KURAL TABANLI ALGORTİMALAR İLE TÜRKÇE DUYGU ANALİZİ

Enes Şevki DÖNMEZ, Kasım DELİACI, Lütfü BEDEL, Yasin Ekici, Yusuf GÜNEY

ÖZET

Bu çalışmada, Türkçe metinlerde duygu analizi gerçekleştirmek için sonlu durum otomatına (FSM) dayanan bir kural tabanlı yaklaşım geliştirilmiştir. Sistem, Türkçenin eklemeli yapısı ve bağlama dayalı anlam farklılıklarını dikkate alarak olumsuzluk ekleri, çifte olumsuzluklar ve pozitif/negatif bağlamları analiz etmektedir. Metin işleme için Zeyrek, NLTK ve Zemberek, Excel dosyaları için ise Pandas ve OpenPyXL kütüphaneleri kullanılmıştır. FSM modeli, hızlı ve açıklanabilir bir yapı sunarak Türkçe'nin karmaşıklığını etkili bir şekilde ele almayı hedeflemektedir.

1) GİRİŞ

1.1) Problemin Tanıtımı

Türkçe gibi eklemeli dillere özgü yapısal karmaşıklıklar, duygu analizi için önemli zorluklar oluşturmaktadır. Olumsuzluk ekleri, çifte olumsuzluklar ve bağlama bağlı anlam değişiklikleri, bu alanda özellikle dikkate alınması gereken unsurlardır. Bu çalışmada, büyük veri ve yüksek hesaplama gücü gerektiren makine öğrenimi yaklaşımlarından farklı olarak, açıklanabilir ve genişletilebilir bir kural tabanlı yöntem geliştirilmiştir.

1.2) Hedef

Sonlu durum otomatı (FSM) tabanlı bu yöntem, Türkçe metinlerde duygu analizi gerçekleştirmek için olumsuzluk eklerini, pozitif ve negatif kelimeleri, bağlam ilişkilerini ve çifte olumsuzlukları değerlendirir. FSM, belirli durumlar ve geçişler üzerinden çalıştığı için, geleneksel if-else yapılarının karmaşıklığını ve yönetim zorluklarını azaltarak daha düzenli ve genişletilebilir bir yapı sunar. Bu sayede, kuralların eklenmesi veya değiştirilmesi kolaylaşır ve sistemin izlenebilirliği artar. Sistem, hızlı, düşük veri ihtiyacıyla çalışabilir ve anlamca pozitif veya negatif sınıflandırmalar yapmayı hedefler. Bu raporda, geliştirilen sistemin yapısı, kullanılan yöntemler ve deney sonuçları detaylandırılmaktadır.

2) Metodoloji: Sonlu Durum Makinesi

Bu bölümde, Türkçe metinlerde duygu analizi yapmak için geliştirilen sonlu durum otomatı (FSM) tabanlı yöntemin yapısı ve kullanılan kural setleri detaylandırılmaktadır. FSM'in tanımı, dil işleme adımları ve duygu analizi için kullanılan kurallar açıklanmıştır.

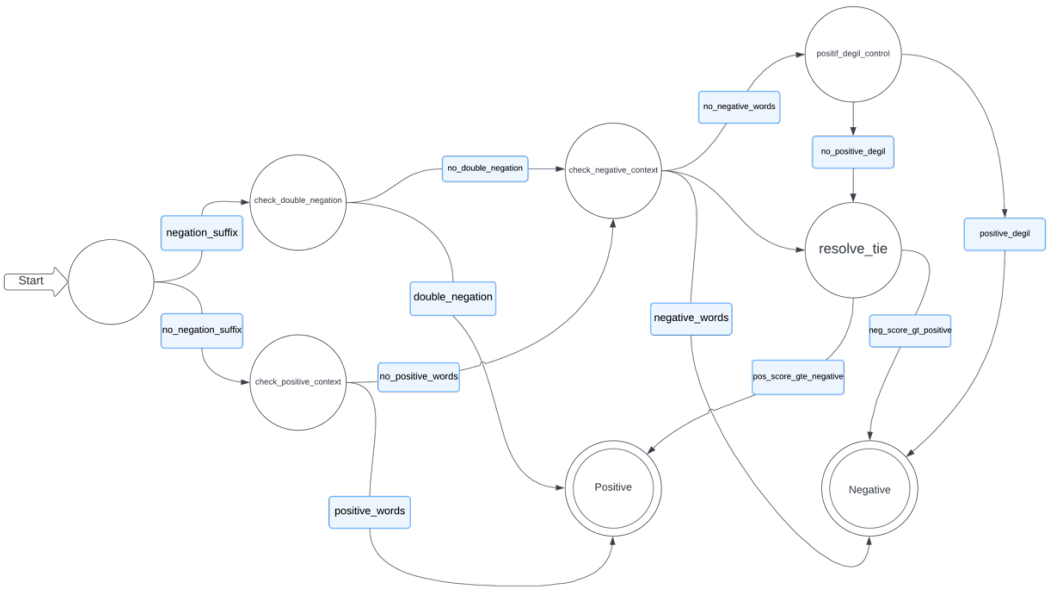
2.1) FSM Yapısı

Sonlu durum otomatı (FSM), belirli bir başlangıç durumundan başlar, giriş verisine bağlı olarak durumlar arasında geçiş yapar ve bir son duruma ulaşır. Bu çalışmada kullanılan FSM, duygu analizi için tasarlanmış olup, cümlelerin pozitif veya negatif sınıflandırılmasını sağlamaktadır. FSM'in her bir durumu belirli bir kontrol mekanizmasını temsil eder ve geçişler, cümlenin içerdiği özelliklere bağlı olarak yapılır.

FSM'in temel durumları şunlardır:

1. **Start:** Başlangıç durumu, analizin ilk adımıdır.
2. **Check Double Negation:** Çifte olumsuzlukları kontrol eder.
3. **Check Positive Context:** Pozitif kelimelerin varlığını kontrol eder.
4. **Check Negative Context:** Negatif kelimelerin varlığını kontrol eder.
5. **Positive Degil Control:** Pozitif kelimelerin "değil" ile kullanılıp kullanılmadığını değerlendirir.
6. **Resolve Tie:** Pozitif ve negatif skorları kıyaslayarak nihai sınıflandırmayı yapar.

FSM'in genel yapısı, sistemin genişletilebilirliğini ve izlenebilirliğini artırmak amacıyla, karmaşık if-else yapılarından kaçınarak tasarlanmıştır.



Şekil 1: Sonlu Durum Makinesi

2.2) Kural Setleri

Sistemin temel çalışma prensibi, belirli dilbilgisel ve anlamsal kurallara dayanmaktadır. Kullanılan başlıca kurallar şunlardır:

1.Olumsuzluk Kontrolü:

Cümlede olumsuzluk eki (-ma, -me) bulunuyorsa, cümlenin anlamının negatif olma olasılığı değerlendirilir.

2.Çifte Olumsuzluk Kontrolü:

Cümlede bir olumsuzluk eki ile birlikte "değil" kullanılmışsa, çifte olumsuzluk durumu tespit edilir ve bu bağlamda pozitif bir anlam çıkarımı yapılır.

3.Pozitif Kelime Tespiti:

Cümlede pozitif anlam taşıyan kelimelerin varlığı kontrol edilir. Örneğin, "harika", "mükemmel" gibi kelimeler pozitif bağlamın göstergesidir.

4.Negatif Kelime Tespiti:

Negatif anlam taşıyan kelimeler kontrol edilir. Örneğin, "kötü", "berbat" gibi kelimeler negatif bağlamın göstergesidir.

5.Pozitif "Değil" Kontrolü:

Pozitif kelimelerin "değil" ile kullanılması durumunda, bağlam negatif olarak değerlendirilir.

6.Skor Bazlı Karar:

Pozitif ve negatif kelime skorlarının kıyaslanması ile nihai sınıflandırma yapılır.

2.3) Dil İşleme Adımları

1.Noktalama İşaretlerinin Kaldırılması:

Metindeki noktalama işaretleri kaldırılarak analiz için temiz bir veri seti elde edilir.

2.Tokenizasyon:

Cümleler ve kelimeler ayrıştırılarak analiz yapılabilir hale getirilir.

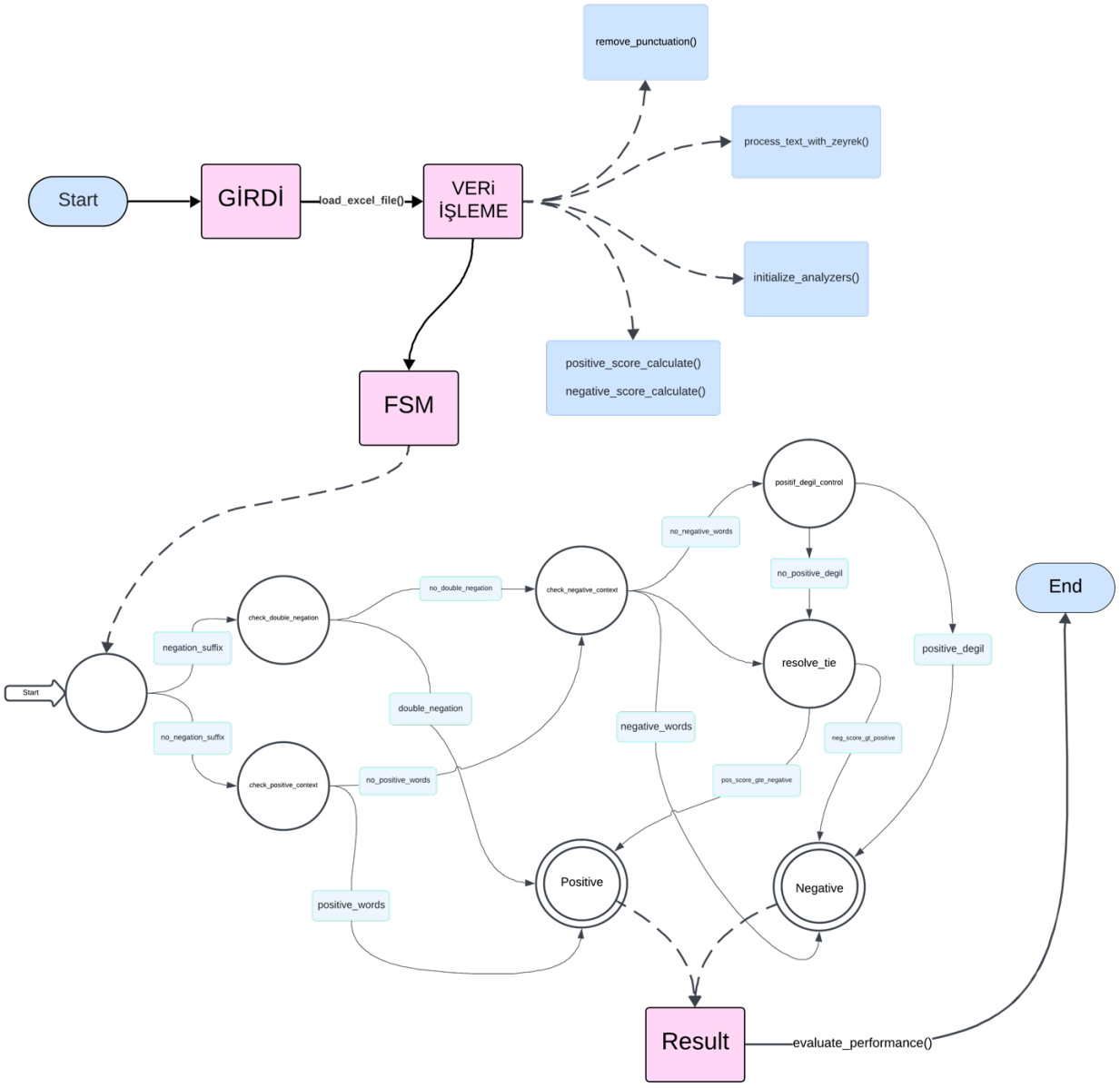
3.Kök ve Ek Ayrıştırma:

Zeyrek analizörü kullanılarak her bir kelimenin kökü ve ekleri ayrıştırılır. Bu, olumsuzluk eklerinin tespiti gibi kurallar için gereklidir.

4.Bağlam Analizi:

Cümlenin genel anlamını belirlemek için kelimeler arasındaki ilişkiler değerlendirilir.

Projenin Akış Diyagramı



Şekil 1.2: Akış Şeması

- - - - - : Kesik çizgili notasyon ilgili blok şemasının içerisini semboller.

3) Deney ve Sonuçlar

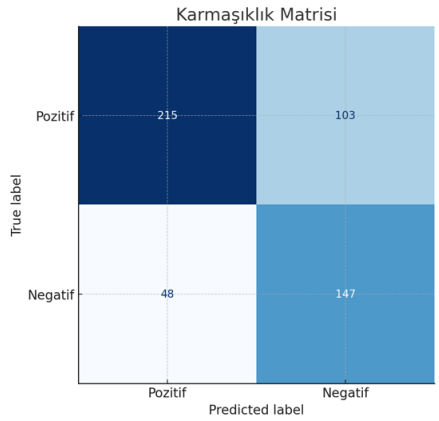
Bu bölümde, geliştirilen sistemin performansını değerlendirmek için gerçekleştirilen deneyler ve elde edilen sonuçlar sunulmaktadır. Analiz, modelin doğruluğunu, kesinliğini, anmasını ve F1-ölçütünü içeren performans metrikleri üzerinden yapılmıştır. Ayrıca, sistemin sınıflandırma başarısı karmaşıklık matrisi ve tahmin oranları üzerinden görselleştirilmiştir.

3.1) FSM Performansı

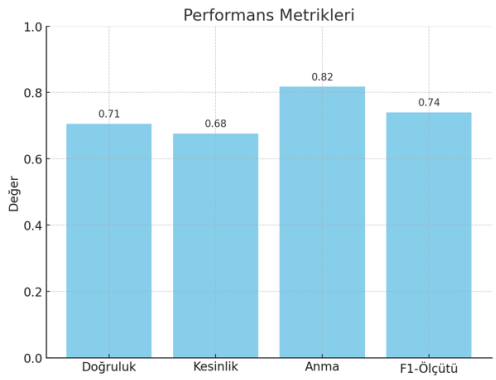
Geliştirmeler sonucunda FSM tabanlı sistemin nihai performansı şu şekilde elde edilmiştir:

- Doğruluk (Accuracy): %71
- Kesinlik (Precision): %68
- Anma (Recall): %82
- F1-Ölçütü: %74

Bu performans metrikleri, sistemin pozitif ve negatif sınıflandırmalar için dengeli bir performans sergilediğini göstermektedir. Özellikle anma oranının (%82) yüksek olması, pozitif sınıflandırmalarda güçlü bir başarıya işaret etmektedir. Ancak doğruluk ve kesinlik değerlerini geliştirmemiz gerektiği ortadadır.



Şekil 2:Karmaşıklık Matrisi



Şekil 3: Performans Metrikleri

3.2) Pozitif ve Negatif Kelime Listesinin Genişletilmesi

Pozitif ve negatif polariteli kelimelerin bulunduğu dosya, sistemin performansını artırmak amacıyla detaylı bir şekilde analiz edilerek güçlendirilmiştir. Yanlış tahminler üzerinde yapılan analizler sonucunda, FSM'nin bazı durumları doğru şekilde yakalayamadığı ve bu durumların eksik kelime listelerinden kaynaklandığı belirlenmiştir. Bu eksiklikleri gidermek için kelime listemiz genişletilmiş ve sistem yeniden değerlendirilmiştir. Yapılan güncellemeler, modelin daha geniş bağlamları algılayabilmesini sağlamış ve performans metriklerinde belirgin iyileşmeler elde edilmiştir.

Geliştirmeler sonucunda FSM tabanlı sistemin nihai performansı şu şekilde elde edilmiştir:

- Doğruluk (Accuracy): %77
- Kesinlik (Precision): %77
- Anma (Recall): %79
- F1-Ölçütü: %78

Doğruluk (Accuracy): %71 → %77

Sistem, genel sınıflandırma doğruluğunu %6 oranında artırmıştır. Bu artış, kelime listesindeki eksikliklerin giderilmesi ve FSM'nin daha fazla bağlamı doğru bir şekilde sınıflandırabilmesi ile sağlanmıştır. Yanlış pozitif ve yanlış negatif tahminlerin azalması, bu metrikteki yükselişe önemli bir katkı sağlamıştır.

Kesinlik (Precision): %68 → %77

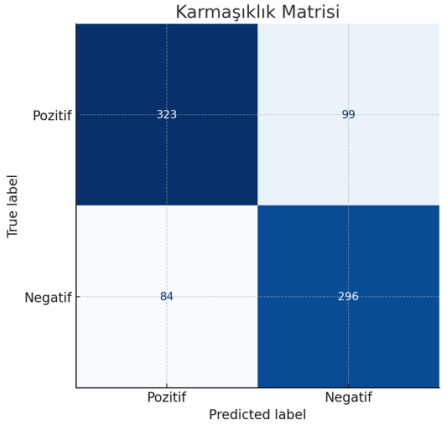
Kesinlik oranındaki %9'luk artış, sistemin pozitif tahminlerinin doğruluğunu artırdığını göstermektedir. Yeni kelime listesi, pozitif polariteli kelimeleri daha doğru bir şekilde tespit etmiş ve yanlış pozitif tahminlerin azalmasını sağlamıştır.

Anma (Recall): %82 → %79

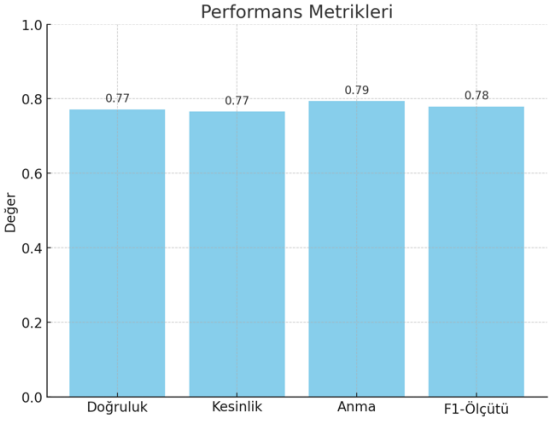
Anma oranındaki %3'lük düşüş, modelin pozitif sınıflamaları yakalama duyarlılığında hafif bir azalmayı göstermektedir. Ancak bu düşüş, diğer metriklerdeki iyileşmelerle dengelenmiş ve modelin genel performansını olumsuz etkilememiştir.

F1-Ölçütü (F1-Score): %74 → %78

F1-ölçütündeki %4'lük artış, kesinlikteki iyileşmenin anma oranıyla dengeli bir şekilde birleştiğini ve modelin genel performansında anlamlı bir gelişme sağlandığını göstermektedir.



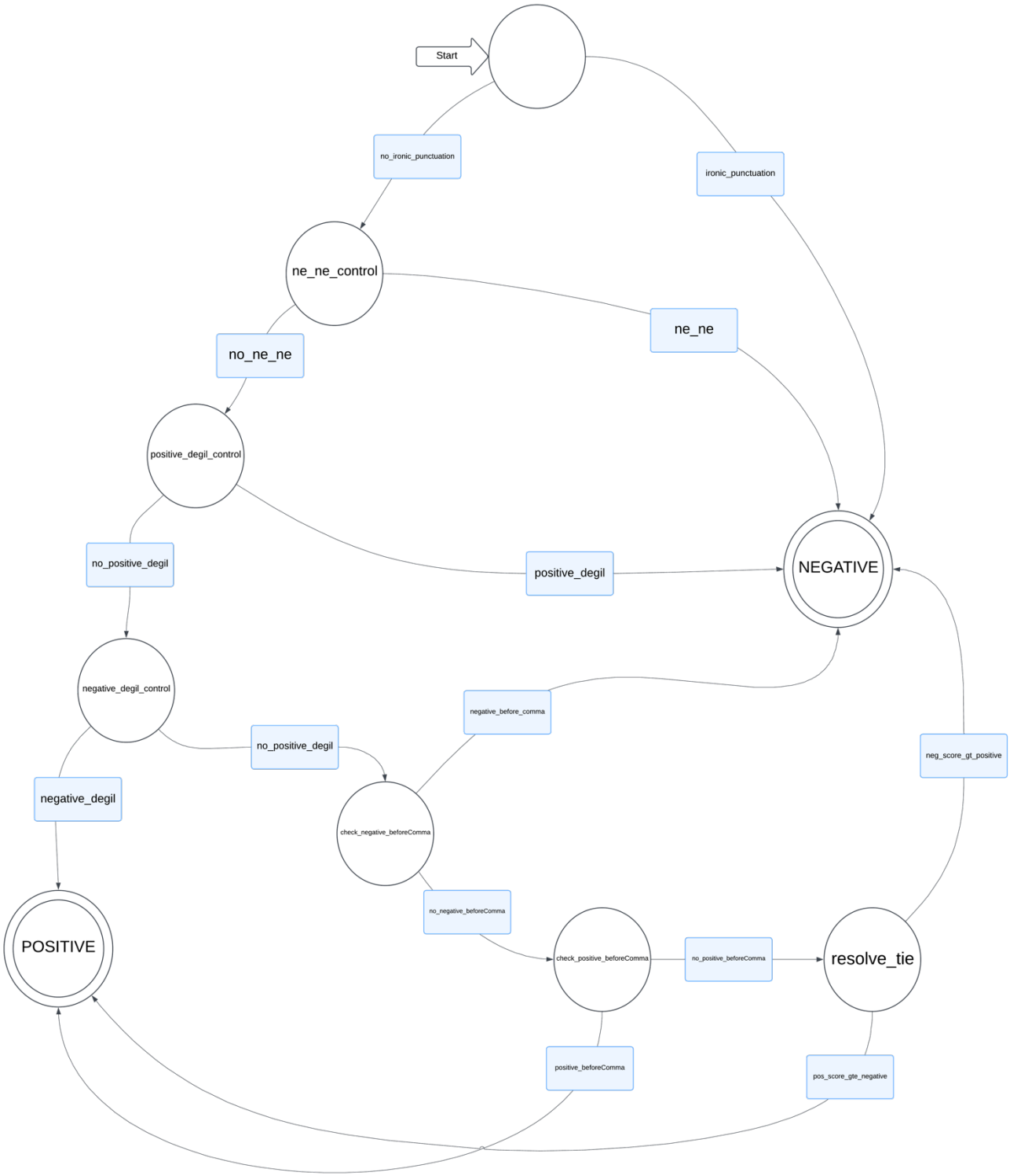
Şekil 4: Karmaşıklık Matrisi



Şekil 5:Performans Metrikleri

3.3) Sonlu Durum Otomatı Tasarımı

Yeni FSM, önceki modelin eksikliklerini gidermek ve daha geniş bağlamları analiz edebilmek için tasarlanmıştır. Bunun için eski FSM'nin hataları detaylıca incelenerek yeni kurallar ve durumlar oluşturulmuştur, yeni sonlu durum makinesi Türkçe'nin bağlama dayalı karmaşıklığını daha iyi ele almayı amaçlamaktadır.



Şekil 6: FSMV2 Tasarımı

Yeni FSM, duygu analizi sürecinde aşağıdaki adımları izler:

1. **Başlangıç Durumu (Start):**
    - FSM, analiz sürecine bu durumdan başlar.
    - **Kontrol Edilen Kural:** Cümlede ironi veya alay ifade eden noktalama işaretleri var mı?
        - **ironic_punctuation**: Eğer "!" veya "?!", ironi ifade eden işaretler tespit edilirse, FSM doğrudan **"Negatif"** durumuna geçer.
        - **no_ironic_punctuation**: Böyle bir işaret yoksa, FSM bir sonraki kontrol noktası olan **"ne_ne_control"** durumuna geçer.

2. **Ne-Ne Kontrolü (ne_ne_control):**
    - **Kontrol Edilen Kural:** Cümlede "ne ... ne" yapısı var mı?
        - **ne_ne**: Eğer bu yapı tespit edilirse, FSM doğrudan **"Negatif"** durumuna geçer.
        - **no_ne_ne**: Eğer böyle bir yapı yoksa, FSM bir sonraki aşama olan **"positive_degil_control"**durumuna geçer.

3. **Pozitif "Değil" Kontrolü (positive_degil_control):**
    - **Kontrol Edilen Kural:** Cümlede pozitif kelimeler "değil" ile birlikte mi kullanılıyor?
        - **positive_degil**: Pozitif kelimeler "değil" ile birlikte kullanılmışsa, FSM **"Negatif"** durumuna geçer.
        - **no_positive_degil**: Böyle bir durum yoksa, FSM **"negative_degil_control"** durumuna geçer.

4. **Negatif "Değil" Kontrolü (negative_degil_control):**
    - **Kontrol Edilen Kural:** Cümlede negatif kelimeler "değil" ile birlikte mi kullanılıyor?
        - **negative_degil**: Negatif kelimeler "değil" ile birlikte kullanılmışsa, FSM **"Pozitif"** durumuna geçer.
        - **no_negative_degil**: Böyle bir durum yoksa, FSM **"check_negative_beforeComma"** durumuna geçer.

5. **Virgülden Önce Negatiflik Kontrolü (check_negative_beforeComma):**
    - **Kontrol Edilen Kural:** Cümlede bir virgülden önce negatif bağlam var mı?
        - **negative_beforeComma**: Böyle bir bağlam varsa, FSM **"Negatif"** durumuna geçer.
        - **no_negative_beforeComma**: Böyle bir bağlam yoksa, FSM **"check_positive_beforeComma"**durumuna geçer.

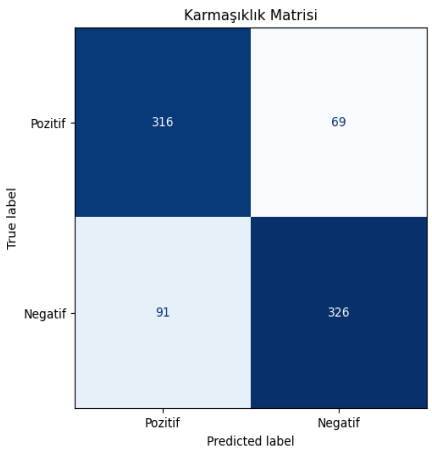
6. **Virgülden Önce Pozitiflik Kontrolü (check_positive_beforeComma):**
    - **Kontrol Edilen Kural:** Cümlede bir virgülden önce pozitif bağlam var mı?
        - **positive_beforeComma**: Böyle bir bağlam varsa, FSM **"Pozitif"** durumuna geçer.
        - **no_positive_beforeComma**: Böyle bir bağlam yoksa, FSM **"resolve_tie"** durumuna geçer.

7. **Skor Tabanlı Karar (resolve_tie):**
    - **Kontrol Edilen Kural:** Pozitif ve negatif kelime skorları kıyaslanır.
        - **positive_score_gte_negative**: Pozitif skorlar negatif skorlara eşit ya da daha büyükse, FSM **"Pozitif"** durumuna geçer.
        - **negative_score_gt_positive**: Negatif skorlar daha büyükse, FSM **"Negatif"** durumuna geçer.

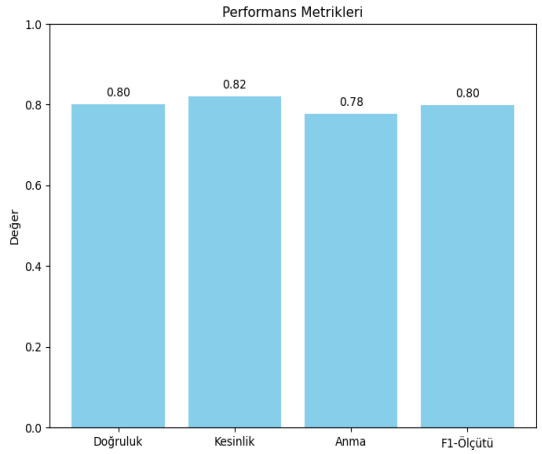
Geliştirmeler sonucunda FSM tabanlı sistemin nihai performansı şu şekilde elde edilmiştir:

- Doğruluk (Accuracy): %80
- Kesinlik (Precision): %82
- Anma (Recall): %78
- F1-Ölçütü: %80

FSM modelindeki değişiklikler, sistemin genel performansında dikkat çekici bir iyileşme sağlamıştır. Doğruluk oranı %77'den %80'e, kesinlik %77'den %82'ye yükselerek modelin pozitif tahminlerindeki doğruluğunu artırmıştır. Anma oranı %79'dan %78'e hafif bir düşüş göstermiş olsa da, F1-ölçütü %78'den %80'e çıkarak kesinlik ve anma dengesini daha iyi bir şekilde yansıtmaktadır. Bu gelişmeler, FSM'in bağlama dayalı kurallarla daha güçlü hale getirildiğini ve yanlış pozitif sınıflamaların azaldığını göstermektedir.



Şekil 6: Karmaşıklık Matrisi



Şekil 7: Performans Metrikleri

3.4) RegEx ile Olumsuzluk Eklerinin Yakalanması

Hatalarımızın analizini yaparken kelimelerdeki olumsuzluk eklerini düzgün yakalayamadığımızı fark ettik. Türkçe metinlerde olumsuzluk ekleri (*-ma, -me, -mı, -mi*) ile isim-fiil eklerini (*-mayı, -meye*) doğru bir şekilde ayırt ederek duygu analizi sürecinde daha yüksek doğruluk oranları elde etmeyi hedefliyoruz. Türkçenin eklemeli yapısının getirdiği zorlukları ele almak için regex tabanlı bir çözüm geliştirilmiştir. Bu çözüm, dilbilgisel yapıya dayalı hatalı sınıflandırmaları en aza indirmeyi amaçlar.

1. İsim-Fiil Eklerini Tespit Eden Regex:

r'^(.?)(ma|me|mı|mi)([yğ][ıiüuae]).*'*

- **Desenin İşleyişi:**
  - (ma|me|mı|mi) → *ma, me, mı, mi* eklerini hedefler.
  - ([yğ][ıiüuae]) → Bu ekten sonra gelen bir *y* veya *ğ* harfiyle başlayıp bir ünlü harfle devam eden yapıları yakalar (*-mayı, -meye, -meyi*).
  - .* → Ekin ardından gelen herhangi bir karakter dizisini kabul eder.
  - ^ ve $ → Kelimenin baştan sona taranmasını sağlar.

- **Dikkat Edilmesi Gereken Husular: Deseni Daraltmak**

  r'^(.?)(ma|me)(y[ıiüuae])(?!n).*'*

- (?!n) → Lookahead kullanılarak, *yı* veya *yi* gibi eklerden sonra *n* harfi gelmesi durumunda eşleşme yapılmaz.
- Bu desen, örneğin *koşmayın* gibi kelimelerin yanlış şekilde isim-fiil olarak algılanmasını önler.

2. Olumsuzluk Eklerini Tespit Eden Regex:

r'^(.?)(ma|me|mı|mi).*'*

- **Desenin İşleyişi:**
  - (ma|me|mı|mi) → Kelimenin herhangi bir yerinde *ma, me, mı, mi* eklerini arar.
  - Bu desen, olumsuzluk anlamı katan yapıları (ör. *okumadım, sevmiyor*) doğru şekilde tespit eder.

- **Lookahead (?!...):**
  - Belirli bir desenin ardından başka bir desenin gelmemesi gerektiğini belirtir.
  - Örneğin (?!n) ile *yı* ekinden sonra bir *n* harfi varsa eşleşmeyi engelleyip bunun isim-fiil olmadığını söyleyebiliriz.

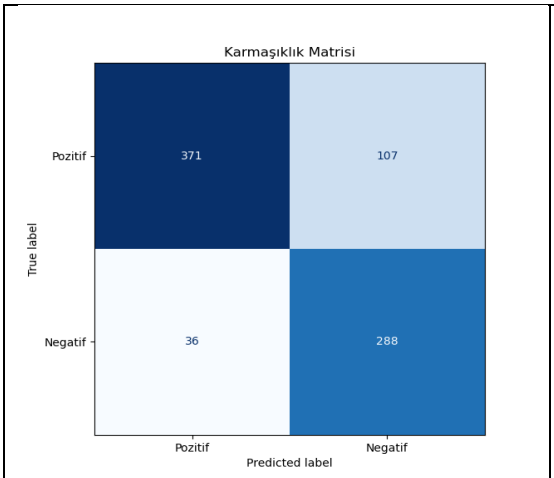
Regex önce isim fiil eklerini kontrol eder ardından daha dar bir isim-fiil kontrolü yaparak "koşmayın" gibi yapıları yakalar. Sonrasında olumsuzluk ekleri için olan yazılmış ifadeyi kontrol edeyerek süreci tamamlar.

Kelime	Sonuç
koşmayı	İsim-fiil eki
koşmayın	Olumsuzluk eki
okumadım	Olumsuzluk eki
gitmeyi	İsim-fiil eki
sevme	Olumsuzluk eki

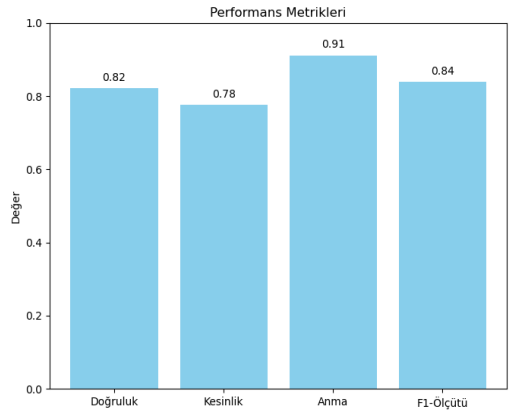
RegEx geliştirmeleri sonucunda sistemin performans metrikleri aşağıdaki gibi elde edilmiştir.

- Doğruluk (Accuracy): %82
- Kesinlik (Precision): %78
- Anma (Recall): %91
- F1-Ölçütü: %84

Regex ile olumsuzluk eklerinin isim fiillerde ayrımını yaparak sistemi geliştirmek, performans metriklerinde belirgin bir iyileşme sağlamıştır. Doğruluk oranı %82'ye yükselmiş ve model genel olarak daha doğru tahminler yapmıştır. Kesinlik %78'e düşmesine rağmen, anma oranındaki %91'e çıkan artış, modelin pozitif örnekleri yakalama kapasitesini önemli ölçüde artırdığını göstermektedir. F1-ölçütü ise %84'e çıkarak kesinlik ve anma arasındaki dengenin daha da iyileştiğini ifade etmektedir. Bu sonuçlar, regex eklemesinin özellikle bağlam analizi ve olumsuzluk tespitinde sistemin duyarlılığını artırdığını göstermektedir.



Şekil 8: Karmaşıklık Matrisi



Şekil 9:Performans Metrikleri

3.5) Son Bakış: RegEx Sonrası Sistemin Kontrolü ve Kelime Listesi Genişletilmesi

Projemizde, ilk olarak temel bir FSM modeli geliştirildi ve performansı analiz edildi. Daha sonra, pozitif ve negatif kelime listesi güçlendirilerek modelin bağlam algısını iyileştirildi. Ardından, eksikleri giderdiğimiz yeni bir FSM modeli tanıttık, bu modelin gelişiminin üzerine regex ile ekleri daha doğru ayrıştırarak duyarlılığı arttırdık şimdi eski yaklaşımların güçlü yönlerini koruyarak daha geniş bağlamları ve dil yapısını etkili bir şekilde ele almayı hedefleyen final FSM modelimizi tanıtacağız.

1. Başlangıç Durumu (start)

FSM analizine başlangıç noktasıdır. Burada, cümlenin genellikle en üst düzeyde anlamını etkileyen özellikler kontrol edilir.

- **Kural:**
    - **ironic_punctuation**: Cümlede ironi veya alay ifade eden bir noktalama işareti varsa (örneğin, (!) FSM doğrudan **"Negatif"** durumuna geçer.
    - **no_ironic_punctuation**: İronik bir noktalama işareti yoksa, FSM **"ne_ne_control"** durumuna geçer.

---

2. Ne-Ne Kontrolü (ne_ne_control)

Bu durum, Türkçe'de sıkça kullanılan "ne ... ne" yapısını değerlendirir.

- **Kural:**
    - **ne_ne**: Eğer "ne ... ne" yapısı varsa (örneğin, "ne güzel ne kötü"), FSM doğrudan **"Negatif"** durumuna geçer.
    - **no_ne_ne**: Böyle bir yapı yoksa, FSM **"positive_degil_control"** durumuna geçer.

---

3. Pozitif "Değil" Kontrolü (positive_degil_control)

Cümlede pozitif kelimelerin "değil" ile birlikte kullanılıp kullanılmadığı değerlendirilir.

- **Kural:**
    - **positive_degil**: Pozitif kelimeler "değil" ile birlikte kullanılmışsa, FSM **"Negatif"** durumuna geçer.
    - **no_positive_degil**: Böyle bir kullanım yoksa, FSM **"negative_degil_control"** durumuna geçer.

---

4. Negatif "Değil" Kontrolü (negative_degil_control)

Negatif kelimelerin "değil" ile birlikte kullanılıp kullanılmadığını değerlendirir.

- **Kural:**
    - **negative_degil**: Negatif kelimeler "değil" ile birlikte kullanılmışsa, FSM **"Pozitif"** durumuna geçer.
    - **no_negative_degil**: Böyle bir kullanım yoksa, FSM **"check_negative_beforeComma"** durumuna geçer.

5. Virgülden Önce Negatiflik Kontrolü (check_negative_beforeComma)

Cümlede bir virgülden önce negatif bir bağlamın olup olmadığı kontrol edilir.

- **Kural:**
  - **negative_beforeComma**: Eğer negatif bir bağlam varsa, FSM **"Negatif"** durumuna geçer.
  - **no_negative_beforeComma**: Negatif bağlam yoksa,
    FSM **"check_positive_beforeComma"** durumuna geçer.

---

6. Virgülden Önce Pozitiflik Kontrolü (check_positive_beforeComma)

Virgülden önce pozitif bir bağlamın olup olmadığı kontrol edilir.

- **Kural:**
  - **positive_beforeComma**: Pozitif bağlam varsa, FSM **"Pozitif"** durumuna geçer.
  - **no_positive_beforeComma**: Pozitif bağlam yoksa,
    FSM **"check_end_with_degil"** durumuna geçer.

---

7. "Değil" ile Bitme Kontrolü (check_end_with_degil)

Cümlenin "değil" ile bitip bitmediği kontrol edilir.

- **Kural:**
  - **end_with_degil**: Cümle "değil" ile bitiyorsa, FSM **"Negatif"** durumuna geçer.
  - **no_end_with_degil**: Cümle "değil" ile bitmiyorsa,
    FSM **"check_conjunctions_word"** durumuna geçer.

---

8. Bağlaç Kelime Kontrolü (check_conjunctions_word)

Cümlede bağlaç kelimelerle negatif bir bağlam oluşturulup oluşturulmadığını kontrol eder.

- **Kural:**
  - **conjunctions_word**: Eğer bağlaçlar negatif bir anlam taşıyorsa,
    FSM **"Negatif"** durumuna geçer.
  - **no_conjunctions_word**: Böyle bir durum yoksa,
    FSM **"check_hic_before_pos"** durumuna geçer.

9. "Hiç" ve Pozitif Bağlam Kontrolü (check_hic_before_pos)

"Hiç" kelimesinin pozitif bir bağlamda kullanılıp kullanılmadığını kontrol eder.

- **Kural:**
    - **hic_before_pos**: Eğer "hiç" pozitif bir bağlamı olumsuz yapıyorsa, FSM **"Negatif"** durumuna geçer.
    - **no_hic_before_pos**: Böyle bir durum yoksa, FSM **"check_hic_before_neg"** durumuna geçer.

---

10. "Hiç" ve Negatif Bağlam Kontrolü (check_hic_before_neg)

"Hiç" kelimesinin negatif bir bağlamda kullanılmasını değerlendirir.

- **Kural:**
    - **hic_before_neg**: Eğer "hiç" negatif bir bağlamı güçlendiriyorsa, FSM **"Pozitif"** durumuna geçer.
    - **no_hic_before_neg**: Böyle bir durum yoksa, FSM **"resolve_tie"** durumuna geçer.

---

11. Skor Tabanlı Karar (resolve_tie)

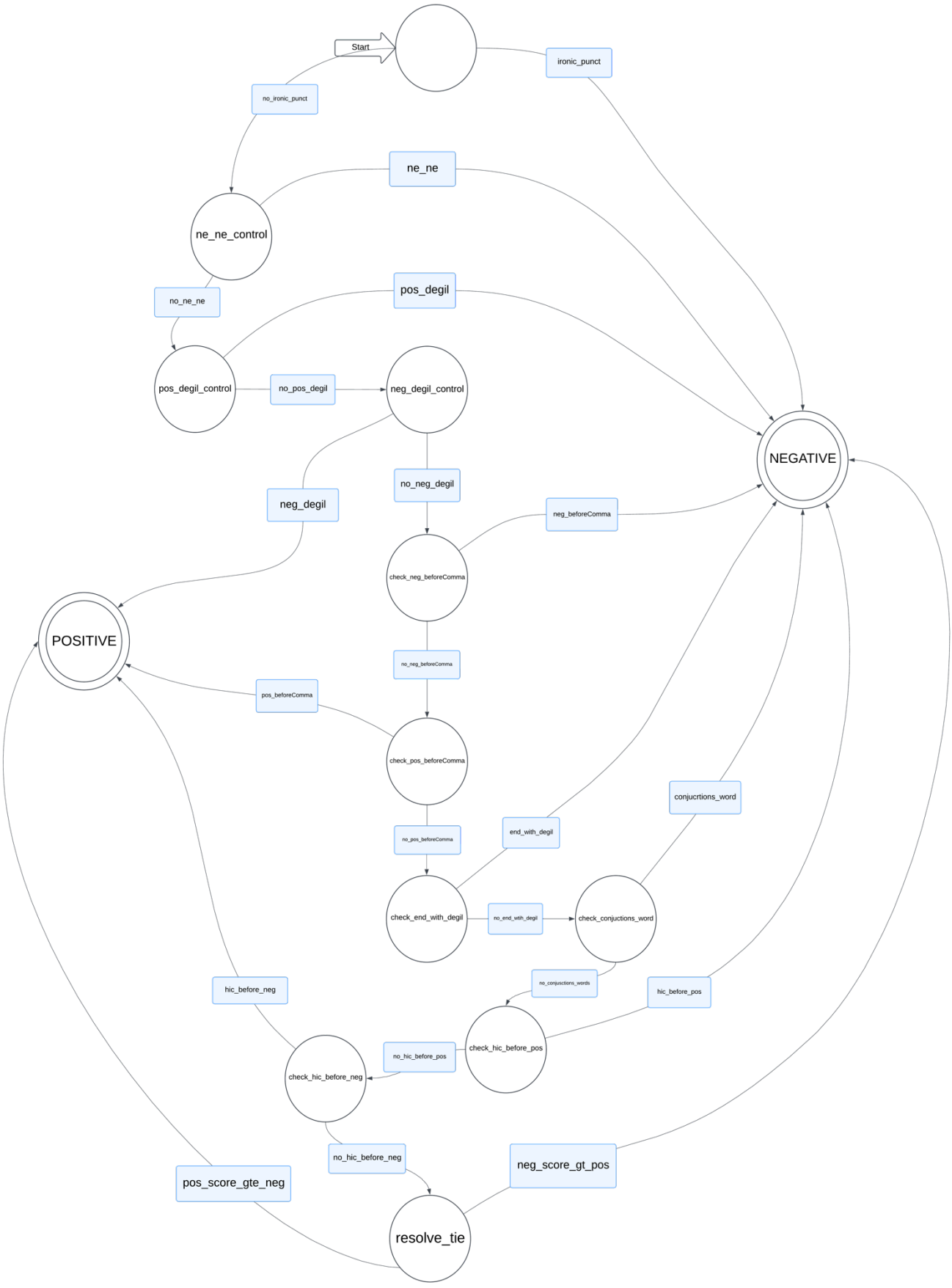
Pozitif ve negatif kelime skorları karşılaştırılarak nihai karar verilir.

- **Kural:**
    - **positive_score_gte_negative**: Pozitif skorlar negatif skorlara eşit ya da daha büyükse, FSM **"Pozitif"**durumuna geçer.
    - **negative_score_gt_positive**: Negatif skorlar daha büyükse, FSM **"Negatif"** durumuna geçer.

Bu yeni FSM modeli, önceki modellerin güçlü yönlerini korurken şu yeniliklerle sistemin bağlam algısını geliştirmiştir:

1. **Bağlaç ile Ayrıştırma:** Cümle bölümleri ayrı ayrı analiz edilerek daha doğru sınıflandırma yapılması sağlanmıştır.
2. **"Hiç" Analizi:** Türkçe'nin özgün kullanımlarından olan "hiç" kelimesinin bağlama etkisi detaylıca değerlendirilmiştir.

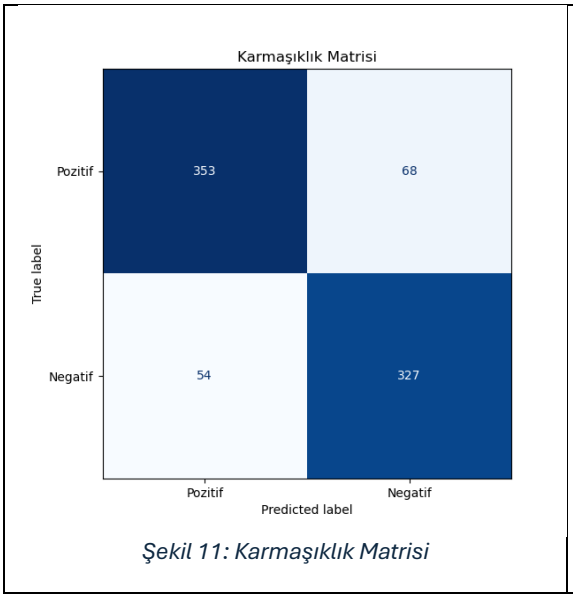
Bu FSM modeli, duygu analizinde daha karmaşık bağlamları ele alarak sistemin doğruluğunu ve hassasiyetini artırmayı hedeflemektedir.



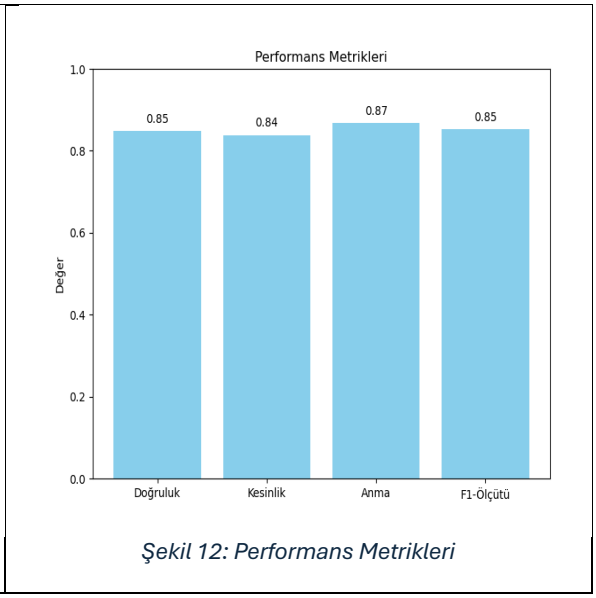
Şekil 10: FSMV3 Tasarımı

Son FSM tasarımı ve geliştirilmiş kelime listesi ile sistemin performans metrikleri aşağıdaki gibi elde edilmiştir.

- Doğruluk (Accuracy): %85
- Kesinlik (Precision): %84
- Anma (Recall): %87
- F1-Ölçütü: %85



Şekil 11: Karmaşıklık Matrisi

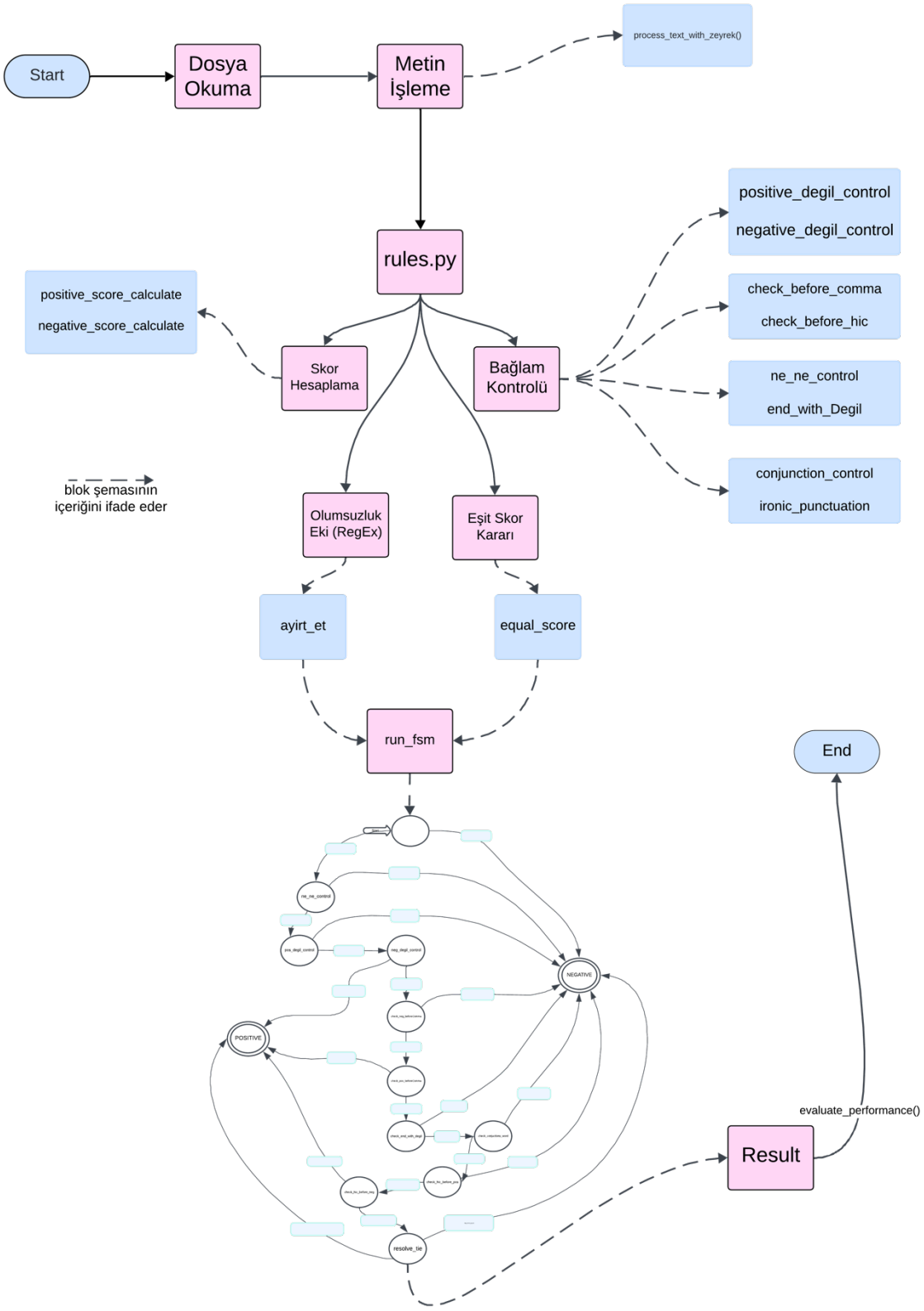


Şekil 12: Performans Metrikleri

Yeni FSM modeli ile elde edilen sonuçlar, sistemin performansında önemli iyileştirmeler sağlandığını göstermektedir**. Doğruluk oranı %82'den %85'e yükselerek** modelin genel doğruluğunun arttığını ortaya koymuştur. **Kesinlik (%78 → %84) ve anma (%91 → %87) değerleri dengelenmiş**, bu da modelin pozitif ve negatif sınıflamaları daha tutarlı bir şekilde yaptığına işaret etmektedir. **F1-ölçütü ise %84'ten %85'e çıkarak kesinlik ve anma arasındaki uyumun güçlendiğini göstermektedir.** Bu sonuçlar, yeni FSM modelinin bağlama dayalı analizlerde daha güçlü bir performans sunduğunu ve sistemi genel olarak daha güvenilir hale getirdiğini göstermektedir.

4) Proje Akış Şeması ve Detayları

Bu bölümde proje ilk andan itibaren hangi süreçler ve yöntemler ile ilerliyor detaylıca ele alınıp projenin işleyiş şeması ve kurallar daha detaylı açıklanmıştır.



Proje Akış Şeması 1

Bu projede, Türkçe metinlerde duygu analizi gerçekleştirmek için bir sonlu durum otomatı (FSM) ve kurallara dayalı bir yaklaşım kullanılmıştır. Proje, iki ana dosyadan oluşmaktadır: main.py ve rules.py.

Genel Akış

1. **Dosya Okuma ve Başlangıç Ayarları (main.py):**
    - main() fonksiyonu, bir Excel dosyasından metin verilerini alır ve işleme başlar.
    - Zeyrek kütüphanesi kullanılarak metin, kelime köklerine, eklerine ve dil bilgisel özelliklerine ayrıştırılır.
2. **Kuralların Uygulanması (rules.py):**
    - rules.py, FSM kurallarını uygulayan ve metni analiz eden çeşitli fonksiyonları içerir.
    - Olumlu ve olumsuz polariteli kelimeler, bağlaçlar ve bağlam analizi gibi unsurlar değerlendirilir.
3. **FSM'nin Çalıştırılması (main.py):**
    - Kurallar sonucunda elde edilen bilgiler, FSM üzerinde yürütülerek cümlenin duygu durumu ("Pozitif" veya "Negatif") belirlenir.
4. **Performans Değerlendirmesi (main.py):**
    - FSM'in tahminleri, gerçek sınıflarla karşılaştırılarak doğruluk, kesinlik, anma ve F1-ölçütü gibi metrikler hesaplanır.

---

1. Dosya Okuma ve Metin İşleme

main.py - main(file_path)

- Girdiler: Excel dosyasındaki cümleler ve sınıflar.
- Çıktılar: İşlenmiş metin ve sonuç dosyaları.
- Akış:
    1. Excel dosyası okunur, cümleler ve gerçek sınıflar alınır.
    2. Zeyrek analizörü başlatılarak cümleler kelime seviyesinde işlenir.
    3. Her cümle için FSM çalıştırılır ve tahminler üretilir.
    4. Tahminler Excel dosyalarına kaydedilir, hatalı tahminler ayrı bir dosyada toplanır.
    5. Performans değerlendirme metrikleri hesaplanır.

process_text_with_zeyrek(text, analyzer)

- Amaç: Cümleleri Zeyrek analizörü kullanarak kelime köklerine, eklerine ve dil bilgisel türlere (POS) ayırmak.
- İşleyiş:
    - Metni cümlelere ve kelimelere ayırır.
    - Kelimelerin köklerini, eklerini ve POS etiketlerini çıkarır.
    - Bağlaç kelimelerden önceki kelimeleri kaldırır.

---

2. Kuralların Uygulanması

rules.py - Fonksiyonlar

1. **Skor Hesaplama:**
    - positive_score_calculate ve negative_score_calculate:
        - Pozitif ve negatif polariteli kelimeleri belirler.
        - polarity_positive.txt ve polarity_negative.txt dosyalarını kullanarak kelimelerin polaritesini tespit eder.
        - Skorları hesaplar ve polariteli kelimeleri listeler.
2. **Bağlam Kontrolü:**
    - positive_degil_control ve negative_degil_control:
        - Pozitif veya negatif bir kelimenin "değil" ile olumsuz hale getirilip getirilmediğini kontrol eder.
    - check_before_comma:
        - Virgülden önceki kelimelerin pozitif veya negatif polariteli olup olmadığını kontrol eder.
    - check_before_hic:
        - Bir kelimenin öncesinde "hiç" gibi bağlama dayalı bir kelime olup olmadığını kontrol eder.
    - conjunctions_control:
        - "Ama", "fakat", "oysa" gibi bağlaçların cümlede bulunup bulunmadığını kontrol eder.

3. **Olumsuzluk Eklerinin Ayrımı:**
    - ayirt_et:
        - Kelimenin olumsuzluk eki mi (ör. "-ma", "-me") yoksa isim-fiil eki mi olduğunu ayırt eder.
4. **FSM Durumlarına Özel Kontroller:**
    - ne_ne_control:
        - "Ne ... ne" yapısını tespit eder.
    - end_with_degil:
        - Cümlenin "değil" kelimesiyle bitip bitmediğini kontrol eder.
    - ironic_punctuation:
        - Cümlede ironik bir yapı olup olmadığını ("(!)") kontrol eder.
5. **Eşit Skor Kararı:**
    - equal_score:
        - Pozitif ve negatif skorlar eşit olduğunda, hangi skorun daha anlamlı olduğunu bağlama göre belirler.

3. FSM'nin Çalıştırılması

main.py - run_fsm(fsm, input_data)

- Amaç: FSM'i çalıştırarak cümlenin pozitif mi yoksa negatif mi olduğunu belirlemek.
- İşleyiş:
    1. FSM, "start" durumundan başlar.
    2. Girdi verilerine (ör. positive_degil, negative_beforeComma) göre durum geçişleri yapılır.
    3. Bir son duruma ("Pozitif" veya "Negatif") ulaşılana kadar durum geçişleri devam eder.

4. Performans Değerlendirmesi

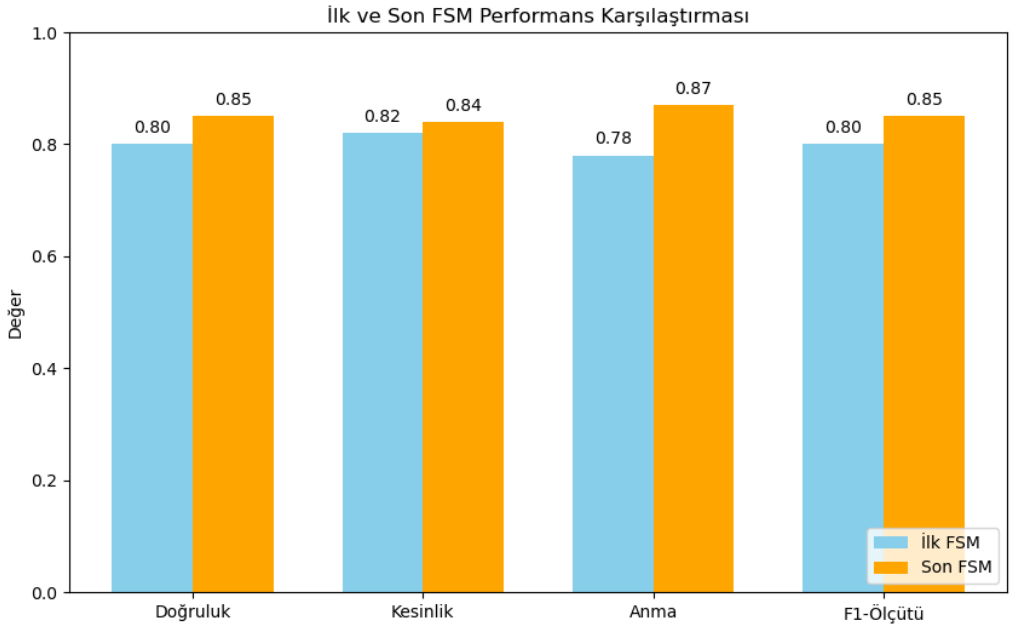
main.py - evaluate_performance(results_df)

- Amaç: FSM'in tahminlerinin performansını ölçmek.
- İşleyiş:
    - Doğru pozitif (DP), yanlış pozitif (YP), yanlış negatif (YN) ve doğru negatif (DN) değerlerini hesaplar.
    - Doğruluk, kesinlik, anma ve F1-ölçütü gibi metrikleri oluşturur.
    - Sonuçları terminalde ve Excel dosyalarında yazdırır.

5) Test ve Analizler

Bu bölümde, geliştirilen modelin farklı veri setleri üzerinde performansı test edilmiş ve sonuçları analiz edilmiştir. Modelin farklı bağlamlarda ve veri setlerinde tutarlılığını değerlendirmek amacıyla doğruluk, kesinlik, anma ve F1-ölçütü metrikleri üzerinden karşılaştırmalar yapılmıştır.

Projedeki örnek verisetinde önceki ve son FSM modelleri arasındaki fark aşağıda verilmiştir:



5.1) Model Validasyonu

Bu işlem, modelin daha önce hiç görmediği bir veri seti üzerinde test edilerek genelleme yeteneğinin değerlendirilmesini ve yanlılığın azaltılmasını içerir, farklı bağlam ve örneklerde ne kadar başarılı olduğunu anlamak için önemlidir.

5.1.1) Dengeli 200 Gözlemlik Veri Seti

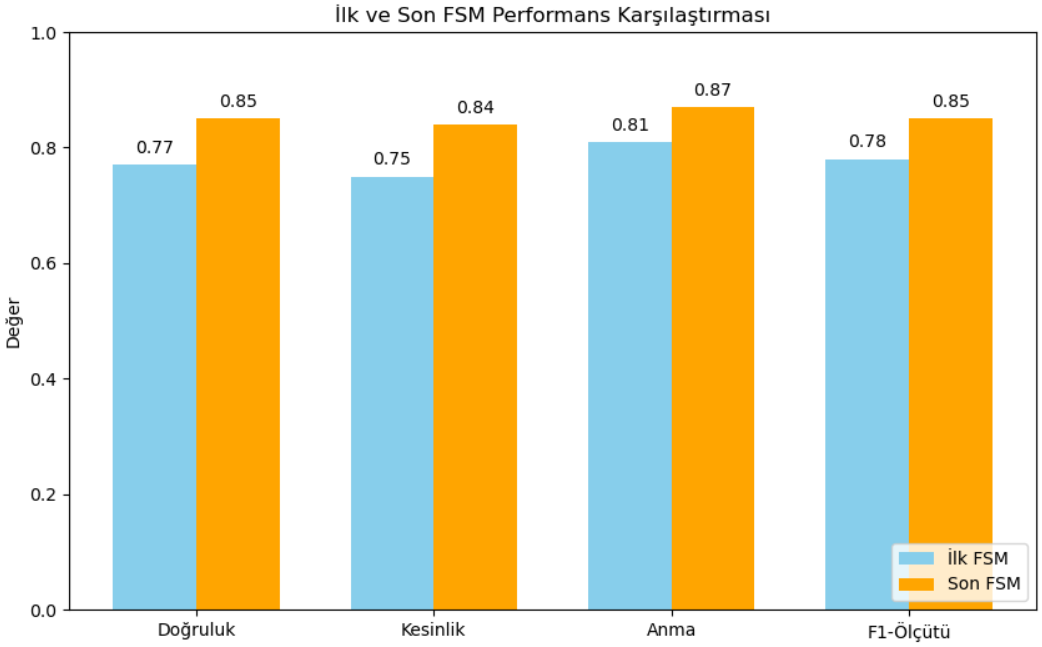
Yaklaşık %50 negatif %50 pozitif gözlem içeren kendi oluşturduğumuz 200 gözlemlik veri setinde iki modeli de deneyerek aşağıdaki sonuçları elde ettik:

FSMV3

Doğru Pozitif (DP): 88	Doğruluk: %87
Yanlış Pozitif (YP): 17	Kesinlik: %84
Yanlış Negatif (YN): 9	Anma: %91
Doğru Negatif (DN): 81	F-1 Ölçütü: %87

FSMV2

Doğru Pozitif (DP): 79	Doğruluk: %77
Yanlış Pozitif (YP): 26	Kesinlik: %75
Yanlış Negatif (YN): 18	Anma: %81
Doğru Negatif (DN): 72	F-1 Ölçütü: %78



5.1.2) Pozitif Yanlı 100 Gözlemli Veri Seti

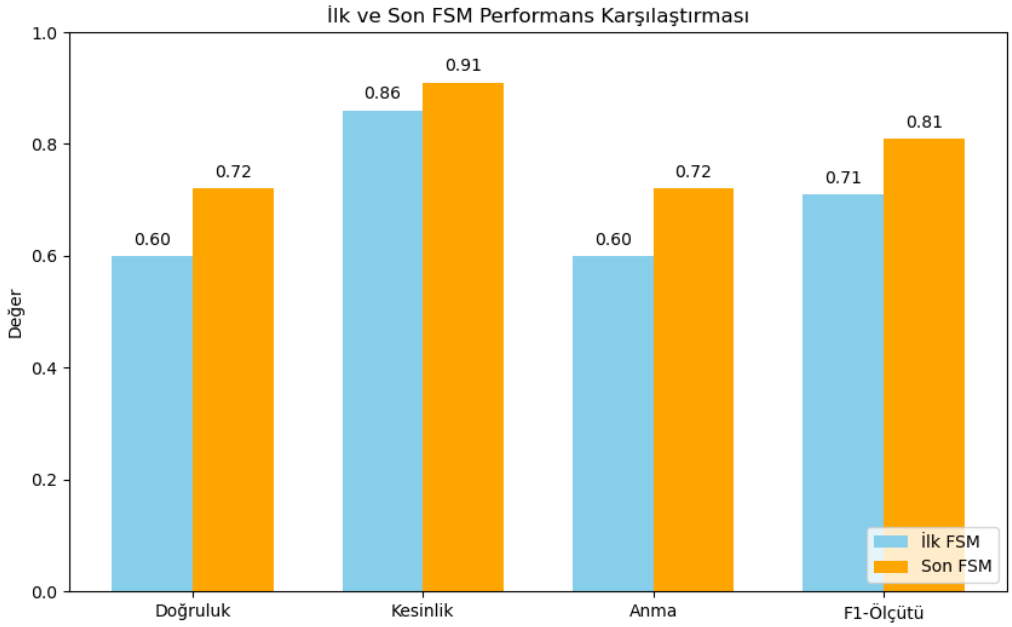
Yaklaşık olarak %20'si negatif %80'si pozitif 100 gözlemli veri setinde iki modeli de deneyerek aşağıdaki sonuçları elde ettik:

FSMV3

Doğru Pozitif (DP): 58	Doğruluk: %72
Yanlış Pozitif (YP): 6	Kesinlik: %91
Yanlış Negatif (YN): 22	Anma: %72
Doğru Negatif (DN): 14	F-1 Ölçütü: %81

FSMV2

Doğru Pozitif (DP): 48	Doğruluk: %60
Yanlış Pozitif (YP): 8	Kesinlik: %86
Yanlış Negatif (YN): 32	Anma: %60
Doğru Negatif (DN): 12	F-1 Ölçütü: %71



5.1.3) Negatif Yanlı 100 Gözlemli Veri Seti

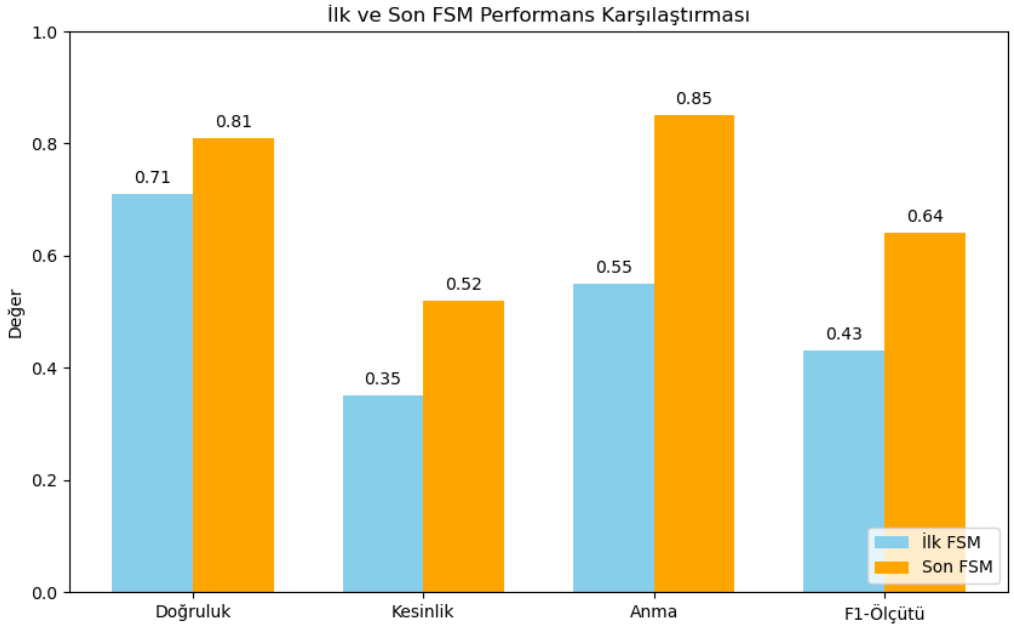
Yaklaşık olarak %20'si pozitif %80'si negatif 100 gözlemli veri setinde iki modeli de deneyerek aşağıdaki sonuçları elde ettik:

FSMV3

Doğru Pozitif (DP): 17	Doğruluk: %80
Yanlış Pozitif (YP): 17	Kesinlik: %50
Yanlış Negatif (YN): 3	Anma: %85
Doğru Negatif (DN): 63	F-1 Ölçütü: %63

FSMV2

Doğru Pozitif (DP): 11	Doğruluk: %71
Yanlış Pozitif (YP): 20	Kesinlik: %35
Yanlış Negatif (YN): 9	Anma: %55
Doğru Negatif (DN): 60	F-1 Ölçütü: %43



Teknik Değerlendirme

FSMv3'ün bu sonuçlara ulaşmasındaki temel faktörler şunlardır:

1. **Bağlama Dayalı Analiz:** FSMv3, "değil" eklerinin cümle sonlarında veya isim fiillerle kullanımını, bağlaç kelimelerini ve "hiç" gibi bağlamı değiştiren kelimeleri doğru analiz edebilmiştir. Bu, modelin bağlam algısının güçlendiğini göstermektedir.
2. **Virgül ve Bölüm Analizi:** Cümle bölümleri arasında pozitif ve negatif bağlamları ayrı ayrı değerlendirme yeteneği, FSMv3'ün özellikle karmaşık ve uzun cümlelerde daha doğru sınıflandırmalar yapmasına olanak tanımıştır.
3. **Genelleme Yeteneği:** FSMv3, farklı veri setlerinde sergilediği tutarlı performans ile overfitting (aşırı öğrenme) problemini azaltmış ve genelleme yeteneğini artırmıştır. Bu, önceki modellerde eksik kalan genelleme sorunlarının giderildiğini göstermektedir.

6) Kaynakça

- Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Toolkit (NLTK)*

https://github.com/nltk/nltk

- Zeyrek NLP. (n.d.). *Zeyrek Morphological Analyzer*

https://github.com/obulat/zeyrek

- Python Software Foundation. (n.d.). *string — Common string operations*

https://docs.python.org/3/library/string.html

- The pandas development team. (2020). *pandas*

https://github.com/pandas-dev/pandas

- The Python Software Foundation. (n.d.). *openpyxl*

https://github.com/chronossc/openpyxl

- Python Software Foundation. (n.d.). *re — Regular expression operations*

https://docs.python.org/3/library/re.html

- Deniz Yüret Türkçe Doğal Dil İşleme

http://www.denizyuret.com/2006/11/turkish-resources.html

- Turkish Textbook

https://www.turkishtextbook.com/most-common-words/

PROJE SUNUSU VE İLGİLİ DRIVE LİNKİ:

Grup4-Drive