri yayılımını gerçekleştirir. get_embeddings fonksiyonu, embedding matrisi ile girdi verilerini çarpar.forward_propagation fonksiyonu, tam bir ileri yayılım gerçekleştirirken, cal_loss_accuracy fonksiyonu kayıp, perplexity ve doğruluk değerlerini hesaplar. calculate_output_cell_error fonksiyonu çıkış hücresinin hata ve aktivasyon hata terimlerini hesaplar.

calculate_single_lstm_cell_error fonksiyonu, bir LSTM hücresinin hatasını hesaplamak için kullanılır. calculate_output_cell_derivatives fonksiyonu çıkış hücresinin türevlerini hesaplar, calculate_single_lstm_cell_derivatives fonksiyonu ise bir LSTM hücresinin türevlerini hesaplar.

backward_propagation fonksiyonu tam bir geri yayılım gerçekleştirir ve update_parameters fonksiyonu, parametreleri Adam optimizer kullanarak günceller. update_embeddings fonksiyonu ise embedding matrisini günceller.

Son olarak, initialize_V ve initialize_S fonksiyonları, Adam optimizer için V ve S parametrelerini başlatır. train fonksiyonu, LSTM modelini eğitirken, save_model fonksiyonu eğitilen modeli bir dosyaya kaydeder. predict fonksiyonu ise eğitilen modeli kullanarak tahminlerde bulunur.

train.py dosyası modüllerinin açıklanması:

Bu kod, Long Short-Term Memory (LSTM) algoritması kullanarak bir dil modeli eğitmek amacıyla yazılmıştır. İlk adımda, "tutunamayanlar.txt" adlı metin dosyasından okunan veriler temizlenir. Her cümledeki noktalama işaretleri kaldırılarak metin, algoritmanın öğrenme sürecini kolaylaştırmak için işlenir. Ardından, LSTM modelinin uygulanması için gerekli fonksiyonlar ve parametreler başka bir modül içinde bulunur.

Veri hazırlığı aşamasında, isimlerin küçük harflere dönüştürülmesi ve en uzun ismin uzunluğuna eşit hale getirilmesi işlemi gerçekleştirilir. Bu adımların ardından vokabüler oluşturulur; her karakter, benzersiz bir sayısal kimlikle eşleştirilir ve tersi için bir sözlük oluşturulur. Bu, karakterleri sayısal bir temsil ile eğitimde kullanmak için gereklidir.

Eğitim verisi hazırlama aşamasında, belirli bir batch boyutunda ve isim uzunluğuna göre gruplandırılmış veri seti oluşturulur. Bu isimler, one-hot encoding kullanılarak nümerik formata dönüştürülür.

Modelin eğitim süreci, belirli bir sayıda iterasyon (epoch) boyunca gerçekleştirilir. Her iterasyonda, önceki adımların sonuçlarına dayanarak parametreler güncellenir. Eğitim sırasında her 1000 iterasyonda, kayıp (loss), doğruluk (accuracy) ve karmaşıklık (perplexity) değerleri ekrana yazdırılır. Eğitim sonuçları, iterasyonların ortalamaları alınarak görselleştirilir.

Eğitim tamamlandıktan sonra, elde edilen model, ağırlıklar, eğitim istatistikleri ve diğer önemli bilgiler "trained_model.joblib" adlı bir dosyaya kaydedilir. Modelin eğitim sonrasında kullanımıyla, yeni isimlerin oluşturulması gibi tahminlerde bulunulabilir. Ayrıca, eğitim sırasında kaydedilen kayıp, doğruluk ve karmaşıklık değerlerinin grafiksel gösterimi gerçekleştirilir.

İkinci Çözümün Açıklanması (LSTM):

Bu çözümde, train.csv veri setinden çekilen cümlelerle modelimizi eğitiyoruz ve eğitilmiş modele kelimeler vererek cümle üretilmesi sağlanıyor.

lstm. .ipynb dosyası modüllerin açıklanması:

keras.utils.pad_sequences: Serileri belirli bir uzunluğa kadar doldurmak için kullanılır. **keras.layers.Embedding**, **keras.layers.LSTM**, **keras.layers.Dense**, **keras.layers.Dropout**: Sinir ağı katmanları. **keras.preprocessing.text.Tokenizer**: Metin verisini tokenize etmek için kullanılır. **keras.callbacks.EarlyStopping**: Erken durma için geri çağrı. **keras.models.Sequential**: Sıralı bir Keras modeli oluşturmak için kullanılır. **keras.optimizers.Adam**: Adam optimizasyon algoritması. Diğer standart Python kütüphaneleri: **os**, **string**, **numpy**, **pandas**, **pickle**, **matplotlib.pyplot**.

İlk adım, veri okuma ve temizleme işlemleridir. `os.listdir(curr_dir)` fonksiyonu, mevcut dizindeki dosyaları listeler ve `pd.read_csv` fonksiyonu ile bir CSV dosyası okunur. Ardından, metin temizleme işlevi olan `clean_text` kullanılarak veri seti temizlenir ve `corpus` adlı bir liste oluşturulur.

Tokenizasyon aşamasında, `keras.preprocessing.text.Tokenizer` kullanılarak metin verisi tokenlara dönüştürülür. Bu işlem, `Tokenizer.fit_on_texts` fonksiyonu ile gerçekleştirilir ve `get_sequence_of_tokens` fonksiyonu ile veri tokenlara dönüştürülerek kullanıma hazır hale getirilir.

Eğitim için giriş verisi hazırlama adımında, modelin beklentilerine uygun giriş verisi oluşturulur. Bu işlemi gerçekleştiren `generate_padded_sequences` fonksiyonu kullanılır. Ayrıca, etiketleri kategorik formata dönüştürmek için `ku.to_categorical` fonksiyonu kullanılır.

Model oluşturma aşamasında, `create_model` fonksiyonu kullanılarak bir Keras modeli oluşturulur. Bu model, `Embedding`, `LSTM` ve `Dense` katmanları içerir. Modelin derlenmesi için kayıp fonksiyonu, optimize edici ve metrikler belirlenir.

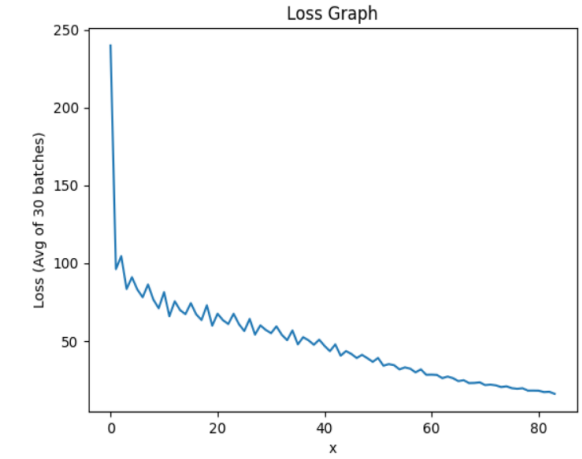
Eğitim aşamasında, `model.fit` fonksiyonu ile model eğitilir ve eğitim geçmişi elde edilir. Eğitilen model, `model.save` fonksiyonu ile diskte kaydedilir ve eğitim geçmişi, `pickle.dump` fonksiyonu kullanılarak bir dosyaya kaydedilir.

Modelin yüklenmesi ve eski eğitim geçmişinin görselleştirilmesi adımında, `load_model` fonksiyonu ile önceden eğitilmiş bir model yüklenir. Eski eğitim geçmişi dosyası yüklenir ve doğruluk ile kayıp grafiği çizilir.

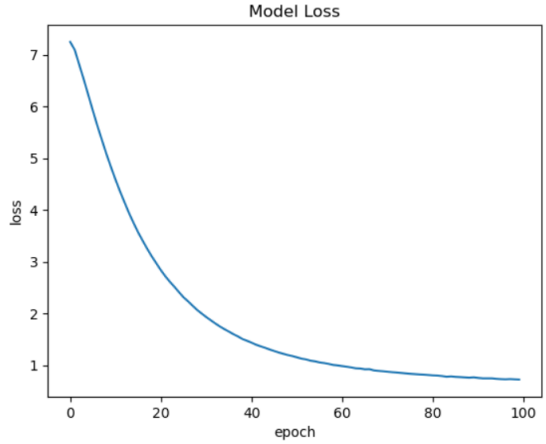
Son olarak, metin üretimi adımında, `generate_text` fonksiyonu kullanılarak model belirli bir başlangıç metniyle yeni metinler oluşturur. Oluşturulan örnek metinler ekrana yazdırılır. Bu aşamada, dil modelinin yaratıcı metin üretme yeteneğini gözlemleyebilir ve modelin eğitim sürecini başarıyla tamamladığını doğrulayabiliriz.

Test ve Değerlendirme

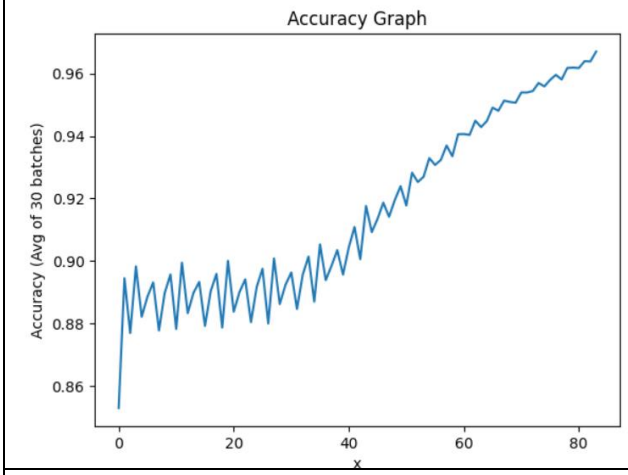
Doğruluk ve Hata Oranları Grafikleri:



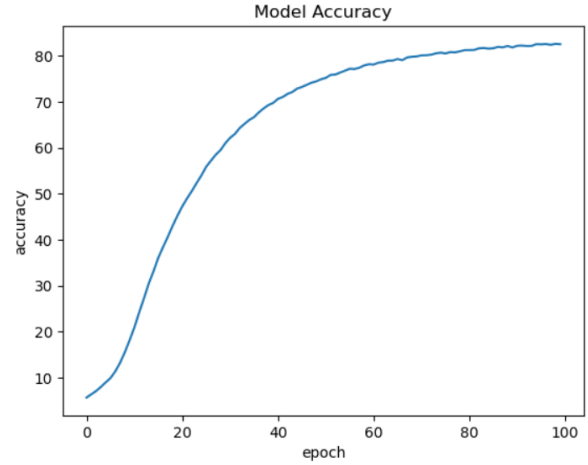
Manuel modeldeki loss grafiği



Hazır modelde loss grafiği



Manuel modeldeki doğruluk grafiği



Hazır modeldeki doğruluk grafiği

Doğruluk ve hata oranları grafiklerine bakıldığında, her iki çözümde de iterasyon ve epoch sayısı arttıkça dikkate değer şekilde doğruluk oranı yükseliyor, hata oranları düşüyor. Ancak ikinci çözümün grafiklerinde artış ve azalışlar stabil şekilde değişirken, birinci çözümde ani değişimler stabiliteyi bozmaktadır. İlk çözümde kullanılan veri seti, ikinci çözümde kullanılan veri setinden çok daha küçük olmasına rağmen, aynı makinede çalıştırıldığında benzer çalışma süreleri gözlemlenmiştir. (1: 110 dk; 2: 128dk)

Sonuçlar:

```
[['yazılardan hanesine yazacaklarımız şimdilik bu kadar . ']
 ['önümüzdeki hanesine yazacaklarımız şimdilik bu kadar . ']
 ['ettiği efendim bırakınca daha önce ne yapacağını kesinolarak bilen insanların görünüşüyle çalışma masasına yürüdü . ']
 ['gitti mağlup oluyorsunuz üstadım . ']
 ['satıcılarının hafta ankaraya gideceğim . ']
 ['ettiği efendim bırakınca daha önce ne yapacağını kesinolarak bilen insanların görünüşüyle çalışma masasına yürüdü . ']
 ['etmiş eserleri zorla ayırdı kendini oyuna gelmeyelim yeniden . ']
 ['gitti mağlup oluyorsunuz üstadım . ']
 ['bazı telaffuz hatalarına rağmen kıraatiniz fena değil turgut bey oğlumuz diyerek turgutun yanağını okşadı selim . ']
 ['önümüzdeki hanesine yazacaklarımız şimdilik bu kadar . ']]
```

Manuel Modelin Çıktıları
Kanada yük treni patlamasında ölü sayısı üçe ulaştı. Polis shevaun pennington'ın cumartesi günü 31 yaşındaki toby studabaker ile kaçmasının ardından uluslararası bir av başlattı Piyano çalan bir adam var. Benzin istasyonlarının dışında uzun kuyruklar oluştu. Benzer tasarılar iki kez meclis'ten geçti. Kuş ağaçtaki yiyecekleri bir kaseden yedi. Adam gitar çalıyor ve şarkı söylüyor.
Hazır Modelin Ürettiği Çıktılar

Beklenilen bir sonuç olarak üretilen çıktılar arasında fark oluşmaktadır. Hazır olarak kullanılan modelin çıktıları daha düzgün cümleler üretirken manuel olarak yazılmış modelin cümleleri daha devrik ve düzgün olmayan cümleler üretmektedir. İlk çözüm problemimizin çözümü için yetersiz kalmaktadır. Bu çözümün genelleme yeteneğini artırmak ve eğitim sürecini stabilize etmek gerekmektedir. İkinci çözüm ise problemin çözümü için daha verimli gözükmesine rağmen, daha optimize sonuçlar elde etmek için daha büyük ve verimli veri setleriyle çalışma yapılması gerekmektedir.

Projenin Zorlukları ve Katkıları:

Projeyi yaparken zorlandığımız en büyük konu modelin iç yapısını çözebilip anlayabilmekti. İç yapısı oldukça uzun ve karışık olmakla birlikte veri seti boyutu arttıkça çalışma zamanı artmaktadır. Başka bir zorluk ise kelimelerle cümle kurarken kelimelerin mantıklı bir şekilde birleştirilmesinde zorluklar yaşandı. Bize kattıkları ise LSTM algoritmasını uygulamaya dökerek doğal işleme alanında fazla bilgi sahibi olmamıza olanak sağladı, derin öğrenme kütüphanelerini bir uygulamada kullanmamız bize pratik kazandırdı.

Kaynaklar:

[1] Yang, S., Yu, X., & Zhou, Y. (2020, June). Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International workshop on electronic communication and artificial intelligence (IWECAI)* (pp. 98-101). IEEE.

[2] Kang, E. (2017). Long Short-Term Memory (LSTM): Concept. Medium: https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359

[3] Phi, M. (2018). *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. Medium: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[4] Jaiswal, A. (2022). Explaining Text Generation with LSTM. Analyticsvidhya: https://www.analyticsvidhya.com/blog/2022/02/explaining-text-generation-with-lstm/

[5] Goyal, R., Kumar, P., & Singh, V. P. (2023). A Systematic survey on automated text generation tools and techniques: application, evaluation, and challenges. *Multimedia Tools and Applications*, 1-56.

[6] Fatima, N., Imran, A. S., Kastrati, Z., Daudpota, S. M., & Soomro, A. (2022). A systematic literature review on text generation using deep neural network models. *IEEE Access*, *10*, 53490-53503.