R Programlama ile Metin Madenciliği

Hazırlayanlar

Raif Burak DURMAZ 121517047

Ayşe Şevval YURTSEVEN 121516057

Mehtap FİL 121516020

İçindekiler

METİN MADENCİLİĞİ NEDİR?

Metin madenciliği çalışmaları metni veri kaynağı olarak kabul eden veri madenciliği (İng. data mining) çalışmasıdır. Diğer bir tanımla metin üzerinden yapısallaştırılmış veri elde etmeyi amaçlar. Metin madenciliği, metinlerin sınıflandırılması, kümelendirilmesi (İng. clustering), metinlerden konu çıkarılması (İng. concept/entity extraction), metinler için sınıf taneciklerinin üretilmesi (İng. production of granular taxonomy), metinlerde duygu analizi yapılması (İng. sentimental analysis), metin özetlerinin çıkarılması (İng. document summarization) ve metinin özü ile ilgili ilişki modellemesi (İng. entity relationship modelling) gibi çalışmaları hedefler.

Yukarıdaki hedeflere ulaşılması için metin madenciliği çalışmaları kapsamında enformasyon getirimi (İng. information retrieval), hece analizi (İng. lexical analysis), kelime frekans dağılımı (İng. Word requency distribution), örüntü tanıma (İng. pattern recognition), etiketleme (İng. tagging), enformasyon çıkarımı (İng. information extraction), veri madenciliği (İng. data mining) ve hatta görselleştirme (İng. visualization) gibi yöntemleri kullanmaktadır.

Metin madenciliği çalışmaları, metin kaynaklı literatürdeki diğer bir çalışma alanı olan doğal dil işleme (İng. natural language processing, NLP) çalışmaları ile çoğu zaman beraber yürütülmektedir. Doğal dil işleme çalışmaları daha çok yapay zeka altındaki dil bilim bilgisine dayalı çalışmaları kapsamaktadır. Metin madenciliği çalışmaları ise daha çok istatistiksel olarak metin üzerinden sonuçlara ulaşmayı hedefler. Metin madenciliği çalışmaları sırasında çoğu zaman doğal dil işleme kullanılarak özellik çıkarımı da yapılmaktadır.

METİN MADENCİLİĞİNİN ÇALIŞMA ALANLARI:

Metin madenciliği sırasında genelde aşağıdaki problemlerle ilgilenilir (bunlarla sınırlı değildir).

Enformasyon Getirimi (Information Retrieval): Bu aşama ilgilenilen külliyet (derlem, corpus) hakkında ön bilginin toplandığı aşamadır. Örneğin metin madenciliği web üzerindeki veri kaynakları üzerinde yapılacaksa web sayfaları, adresleri veya dosya sistemi üzerindeyse dosyaların tarihleri, kullanıcı bilgileri, dosya isimleri, dizin bilgileri gibi bilgilerin toplandığı aşamadır.

Doğal dil işleme aşaması (natural language processing): Bu aşama bütün metin madenciliği aşamalarında kullanılmasa bile genelde özellik çıkarımı ve metinden bazı anlamsal bilgilerin elde edilmesinde sıklıkla başvurulan aşamadır. Örneğin, konuşma parçalarının etiketlenmesi (part of speech tagging) veya cümle bilimsel parçalama (syntactic parsing) veya diğer dil bilimsel işlemler doğal dil işleme aşamasında yapılır.

Adlandırılmış varlık tanıma (named entity recognition): Genellikle metin işleme aşamasında istatistiksel bazı özelliklerin çıkarılması için kullanılır. Örneğin, metnin içerisindeki kişi isimleri, yer isimleri, semboller, kısaltmalar v.s. bu yöntemle bulunur. Metin madenciliği çalışmalarının her zaman temiz metinlerde yapılmadığını hatırlatmakta yarar vardır. Örneğin facebook, twitter mesajları, telefonlardan yollanan SMS mesajları gibi mesajların çoğunda yazım hataları hatta kısaltmalar kullanılmaktadır. Metin madenciliği bu ihtimallerin de göz önünde tutulması gereken çalışmalardır. Örneğin ''osmanbey'' kelimesi, istanbulda bir semt ismi olabileceği gibi bir kişi ismi de olabilir. Adlandırılmış varlık tanıma çalışmalarında, hedeflenen kelime gruplarının metin içerisinden çıkarılması, sayılması, yoğunluğunun bulunması, etiketlenmesi gibi işlemler yapılabilir.

Örüntüsü tanımlı varlıkların bulunması (pattern identified entities): Bazı durumlarda, metnin içerisinden özel bazı bilgilerin metin madenciliğine konu olması mümkündür. Örneğin e-posta adresleri, telefon numaraları, adresler, tarihler gibi bazı bilgileri özel olarak almak isteyebiliriz. Genelde bu durumlarda düzenli ifadeler (regular expressions) veya içerik bağımsız gramerler (context free grammers) tanımlanarak metin üzerinde çalıştırılır.

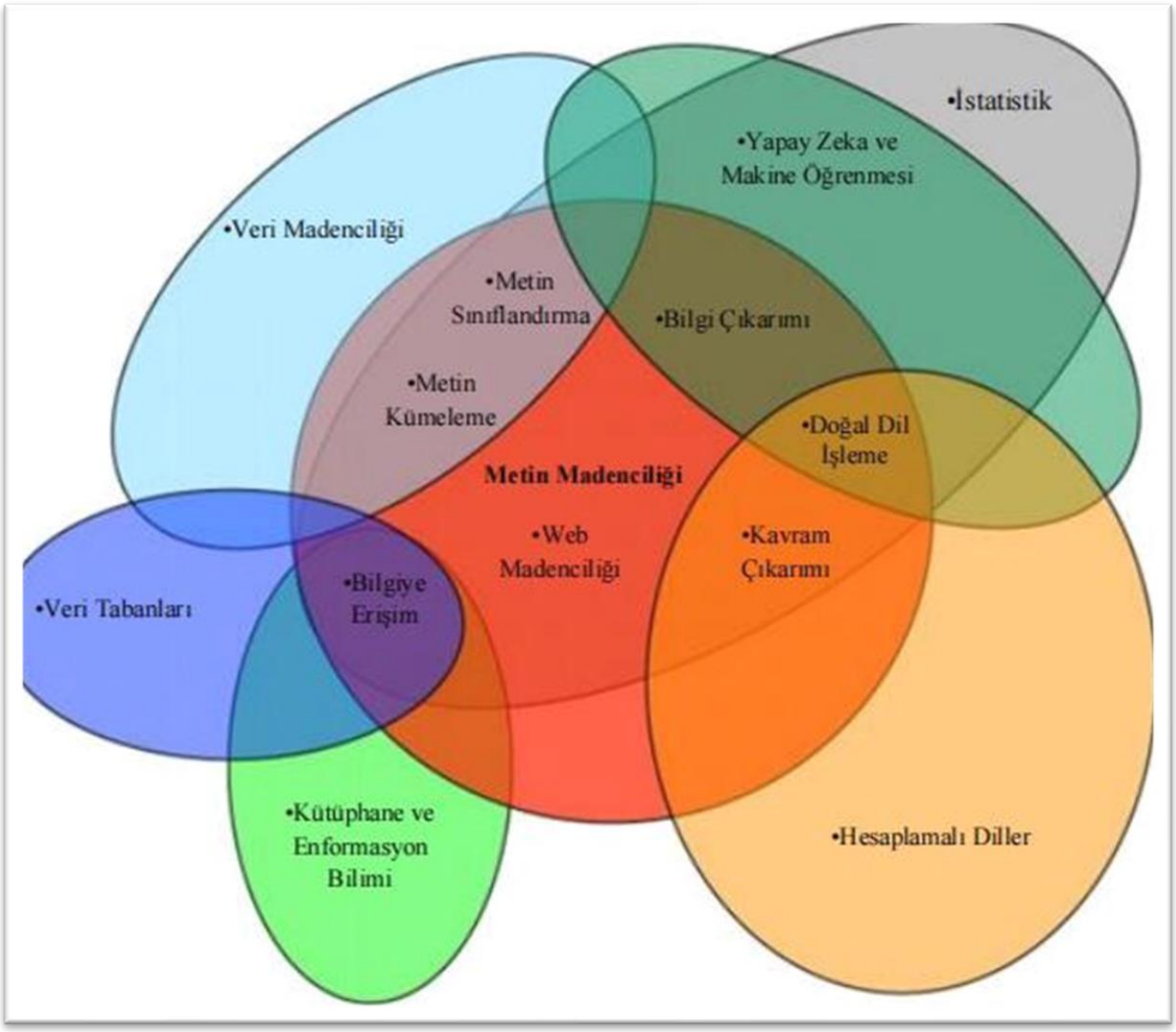
Eş Atıf (Coreference): Bir varlığa işaret eden (atıf eden) isim kelime gruplarını ve diğer terimlerin bulunması/ayrılmasını hedefler.

İlişki, kural, olay çıkarımları: Çeşitli amaçlarla metnin içerisinden bazı bilgilerin çıkarılması istenebilir.

Duygu analizi (sentimental Analysis) : Metinlerde geçen duygusal ifadelerin çıkarılmasını amaçlar. En sık kullanılanı duygusal kutupsallıktır (sentimental polarity). Buna göre bir konu hakkında geçen mesajların veya yazıların olumlu veya olumsuz olmasına göre iki sınıfa ayrılması hedeflenir. Ancak duygu analizi bunun dışında, metinlerdeki ruh hali, kanaat ve daha karmaşık duyguların çıkarılması üzerinde de çalışmaktadır.

Metin madenciliği disiplinler arası bir çalışma alanı olup enformasyon getirimi, veri madenciliği, makine öğrenmesi, istatistik ve işlemsel dilbilim kavramlarının ortak çalışma alanıdır. Günümüzde enformasyonun büyük kısmı (bilimsel tahminlere göre %80'lik kısmı) metin olarak tutulmaktadır. Bu yüzden metin madenciliği çalışmalarının yüksek ekonomik değeri olduğu ve olacağı söylenebilir. Ayrıca çok dilli veri madenciliği gibi farklı dillerden aynı özellikleri taşıyan değerlerin çıkarılması da güncel konulardandır.

Aşağıdaki görselde metin madenciliğinin ilişkili olduğu disiplinler ve yöntemler yer almaktadır.



Metin Madenciliğinin Adımları

Büyük miktardaki metinsel verilerden potansiyel olarak yararlı ve önceden bilinmeyen belirli bir önemi olan bilginin çıkarılması olarak nitelendirilen Metin Madenciliği şekilde görüldüğü gibi temelde altı adımdan oluşmaktadır

Metin	Metin Ön İşleme	Metin Dönüşümü	Özellik Seçimi	Veri Madenciliği/ Bilgi Keşfi	Yorum/ Değerlendirme
	Söz dizimsel/ Semantik analiz Sözcük türü etiketleme Kelime anlamı belirginleştirme Ayrıştırma(parsing)	Kelime torbası, Kelimeler Kök bulma, Durdurma kelimeleri	Basit hesaplama İstatistik (boyut azaltma, ilişkisiz özellikler)	Sınıflandırma (danışmanlı) Kümeleme (Danışmansız)	Analiz Sonuçları

BİLGİYE ERİŞİM

Bilgiye Erişim kavramı ilk kez Calvin Mooers tarafından 1948 yılında "Application of Random Codes to the Gathering of Statistical Information" başlığını taşıyan yüksek lisans tezinde Information Retrieval terimi altında kullanılmıştır. Vickery, Mooers'in kavrama İngilizce olarak getirdiği ilk tanımı şu şekilde aktarır. Bilginin bir depodan özelliklerine göre konusal olarak aranarak erişilmesidir.

Bilgiye Erişim (IR), metin madenciliğinde ilk adımdır. IR'nin amacı kullanıcıların bilgi ihtiyaçlarını karşılayacak olan belgeleri bulmasına yardımcı olmaktır.

Bilgiye Erişim sistemlerinde ağırlık(w) verme önemli bir rol oynar ve birçok farklı ağırlık verme modeli geliştirilmiştir. En yaygın olarak kullanılan model, yerel(local) ve genel(global) ağırlık verme şemalarının bir arada kullanılmasıdır.

Yerel ağırlık vermede terim frekansı (Term Frequency, TF), genel ağırlık vermede ise ters doküman frekansı (Inverse Document Frequency,IDF) kullanılır.

Terim Frekansı (TF), bir doküman içerisinde bir terimin tekrar sıklığıdır. Ters Doküman Frekansı (IDF) bir terimin bütün doküman koleksiyonu (D) içindeki önemidir.

Bu modele göre, terimin önemi, belge içerisinde o terimin geçme sayısıyla doğru orantılıyken;

bütün belge havuzu içerisinde o terimin geçme sıklığıyla ters orantılıdır. D belgesinde, i teriminin ağırlığı şu şekilde hesaplanır:

$$w_i = tf_i \times \log \frac{D}{df_i}$$

Frekansı düşük olan terimler için IDF skoru yüksek, frekansı yüksek olan terimler için IDF skoru düşüktür. TF-IDF değeri, az miktarda doküman içerisinde terim yüksek miktarda geçiyor ise yüksek değer alır. Eğer terim her dokümanda geçiyorsa TF-IDF değeri en düşük değerini alır.

METİN ÖNİŞLEME:

Metni kelimelere ayırma, kelimelerin anlamsal değerlerini bulma (isim, sıfat, fiil, zarf, zamir vb.), kelimeleri köklerine ayırma ve gereksiz kelimeleri ayıklama, yazım kurallarına uygunluğunu tespit etmek ve var olan hataları düzeltmek gibi metin belgelerin yapıtaşı olan kelimelerle ilgili işlemleri içeren süreçtir.

Metin madenciliğinin en büyük sorunu işleyeceği veri kümesinin yapısal olmamasıdır. Genellikle doğal dil kullanılarak yazılmış dokümanlar üzerinde çalışılan metin madenciliği alanında ön işleme aşaması veri temizlemenin yanında veriyi uygun formata getirme işlemini de gerçekleştirmektedir.

Belgeler için dizin oluşturmadan önce yapılacak ön işleme işlemleri şöyledir.

- Doküman doğrusallaştırma
- ✓ Markup & Format Removal: Dokümanı oluşturan etiket ve özel formatların çıkarılması
- ✓ Tokenization: Metin küçük harflere çevrilmesi ve noktalama işaretlerinin çıkarılması

Metin önişleme çalışmaları aynı zamanda doğal dil işleme çalışmaları kapsamında incelenen bir alandır. Doğal dil işlemenin belge analizi sürecindeki en önemli faydası terimlerin yani kelimelerin ayrıştırılması, eklerinden arındırılarak anlamını kaybetmeyen en kısa biçimlerine dönüştürülmesidir. Çünkü aynı anlam için kullanılan kelimeler dilbilgisi kuralları gereği farklı biçimlerde bulunabilir ve bu farklı kullanım biçimleri ortadan kaldırılmadığı takdirde farklı anlam taşıyan terimler gibi işleme alınarak, belgelerin gerçek anlamına ulaşılmasını engelleyebilirler. Doğal dil işleme çalışmaları kapsamında yürütülen girişimler dört ana grup altında toplanabilir.

a) Morfolojik Analiz: Biçimbirim, sözcüklerin yapısıyla ile ilgilenir. Her dilde iki farklı şekilde sözcük oluşturulabilir. Bunlardan biri çekim, diğeri ise türetme yöntemidir. Çekim yoluyla sözcük oluşturulurken bir sözcüğün farklı şekilleri kullanılır. Türetme ise var olan eski sözcüklere yapım ekleri eklenmesi yoluyla yeni sözcük oluşturma yöntemidir

b) Sözdizimsel Analiz: Bilgisayarla doğal dil modellemelerinde anlamsal analize geçmeden önce, kelimeler yığınının geçerli bir cümle yapısı oluşturup oluşturmadığı kontrol edilmelidir. Rasgele kelimelerin yan yana gelmesiyle geçerli bir cümle meydana gelmeyecektir. Geçerli bir cümle yapısı oluşturulamadığı zaman, buradan anlam çıkarılmasını beklemek yanlış olacaktır

Sözdizimsel analiz, cümlenin yapısal bir tanımını oluşturabilmek için morfolojik analizin sonuçlarını kullanır. Bu işlemi yapmanın amacı, ardı ardına gelen kelime yığınlarının bu kelimeler yığınının ifade ettiği cümle birimlerini tanımlayan bir yapıya dönüştürmektir. Cümle birimleri, kelimeler tamlamalar veya buna benzer cümle parçacıkları olabilir.

c) Semantik Analiz: Bir cümlenin ne demek istediğinin anlaşılması, diğer bir deyişle bir cümle ile ifade edilmek istenilen duygu veya düşüncenin ne olduğunun anlaşılması, anlamsal analiz yardımıyla yapılır.

d) Anlam kargaşasının giderilmesi: Anlamsal analiz yapılırken, öncelikli olarak kelimelerin tek tek veritabanından uygun nesnelerle eşleştirilme işleminin yapılması gerekir. Bu işlem, her zaman birebir eşleme olamayabilir. Diğer bir deyişle, kelimelerin ifade ettikleri anlamlar her zaman bir tane olmayabilir. Ayrık kelimelerin bir cümledeki doğru anlamını bulma işlemine “kelime anlam berraklaştırılması” denir.

Bu işlem, cümle içinde geçen bir kelimenin sözlükteki anlamlarının belirlenip bunlardan uygun olanının seçilmesidir. Cümle içinde geçen her bir kelime, diğer kelimelerin doğru anlamlarının ortaya çıkarılması için önem taşımaktadır.

Metin dönüşümü

Kelimelerin düzgün bir biçimde hecelerine ve eklerine ayrılmasından sonraki işlem, kelimelerin kökünün tespit edilmesidir. İngilizce için Porter Stemmer Yöntemi gibi kök bulma algoritmaları kullanılmaktadır.

Kelime Türü: Bir kelimenin kökü bulunduktan sonraki adım kelimenin türünün bulunmasıdır. Bu işleme Pos Tagging denir. Pos Tagging 2 fazdan oluşur. Birincisi eğitim(training) fazıdır. Bu fazda kelimelerin kökleri manuel olarak tanımlanmış algoritmalar kullanılarak machine learning sistemi vasıtasıyla işlenir. İkinci faz ise tagging fazıdır. Bu fazda, birinci adımda kullanılan algoritma, öğrenilen parametrelere göre yeniden işlenir ve kelimeler türlerine ayrılır.

Stopword İşlemi: Tekrar eden ve tek başına anlam taşımayan kelimelere stopword kelimeleri denir. Bilgiye Erişimde bir stopword listesi, belgeleri bir diğerinden ayırt etme durumuna etkisi olmayan sıklıkla kullanılan kelimeleri içerir. Stopword kelimelerini azaltmak sorgu sürecinin verimini arttırır. Bir stopword listesinin yapılandırması farklı ve bazen rastgele kararları içerir. Bilgiye Erişim literatüründe, verilen özel diller için farklı uzunluklarda stopword listelerini bulmak mümkündür.

Bag of words: Bu aşamada gruplanan tüm dokümanlardaki tüm kelimelerin kullanım sıklıkları hesaplanır ve bir havuzda toplanır. Daha sonrasında ise bu kelimelerin değerleri (Word Weighting) hesaplanır. Kelime değeri, bir kelimenin belirli bir alan (sağlık, spor, politika, ...) ile ilgili bir metnin içinde bulunma sıklığı olarak açıklanabilir. Örneğin 10000 kelimelik spor kategorisindeki bir haberin içinde gol veya hakem kelimelerinin bulunma sıklıkları, aynı kelimelerin sağlık kategorisindeki bir haber içinde bulunma sıklığına göre kat ve kat fazladır (ORHAN, 2006; KARADENİZ, 2007).

Özellik seçme

Metin madenciliği uygulamalarında her zaman gürültülü ve önemsiz bilgi içeren metin koleksiyonlarıyla uğraşma ihtiyacı bulunmaktadır. İlgili verilerin saptanması üzerine odaklanan özellik seçme, büyük miktarlardaki veriler üzerinde işlem yapılırken iş yükünü azaltmada yardımcı olmaktadır. Özellik seçme aşamasında, ön işlemden geçen metinlerdeki önemli kelimeleri (varlıkları) belirleme (isimler, tamlamalar, bileşik kelimeler, kısaltmalar, sayılar, tarihler, para birimleri vb.) ve ilişkili olmayan özelliklerin çıkarılması, sadece birkaç dokümanda gözlemlenen özelliklerin çıkarılması, birçok dokumanda gözlemlenen özellikleri azaltma vb. işlemleri yapılmaktadır

Veri Madenciliği/Bilgi Keşfi

Metinsel verilerden bilgi keşfi için veri madenciliğinde geçen Sınıflandırma ve Kümeleme yöntemleri kullanılabilir. Sınıflandırma yöntemleri şu şekilde özetlenebilir.

- Entropiye Dayalı algoritmalar (ID3, C4.5)
- Sınıflandırma ve Karar Ağaçları (Twoing, Gini,)
- Bellek tabanlı sınıflandırma modelleri (En yakın komşu algoritması)
- Optimizasyon tabanlı Sınıflandırma Modelleri (Destek Vektör Makinesi)
- İstatistiksel Sınıflandırma Modelleri (Navie Bayes)

Kümeleme yöntemleri de aşağıda sıralandığı gibi özetlenebilir.

- Hiyerarşik Metotlar (En yakın komşu algoritması, En uzak komşu algoritması,)
- Hiyerarşik olmayan Metotlar (k-ortalamalar)

Veri Madenciliği için belirtilen bu sınıflandırma ve kümeleme yöntemleri ön işleme adımlarından geçirilerek metinsel verilere uygulanmaktadır

ZİPF KANUNU NEDİR?

Zipf kanunu, 1930'lu yıllarda Amerika'da Harvard Üniversitesi'nde dilbilim profesörü olan George Kingsley Zipf, tarafından geliştirilmiştir. Bu yüzden onun adı ile anılan yasa, istatistiki bir durumun deneysel keşfine dayanmaktadır.

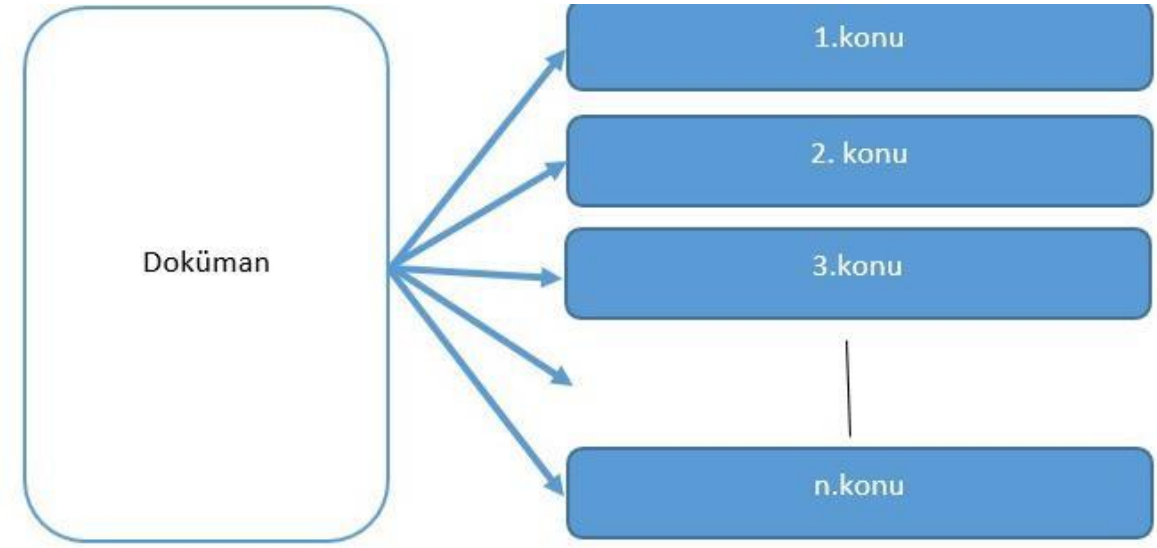
Basit Tarif

Hangi dilde olursa olsun herhangi bir metinde geçen sözcükler, kullanılma sıklığına göre sıralandığında oluşan listede her zaman geçerli bir kural vardır. Sıralama listesindeki sözcüklerin sıra numarası ile o sözcüğün sıklık sayısı sabit bir sayıdır. Bu kuralı formülle ifade edecek olursak; N sırasında yer alan sözcük, o metinde 1/N defa tekrarlanmıştır. Yani bu; listenin birinci sırasındaki sözcük, ikinci sıradaki sözcükten 2, üçüncü sıradaki sözcükten 3 kat daha fazla kullanılmış olduğu anlamına gelir.

KONU MODELLEME (TOPIC MODELLING)

Konu modelleme (Topic Modelling), metin içeren dokümanın anlamsal yapısını belirleyen bir makine öğrenmesi yöntemdir. Ayrıca doğal dil işleme araştırma alanı olarak gösterilmektedir.

Konu modelleme yöntemleri, yüksek içerikli metin belgelerini organize edilebilmekte ve özetleyebilmektedir. Konu modelleme, otomatik belge indeksleme, doküman sınıflandırma, konu keşfi gibi birçok alanda başarıyla uygulanabilmektedir.



Doküman, konuların birleşimi olarak gösterilebilir. Konular, kelimeler üzerinde bir olasılık dağılımı olarak ; dokümanda konular üzerinde bir olasılık dağılımı olarak hesaplanmaktadır.

LATENT DIRICHLET ALLOCATION ALGORITMASI

LDA, olasılık tabanlı bir konu modelleme yöntemidir. Model bir dizi dokümandan kelime ağırlığına dayalı olarak konuları oluşturmaktadır. LDA'nın temelinde, konular kelimeler üzerinde bir olasılık dağılımına, metin belgeleri de konular üzerinde bir olasılık dağılımına sahiptir. Her bir konunun ise kelime dizisi üzerinde bir dağılımı olmaktadır.

LDA denetimsiz öğrenme algoritmasıdır, önceden tanımlanmış kelimelere ihtiyaç duymamaktadır. Konu sayısı belirlendikten sonra, sınıflara göre konulara etiket atanmaktadır.

Çalışma prensibi;

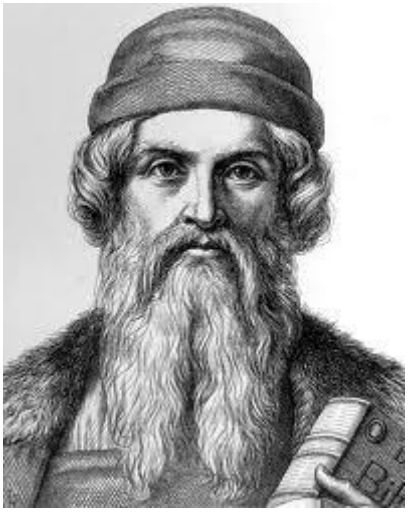
Her doküman için dokümandaki kelimelere rastgele konu ataması yapmaktadır. Bu bilgiyi kullanarak model çeşitli istatistikler çıkartmaktadır.

Yerel istatistik, her dokümandaki konulara kaç adet kelime atandığını gösterirken, global istatistik ise tüm doküman için her kelimenin her konuya kaç kere atandığını göstermektedir.

İstatistiksel bilgiler elde edildikten sonra her doküman için her kelimenin yeniden konu ataması gerçekleştirilir. Bunun için mevcut kelime bilgileri sürekli iterasyon sayısı kadar güncellenmelidir.

JOHANNES GUTENBERG

Johannes Gutenberg (1400-1468) baskı makinesinin mucidi olmakla beraber baskı tekniğinin babasıdır. Baskı makinesini kullanılır hale getirmiştir. Onun bu başarısı medeniyetin ilerlemesinde, gelişmesinde çok önemli rol oynamıştır.



Johannes, aileden bir çocuğun annesiyle ilgili bir adı taşıması geleneğine uyarak annesinin doğduğu şehrin adını kendine soyadı yapmıştır. Babası eski altın paralar üzerine şekiller yaparak para kazanırdı. Gutenberg'in baskı tekniği merakının bundan doğduğunu ileri sürenler mevcuttur.

Gutenberg'in makinesinde bastığı kitapların sayısı elliyi bulmaktadır. Onun yayınladığı İncil'e «42 Satırlı İncil» denir. Kitabın sayfalarında çoğunlukla 42 satır bulunduğu için bu ad verilmiştir. Gutenberg'in İncil'lerinden birkaç tanesi Amerikan kütüphane ve müzelerinde saklanmaktadır.

GUTENBERG PROJESİ

Gutenberg Projesi (PG) telif hakkı olmayan kültürel eserleri dijital ortama aktarmak, arşivlemek ve dağıtmak amacıyla yürütülen gönüllü bir çalışmadır.

1971 yılında kurulan Gutenberg Projesi, dünyanın en eski dijital kütüphanesidir; İnternet'in doğmasından sonra web ortamına taşınmıştır. Proje, özellikle edebiyat eserlerinin olabildiğince ücretsiz ve kalıcı bir biçimde, herhangi bir bilgisayarda kullanılabilen açık formatlarda serbestçe dağıtılmalarını sağlar. Gutenberg Projesi kapsamında yer alan eserlere erişmek için http://www.gutenberg.org/ sitesini inceleyebilirsiniz.

Gutenbergr paketi, Gutenberg Projesi koleksiyonundaki eserlere erişim sağlar. Paket, hem kitapları indirmek (yardımcı olmayan üstbilgi/altbilgi bilgilerini çıkarmak) hem de ilgi çekici eserleri bulmak için kullanılabilecek Gutenberg Projesi meta verilerinin tam bir veri kümesini içerir.

METIN MADENCILIGI YONTEMLERI ILE ESER VE YAZAR ANALIZI

Projemiz kapsamında metin madenciliği teknikleri ile İngiliz edebiyatının en büyük 2 yazarını incelemeyi tercih ettik. Bunlar William Shakespeare ve Ben Jonson'dir. Bu iki yazarı seçmemizin en büyük sebeplerinden biri bazı araştırmalarda Ben Jonson'un eserlerini kaleme alırken William Shakespeare'in eserlerinde kullandığı dilden, olay örgüsünden ve metin yapısından etkilendiği söylenmektedir. Bu düşünceleri destekleyen en büyük özellik ise bu iki yazarın çok yakın arkadaş olması ve yaşadıkları dönem içerisinde bu tarz dedikoduların çok fazla gündemde yer almasıdır.

William Shakespeare (26 Nisan 1564 – 23 Nisan 1616), İngiliz şair, oyun yazarı ve oyuncudur. Kendisi İngiliz dilinin en büyük yazarı ve dünyanın en iyi dram oyun yazarı olarak anılmaktadır. İngiltere'nin ulusal şairi ve "Avon'un Ozanı" olarak da bilinir.

Shakespeare'in kişisel yaşamına dair bazı kayıtlar günümüze ulaşmıştır. Fiziksel görünüşü, cinsel yönelimi, dini inançları, ve başkaları tarafından yazılıp ona atfedilen eserler olup olmadığı hakkında önemli tahminler yürütülmüştür.

Ben Jonson, (11 Haziran 1572 - 6 Ağustos 1637),

17. yüzyıl İngiliz oyun yazarı, şair, oyuncu ve eleştirmen.

İngiltere'nin ikinci büyük tiyatro adamı (Shakespeare'dan sonra) olarak kabul edilir.

Yaşamının İngiltere'de oyuncu ve yazar olarak adını duyurana kadarki bölümü hakkında pek fazla bir şey bilinmemektedir.

Ben Jonson 1604 yılında I. James tarafından saraya çağrılmış bu işte 20 yılını harcamıştır, fakat sarayda gözden düşünce yeni komediler yazmaya karar vermişse de eski gücünü yitirdiğini fark etmiştir.

William Shakespeare

Gutenberg'den Metin Verisi Cekmek

```
library(gutenbergr)

## Warning: package 'gutenbergr' was built under R version 4.0.4

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

shakespeare_eserleri<- gutenberg_metadata %>%
  filter(author %in% "Shakespeare, William",
         language == "en")

shakespeare<- gutenberg_download(c(1515,1508,1526))

## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest

## Using mirror http://aleph.gutenberg.org
```

Burada filtreleme islemi ile gutenberg kütüphanesinde Shakespeare'e ait olan 3 adet komedi türündeki eserlerini çektik.

```
basliklar<-shakespeare_eserleri %>%
  select(gutenberg_id, title)

shakespeare <- shakespeare %>%
  inner_join(basliklar, by = "gutenberg_id")
```

Yukarida gutenberg id numarasina gore cektigimiz eserlere baslik atadik. Bu sayede kitaplari ayirt edebiliriz.

PARÇALAMA İŞLEMLERİ

```
library(tidytext)

## Warning: package 'tidytext' was built under R version 4.0.5
```

```
shakespeare1<-shakespeare %>%
  unnest_tokens(word, text) %>%
  count(word, sort = TRUE)
shakespeare1
```

```
## # A tibble: 6,550 x 2
##    word      n
##    <chr> <int>
##  1 and    1917
##  2 the    1910
##  3 ı      1883
##  4 to     1438
##  5 you    1401
##  6 a      1315
##  7 of     1135
##  8 my     1021
##  9 that    764
## 10 in      751
## # ... with 6,540 more rows
```

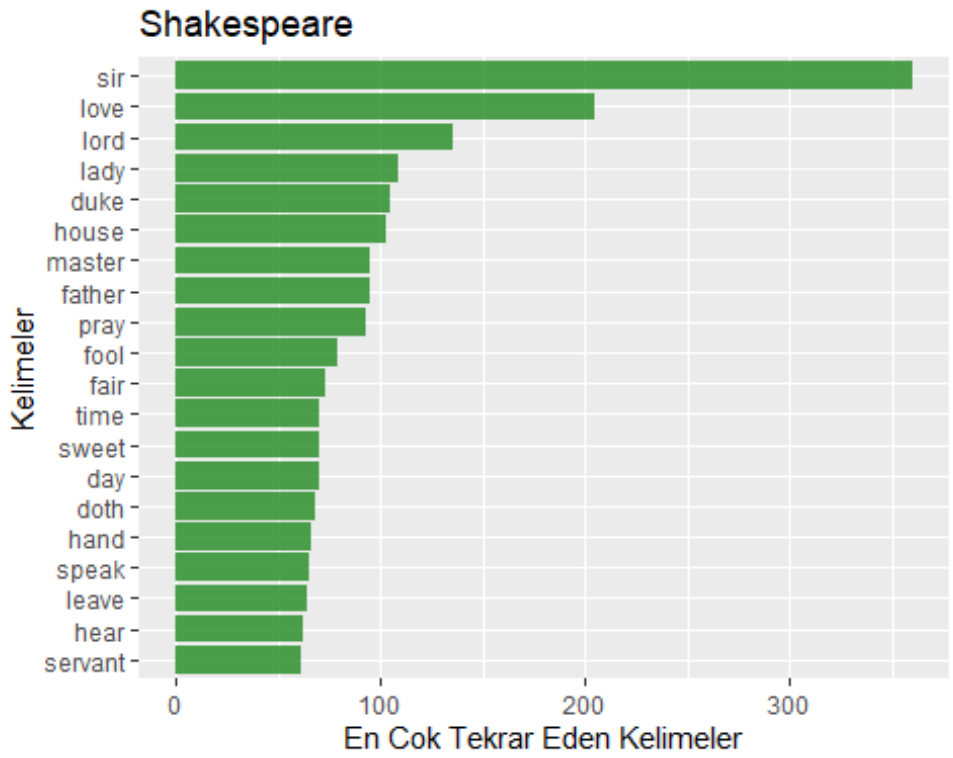
Burada unnest_tokens fonksiyonu ile metindeki her bir kelimeyi parçaladık. Kelime frekanslarına bakmak icin count fonksiyonunu kullandık. Burada her bir kelimenin metin icerisinde kac kez gectigini göruyoruz. Burada gördüğümüz gibi en fazla tekrar eden kelimeler anlamsız görünüyor. Burada baktığımızda 57. gözlemde Petruchio karakterinin geldiğini görüyoruz. Yani ilk 56 gözlem bizim için bir anlam ifade etmiyor. Daha sonra 77. gözlemde Viola karakterini görüyoruz. Yani aslında Petruchio karakterini dışarda bıraktığımızda ilk 76 gözlemin bizim için anlamsiz olduğu görülüyor. Bu durumda metinde cok fazla tekrar eden ama tek başına bir anlam ifade etmeyen kelimelere duraklama kelimeleri denir. Duraklama kelimelerini metin içerisinden çıkarmamız gerekiyor.

```
tidyshakespeare<-shakespeare %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE)

## Joining, by = "word"

tidyshakespeare

## # A tibble: 6,059 x 2
##    word          n
##    <chr>     <int>
##  1 ı          1883
##  2 sir         360
##  3 thou        353
##  4 sır         274
##  5 thy         248
##  6 thee        239
##  7 toby        206
##  8 love        205
##  9 petruchıo   176
## 10 enter       174
## # ... with 6,049 more rows
```

Şu anda tek başına anlamsız görünen kelimeler veri setimizden çıkarttık ve verimiz tidy hale geldi. Verimiz duzenli metin formatına dönüşmüş oldu. Metin eski dilde yazıldığından ötürü stopwords olarak algılanmayan bazı kelimeler de mevcut. Örnegin thy:sizin gibi. Bunlari da

temizlememiz gerekli. Yazarın eserleri analiz edileceğinden karakter isimlerini de veri setinin dısında bırakmalıyız.

```
mystopwords<-tibble(word = c("ı", "thou", "sır", "thy", "thee", "ı'll", "ıf", "hath", "tis"
, "ın", "ay", "mine", "ıt", "ıs", "la", "toby", "petruchıo", "enter", "vıola", "olıvıa", "p
ortıa", "clown","andrew", "tranıo", "shylock", "katherına", "malvolıo", "antonıo", "exit",
"hortensıo", "bassanıo", "lucentıo", "launcelot", "baptısta", "lorenzo", "marıa", "grumıo",
"exeunt", "gremıo", "fabıan", "kate", "mos","jonson", "cler",  "volp", "cokes", "daw", "mor
", "daup", "nay","quar", "waspe", "lıt", "avoc", "ott", "corv", "corb", "mosca", "john", "j
onson's", "numps", "volt", "wınw", "p", "f", "exıt", "i'faith", "epi","urs"))
tidyshakespeare1<-shakespeare %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords) %>%
  count(word, sort = TRUE)

## Joining, by = "word"
## Joining, by = "word"

tidyshakespeare1

## # A tibble: 6,016 x 2
##    word       n
##    <chr>  <int>
##  1 sir      360
##  2 love     205
##  3 lord     135
##  4 lady     109
##  5 duke     105
##  6 house    103
##  7 father    95
##  8 master    95
##  9 pray      93
## 10 fool      79
## # ... with 6,006 more rows

library(ggplot2)
tidyshakespeare1 %>%
  filter(n > 60) %>%
  ggplot(aes(reorder(word,n),n))+
  geom_col(fill = "forestgreen", alpha = 0.8)+
  coord_flip()+
  labs(x = "Kelimeler",
       y = "En Cok Tekrar Eden Kelimeler",
       title = "Shakespeare")
```



William Shakespeare'in hangi kelimelere sıklıkla yer verdiğini görebilmek adına eserlerinde kullanmış olduğu ve bir anlam ifade etmeyen kelimeleri çıkardıktan sonra amacımıza uygun olması açısından veri setini karakter isimlerinden de arındırdık. Baktığımız zaman asalet ünvanlarının sıkça kullanıldığını görüyoruz. Eserlerinde genellikle aşk temasını işlediğini ve eserlerinde dine yer verdiği gibi çıkarımlarda bulunabiliriz. Komedi türündeki bu eserlerde "fool" kelimesinin de sıklıkla geçtiği görülmektedir. Gülünç durumu yaratan eylemin karakterlerin kandırılmasından kaynaklı olduğu söylenebilir. Bu kandırılma durumundaki bir etkenin aşık olmak olduğu konusunda bir çıkarım yapılabilir.

DUYGU ANALIZI

Metin icerisindeki kelimelerin hangi duygulari icerdigine bakilir. Metin olumlu mu olumsuz mu temel prensibimiz bu. 3 tane temel duygu sözlüğümüz var.

AFINN (Finn Arup Nielsen): -5 ile +5 araliginda kelimeleri skorluyor.

bing (Bing Liu and Collaborators): olumlu olumsuz seklinde kelimeleri ayrıştırıyor.

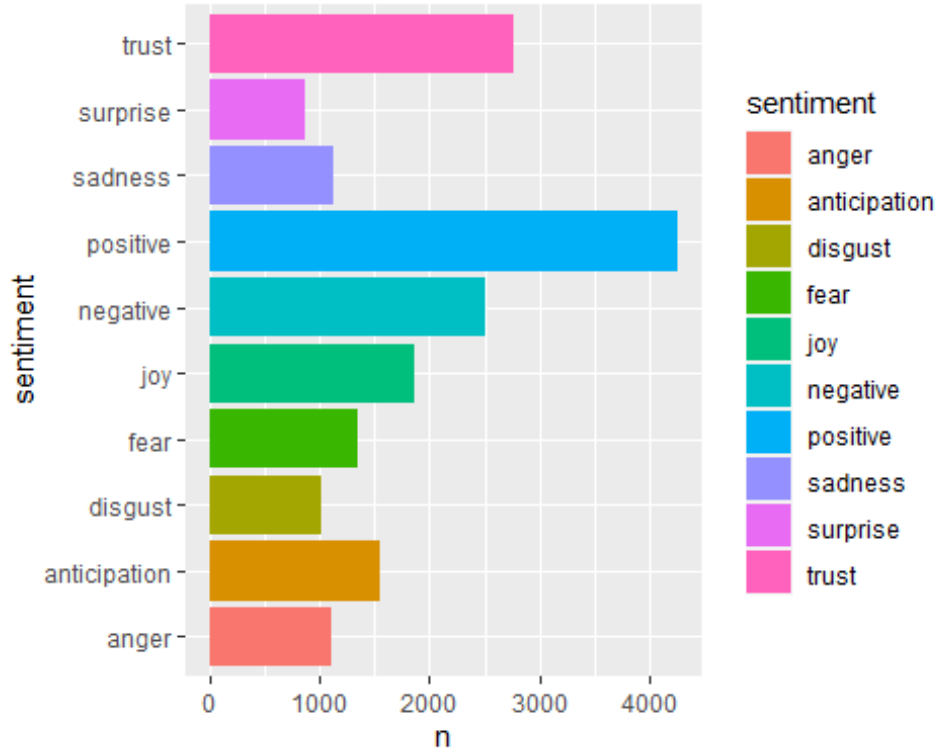
nrc (Saif Mohammad and Peter Turney): birden fazla duygu icerir.

```
tidyshakespeare1 %>%
  inner_join(get_sentiments("nrc"))

## Joining, by = "word"

## # A tibble: 3,195 x 3
##    word       n sentiment
##    <chr>  <int> <chr>
##  1 sir      360 positive
##  2 sir      360 trust
##  3 love     205 joy
```

```
##  4 love     205 positive
##  5 lord     135 disgust
##  6 lord     135 negative
##  7 lord     135 positive
##  8 lord     135 trust
##  9 duke     105 positive
## 10 father    95 trust
## # ... with 3,185 more rows

tidyshakespeare1 %>%
  inner_join(get_sentiments("afinn"))

## Joining, by = "word"

## # A tibble: 555 x 3
##    word      n value
##    <chr> <int> <dbl>
##  1 love    205     3
##  2 pray     93     1
##  3 fool     79    -2
##  4 fair     73     2
##  5 sweet    70     2
##  6 leave    64    -1
##  7 true     55     2
##  8 god      51     1
##  9 faith    48     1
## 10 mad      44    -3
## # ... with 545 more rows

tidyshakespeare1 %>%
  inner_join(get_sentiments("bing"))

## Joining, by = "word"

## # A tibble: 1,020 x 3
##    word         n sentiment
##    <chr>    <int> <chr>
##  1 love       205 positive
##  2 master      95 positive
##  3 fool        79 negative
##  4 fair        73 positive
##  5 sweet       70 positive
##  6 mistress    53 negative
##  7 faith       48 positive
##  8 mad         44 negative
##  9 fear        38 negative
## 10 fortune     34 positive
## # ... with 1,010 more rows
```

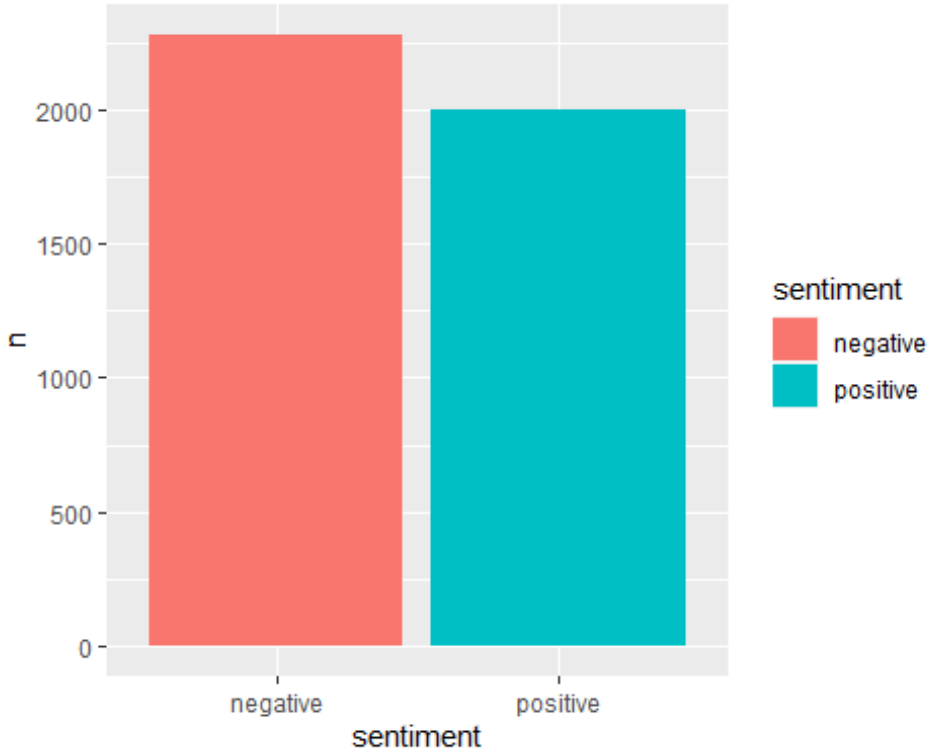
NRC sözlüğüne göre duyguların frekanslarına dayalı bir görselleştirme yapalım.

```
tidyshakespeare1 %>%
  inner_join(get_sentiments("nrc"))%>%
  ggplot(aes(sentiment, n, fill = sentiment))+
  coord_flip()+
  geom_col()

## Joining, by = "word"
```



Yukarıdaki grafikte Shakespeare'in eserlerinin NRC sözlüğüne göre duyguların frekanslarına dayalı bir görselleştirme yapılmıştır. NRC sözlüğünde pozitiften çok negatif duygular yer almaktadır. Bu grafikte pozitif duygular ağır basmıştır. Dolayısıyla Shakespeare'in komedi türüne ait eserlerinin pozitif duygular içerdiği söylenebilir.

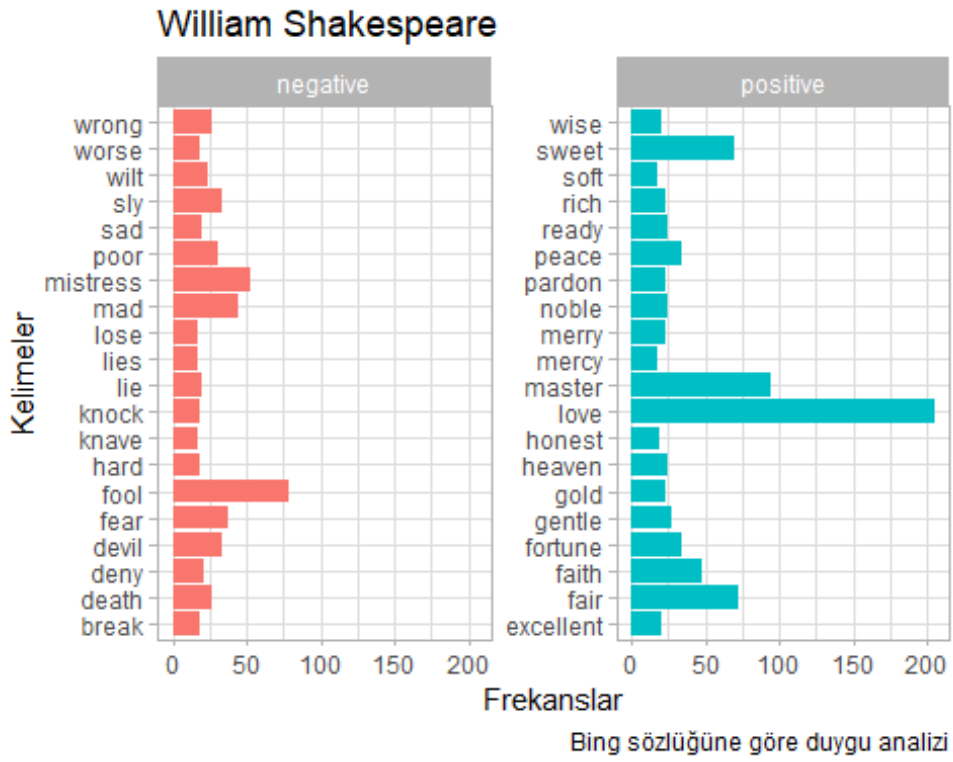
```
tidyshakespeare1 %>%
  inner_join(get_sentiments("bing"))%>%
  ggplot(aes(sentiment, n, fill = sentiment))+
  geom_col()

## Joining, by = "word"
```



Bing sözlüğüne göre ise negatif duygular daha ağır basmıştır.Bu demek oluyor ki Shakespeare'in eserlerinde negatif kelimeler daha fazladır.

Hangi kelimeler daha fazla ön plana çıkmış?

```
rbind(

tidyshakespeare1 %>%
  inner_join(get_sentiments("bing"))%>%
  arrange(-n) %>%
  filter(sentiment == "positive") %>%
  head(20),

tidyshakespeare1 %>%
  inner_join(get_sentiments("bing"))%>%
  arrange(-n) %>%
  filter(sentiment == "negative") %>%
  head(20)
) %>%
  ggplot(aes(word, n, fill = sentiment))+
  coord_flip()+
  facet_wrap(~sentiment, scales = "free_y")+
  geom_col(show.legend = "FALSE")+
  theme_light()+
  labs(x = "Kelimeler",
       y= "Frekanslar",
       title= "William Shakespeare",
       caption = "Bing sözlüğüne göre duygu analizi")

## Joining, by = "word"
## Joining, by = "word"

## Warning: `show.legend` must be a logical vector.
```



Negatif kelimeler arasında “yanlış, kötü, solgun, sinsi, mutsuz, zavallı” gibi kelimelerin ön planda olduğu görülürken bu kelimelerden en ağır basan kelimenin “ahmak” olduğu görülmektedir.Pozitif kelimeler arasında “bilge, tatlı, yumuşak, zengin” gibi kelimelerin ön planda olduğu görülürken bu kelimelerden en ağır basan kelimenin “aşk” olduğu görülmektedir.

```
library(stringr)

shakespeare <- shakespeare %>%
  group_by(title) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text,
                                     regex("^scene [\\divxlc]",
                                           ignore_case = TRUE)))) %>%
  ungroup()
shakespeare
```

```
## # A tibble: 13,637 x 5
##    gutenberg_id text                      title                linenumber chapter
##           <int> <chr>                     <chr>                     <int>   <int>
##  1         1508 "THE TAMING OF THE SHRE~ The Taming of the S~          1       0
##  2         1508 ""                       The Taming of the S~          2       0
##  3         1508 "by William Shakespeare" The Taming of the S~          3       0
##  4         1508 ""                       The Taming of the S~          4       0
##  5         1508 ""                       The Taming of the S~          5       0
##  6         1508 ""                       The Taming of the S~          6       0
##  7         1508 ""                       The Taming of the S~          7       0
##  8         1508 "Dramatis Personae"      The Taming of the S~          8       0
##  9         1508 ""                       The Taming of the S~          9       0
## 10         1508 "Persons in the Inducti~ The Taming of the S~         10       0
## # ... with 13,627 more rows
```

Stringr ile bir veri manipulasyonu islemi yapildi. Daha sonra her bir satira bir satir numarasi ekledik.

```
tidy_shakespeare<- shakespeare %>%
  unnest_tokens(word, text)
tidy_shakespeare
```

```
## # A tibble: 66,016 x 5
##    gutenberg_id title                     linenumber chapter word
##           <int> <chr>                          <int>   <int> <chr>
##  1         1508 The Taming of the Shrew          1       0 the
##  2         1508 The Taming of the Shrew          1       0 tamıng
##  3         1508 The Taming of the Shrew          1       0 of
##  4         1508 The Taming of the Shrew          1       0 the
##  5         1508 The Taming of the Shrew          1       0 shrew
##  6         1508 The Taming of the Shrew          3       0 by
##  7         1508 The Taming of the Shrew          3       0 william
##  8         1508 The Taming of the Shrew          3       0 shakespeare
##  9         1508 The Taming of the Shrew          8       0 dramatis
## 10         1508 The Taming of the Shrew          8       0 personae
## # ... with 66,006 more rows
```

Metini elde ettikten sonra yukarıda incelediğimiz 3 eseri parçaladık. Parçamala işleminden sonra duraklama kelimelerini yani tek başına anlam ifade etmeyen ifadeleri çıkarmamız gerekiyor.

```
data(stop_words)

tidy_shakespeare <- tidy_shakespeare %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords)
```

```
## Joining, by = "word"
## Joining, by = "word"
```

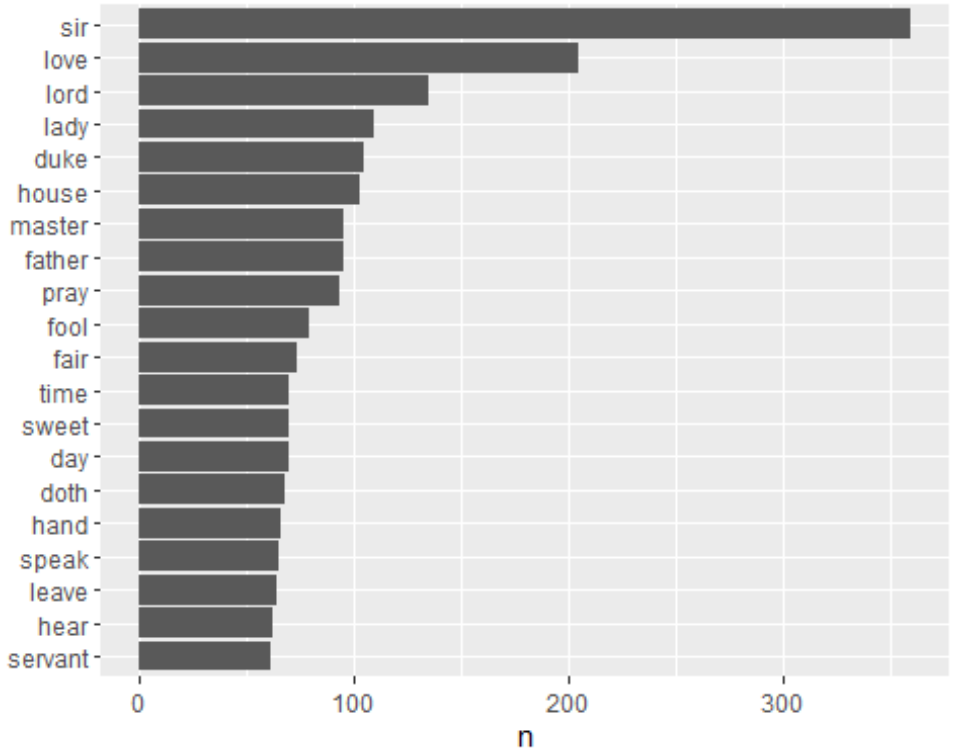
Duraklama kelimelerini veri setinden çıkardığımıza göre kelimelerin frekanslarını gösterelim.

```
tidy_shakespeare %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 6,016 x 2
##    word       n
##    <chr>  <int>
##  1 sir      360
##  2 love     205
##  3 lord     135
##  4 lady     109
##  5 duke     105
##  6 house    103
##  7 father    95
##  8 master    95
##  9 pray      93
## 10 fool      79
## # ... with 6,006 more rows
```

```
library(ggplot2)

tidy_shakespeare %>%
  count(word, sort = TRUE) %>%
```

```
  filter(n > 60) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



Şimdi de **Ben Jonson'dan** 3 adet komedi turundeki eseri ele alalim.

```
library(gutenbergr)

benjonson <- gutenberg_download(c(4011, 49461, 4039))
```

Bu 3 esere ait duraklama kelimelerini çıkaralım.

```
tidy_benjonson <- benjonson %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords)

## Joining, by = "word"
## Joining, by = "word"
```

Daha sonra kelime frekanslarına bakalım.

```
tidy_benjonson %>%
  count(word, sort = TRUE)

## # A tibble: 12,431 x 2
##    word          n
##    <chr>     <int>
##  1 sir        1087
##  2 true        400
```

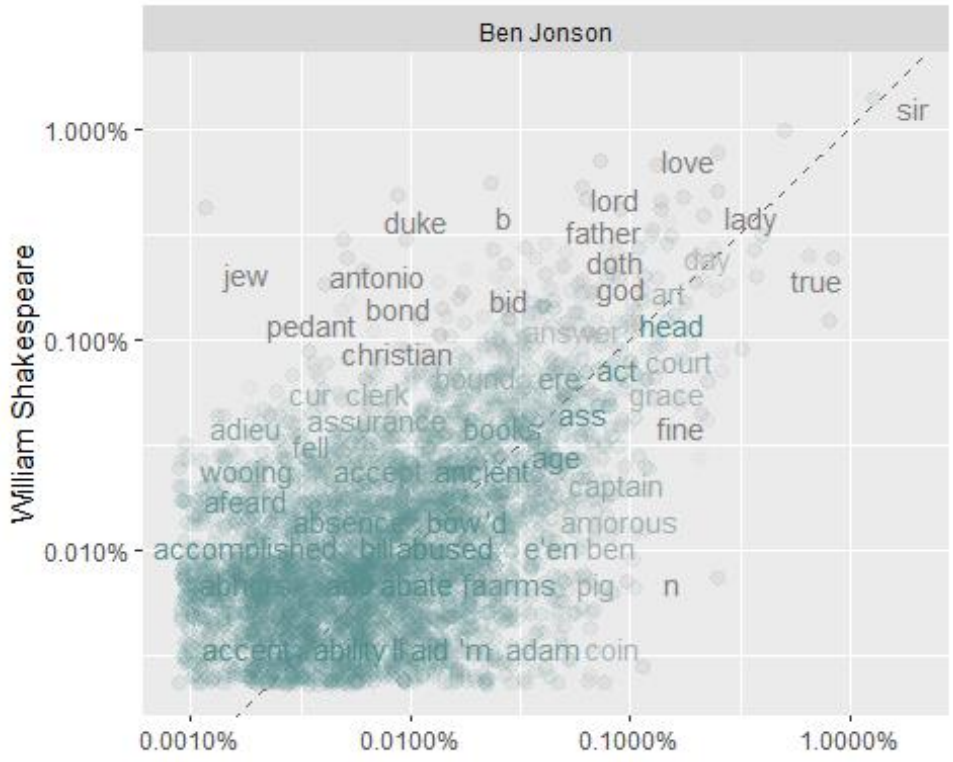
```
##  3 master      221
##  4 lady        198
##  5 time        180
##  6 fair        171
##  7 pray        142
##  8 cut         136
##  9 day         127
## 10 gentlemen   123
## # ... with 12,421 more rows
```

Şimdi 2 yazarın birbirleri ile olan benzerliklerine bakalım. Benzerliklere bakarken öncelikle burada 2 veri setini birleştirdik. Daha sonra proportion yani oran diye yeni bir değişken tanımladık. Bu oran kelimelerin toplam kelimelere bölünmesi şeklinde oluşmaktadır. Fakat burada toplam kelime dediğimiz şey her bir yazarın kullandığı kelimelerin toplamıdır.

Aşağıda William Shakespeare'in Ben Jonson'a göre oransal karılaştırmalarını yaptık.

```
library(tidyr)

## Warning: package 'tidyr' was built under R version 4.0.5

frequency <- bind_rows(mutate(tidy_shakespeare, author = "William Shakespeare"),
                       mutate(tidy_benjonson, author = "Ben Jonson")) %>%
  mutate(word = str_extract(word, "[a-z']+")) %>%
  count(author, word) %>%
  group_by(author) %>%
  mutate(proportion = n / sum(n)) %>%
  select(-n) %>%
  spread(author, proportion) %>%
  gather(author, proportion, `Ben Jonson`)

frequency

## # A tibble: 14,231 x 4
##    word      `William Shakespeare` author     proportion
##    <chr>                     <dbl> <chr>           <dbl>
##  1 'd                    0.000183  Ben Jonson  0.000250
##  2 'faith               NA         Ben Jonson  0.000107
##  3 'm                    0.0000456 Ben Jonson  0.000196
##  4 've                  NA         Ben Jonson  0.0000714
##  5 a                    NA         Ben Jonson  0.000535
##  6 a'leven               0.0000456 Ben Jonson NA
##  7 abandon              NA         Ben Jonson  0.0000178
##  8 abandon'd             0.0000913 Ben Jonson NA
##  9 abate                 0.0000913 Ben Jonson  0.000107
## 10 abatement             0.0000456 Ben Jonson NA
## # ... with 14,221 more rows

library(scales)

# expect a warning about rows with missing values being removed
ggplot(frequency, aes(x = proportion, y = `William Shakespeare`,
                      color = abs(`William Shakespeare` - proportion))) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
  geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format()) +
  scale_y_log10(labels = percent_format()) +
  scale_color_gradient(limits = c(0, 0.001),
                       low = "darkslategray4", high = "gray75") +
  facet_wrap(~author, ncol = 2) +
```

```
  theme(legend.position="none") +
  labs(y = "William Shakespeare", x = NULL)

## Warning: Removed 10766 rows containing missing values (geom_point).

## Warning: Removed 10767 rows containing missing values (geom_text).
```



Grafiğe baktığımızda hem Shakespeare hem de Ben Jonson'in eserlerinde ("lady", "fair", "day", "art", "beauty", "bride") kelimeleri benzer frekanslara sahiptir. Çünkü bu kelimeler kesikli çizgiye yakın olan kelimelerdir. Kelimelerin çizgiye olan yakınlığı benzer frekanslara sahip olduklarının göstergesidir. "daughter", "father", "love" gibi kelimeler Shakespeare'in eserlerinde daha sık bulunur, Ben Jonson eserlerinde bu kelimelere çok fazla değinmemiştir. Çünkü bu kelimeler çizgiden uzakta ve grafiğin üstünde yer almaktadır. "busy", "fine", "coin" gibi kelimeleri ise Ben Jonson eserlerinde daha sık kullanmıştır. Bu grafik sayesinde yazarların eserlerindeki ortak ve farklı kelimeleri görebilmekteyiz. Fakat genel olarak baktığımızda Ben Jonson'ın Shakespeare ile ortak olmayan kelime sayısı daha azdır. Bu da eserlerini William Shakespeare ile benzer yazdığı kanısına bir işarettir. Çünkü Shakespeare'in eserlerindeki özgün kelime sayısı çok daha fazladır.

Şimdi kelime frekans kümesinin birbirine benzerliklerine korelasyon ile bakalım. William Shakespeare ile Ben Jonson'un eserlerindeki kelime frekansları arasındaki ilişkinin nasıl olduğunu inceleyelim.

```
cor.test(data = frequency[frequency$author == "Ben Jonson",],
         ~ proportion + `William Shakespeare`)

##
##  Pearson's product-moment correlation
##
## data:  proportion and William Shakespeare
## t = 60.757, df = 3463, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
```

```
##  0.7017904 0.7340444
## sample estimates:
##       cor
## 0.7183031
```

Orana göre Pearson Korelasyon katsayısının sonucuna baktığımızda p değeri<0.05 oldugundan H0: 2 değişken arasında ilişki yoktur hipotezi reddedilir. Burada H0 hipotezini reddetiğimize gore William Shakespeare ile Ben Jonson arasında bir ilişki olduğu söylenir. Korelasyon katsayısına baktığımızda 0.71 olarak bulunduğu görüyoruz ve bu değer anlamlı bir ilişki olduğunun göstergesidir.

DETAYLI DUYGU ANALIZI

```
nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

tidy_shakespeare %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE)

## Joining, by = "word"

## # A tibble: 228 x 2
##    word         n
##    <chr>    <int>
##  1 love       205
##  2 pray        93
##  3 sweet       70
##  4 daughter    57
##  5 true        55
##  6 god         51
##  7 art         49
##  8 faith       48
##  9 friend      48
## 10 marry       43
## # ... with 218 more rows
```

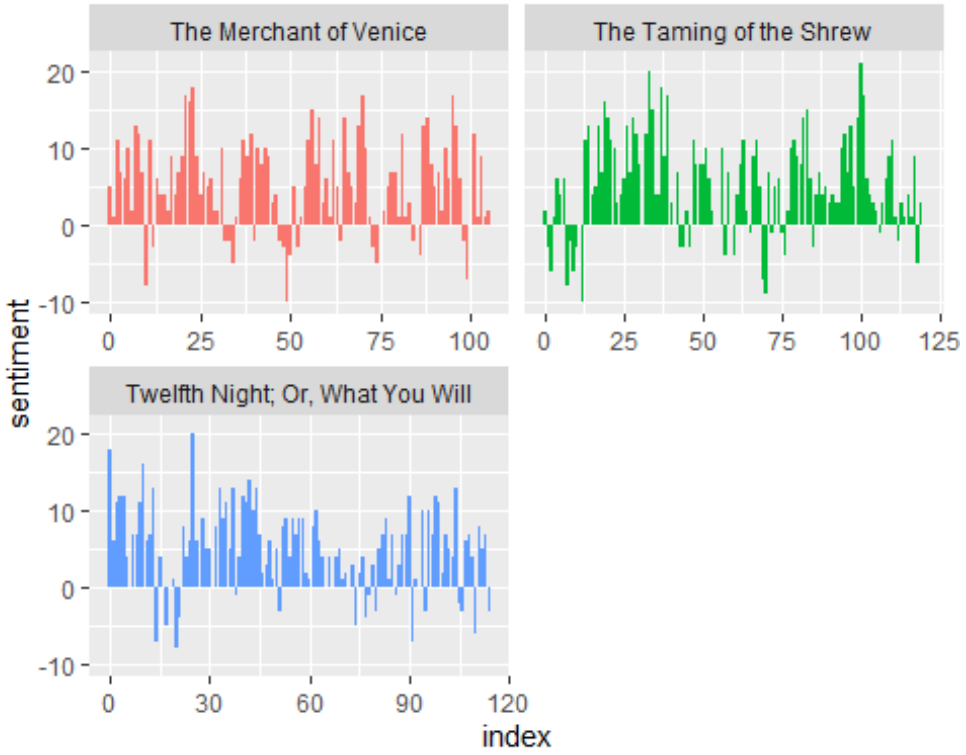
Burada Shakespeare'in eserlerini seçtik daha sonra nrc sözlüğündeki joy duygusunu seçip count olarak kelimeleri saydırdık.

NOT: %/% bu operator tam sayı bölmeye yarıyor. Biz elimizdeki metni 40 satırlık parçalara ayırmak istiyoruz. Daha sonra bunun duygu analizini yapmak istiyoruz her bir kitap için.

```
library(tidyr)

shakespeare_sentiment <- tidy_shakespeare %>%
  inner_join(get_sentiments("nrc")) %>%
  count(title, index = linenumber %/% 40, sentiment) %>%
  spread(sentiment, n , fill = 0) %>%
  mutate(sentiment = positive - negative)

## Joining, by = "word"
```

Şimdi ise daha detaylı bir duygu analizi yapmak için eserlerimiz ile nrc duygu sözlüğünü birleştirdik. Yani bizim elimizdeki kitaplar olumlu mu olumsuz mu bunları karşılaştırmak istiyoruz. Bunu yapmak için öncelikle eserlerimizi 40 satırlık tam sayıya böldük. Daha sonra nrc sözlüğü içerisindeki pozitif ve negatif olarak tanımlı duyguları eserlerimizdeki kelime frekanslarına göre bir değişken olarak tanımladık daha sonra pozitif ve negatif frekanslar arasındaki farkı alıp görselleştirdik.

```
library(ggplot2)

ggplot(shakespeare_sentiment, aes(index, sentiment, fill = title)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~title, ncol = 2, scales = "free_x")
```



Yukarıdaki grafiklere bakıldığında Shakespeare'e ait eserlerin olay örgüsünün, hikayenin gidişatı üzerinde daha olumlu veya olumsuz duygulara doğru nasıl değiştiğini görebiliriz. Genel olarak 3 eserde de olumlu ve olumsuz bölümler olduğu görülmektedir. Genel olarak Shakespeare'in eserlerinin olumlu duygular içerdiğini görüyoruz. Yazara ait eserlerin tümünde ani duygu değişimleri görülmektedir.

Simdi nrc sözlüğüne bir göz atalim. Pozitif ve negatif oldugu durumda bunun duygularinin frekanslarina bakalim.

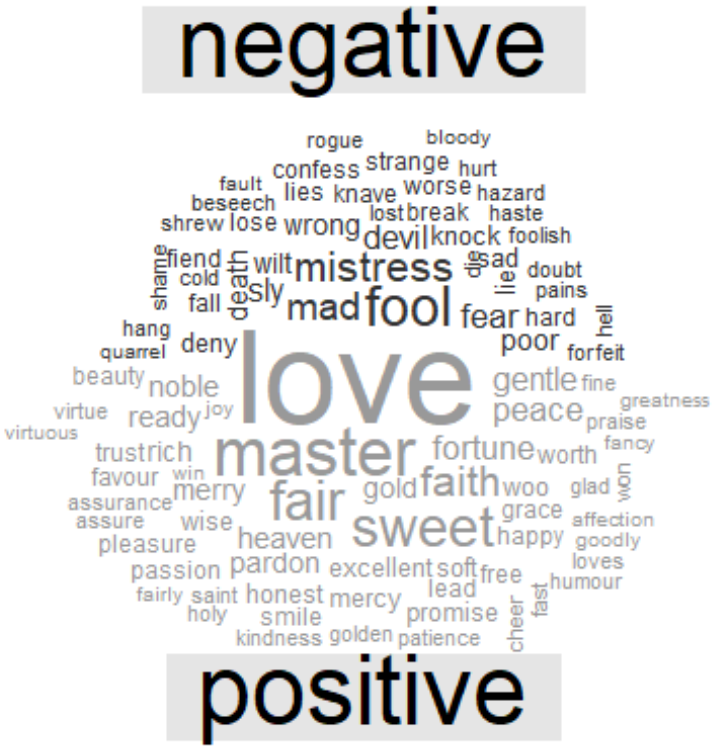
```
get_sentiments("nrc") %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  count(sentiment)

## # A tibble: 2 x 2
##   sentiment     n
```

```
##   <chr>     <int>
## 1 negative   3324
## 2 positive   2312
```

Negatif duygular daha fazladır. Negatif kelimelerin daha fazla olduğu bir durumda bir metnin olumlu çıkmasi gayet iyi bir durumdur.

KELIME BULUTU

```
library(reshape2)

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths

library(wordcloud)

## Warning: package 'wordcloud' was built under R version 4.0.5

## Loading required package: RColorBrewer

tidy_shakespeare %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray60"),
                   max.words = 100)

## Joining, by = "word"
```



Kelime bulutu bir sekil etrafında kelimeleri frekanslarına göre dağıtıyor. Burada yazının boyutunun büyümesi onların eserlerde daha sık kullanılan kelimeler olduğunu ifade etmektedir. Kelimelerin tonları saydamlaştıkça anlamlarının pozitifleştiğini söyleyebiliriz.

TF-IDF ISTATISTIGI (KELIME VE BELGE SIKLIGI ANALIZI)

Burada Shakespeare'in kitaplarını çağırdık ve unnest tokens ile metinleri kelimelere parçaladik. count işlemi ile de kitapların ve kelimelerin frekanslarına baktık. Burada group by ile her bir kitabı gruplayıp daha sonra her bir kitap içerisindeki kelimelerin frekanslarına bakıyor olacağız. left join ile de kelime frekanslarını ve toplam frekanslarını birleştirmiş olduk.

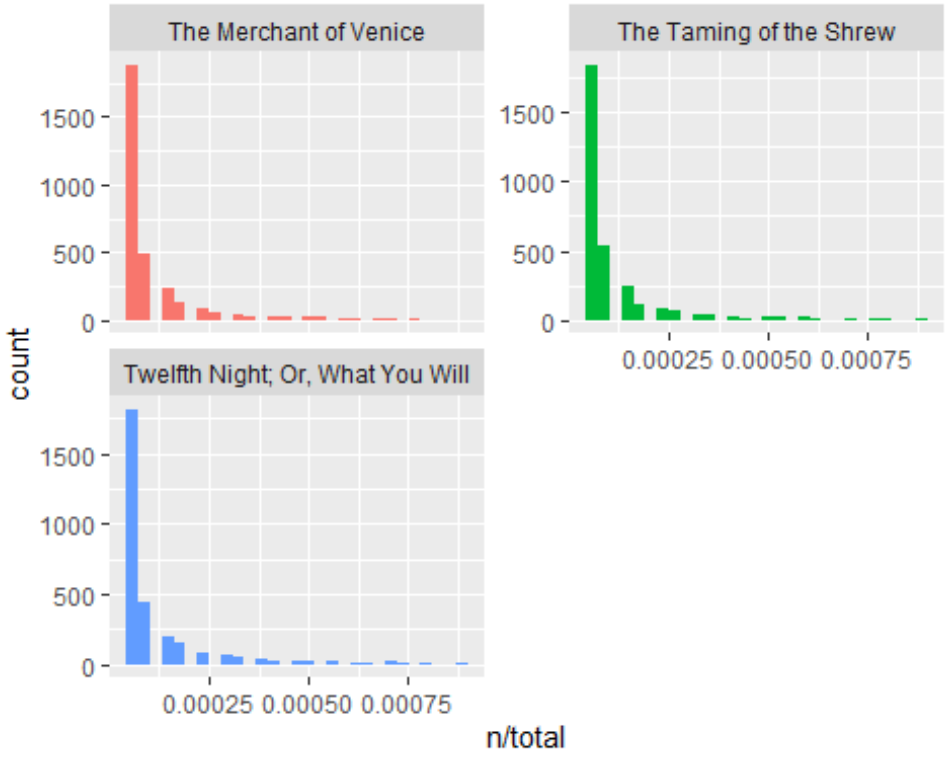
```
library(dplyr)
library(tidytext)

book_words <- shakespeare %>%
  unnest_tokens(word, text) %>%
  count(title, word, sort = TRUE)

total_words <- book_words %>%
  group_by(title) %>%
  summarize(total = sum(n))

book_words <- left_join(book_words, total_words)

## Joining, by = "title"

book_words

## # A tibble: 9,810 x 4
##    title                             word      n total
##    <chr>                             <chr> <int> <int>
##  1 The Merchant of Venice            the     829 22328
##  2 The Taming of the Shrew           and     776 22302
##  3 The Merchant of Venice            i       653 22328
##  4 The Taming of the Shrew           i       630 22302
##  5 The Merchant of Venice            and     614 22328
##  6 Twelfth Night; Or, What You Will  i       600 21386
##  7 Twelfth Night; Or, What You Will  the     565 21386
##  8 The Taming of the Shrew           to      536 22302
##  9 Twelfth Night; Or, What You Will  and     527 21386
## 10 The Taming of the Shrew           the     516 22302
## # ... with 9,800 more rows
```

Burada örnek vermek gerekirse The Merchant of Venice kitabı içerisinde the kelimesi 829 kere geçmis ve toplam da The Merchant of Venice kitabında 22328 kelime geçmiştir.

Her bir kitap için bir kullanım sıklığı var ve farklı davranışlar sergiliyor mu sergilemiyor mu bunu araştırmak istiyoruz. Bunun için her bir kitap için kelimelerin dağılımını gösteren bir grafik çizdirelim.

```
library(ggplot2)

ggplot(book_words, aes(n/total, fill = title)) +
  geom_histogram(show.legend = FALSE) +
  xlim(NA, 0.0009) +
  facet_wrap(~title, ncol = 2, scales = "free_y")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 503 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 3 rows containing missing values (geom_bar).
```



Görüldüğü üzere 3 kitapta da benzer kelime frekansı kullanılmış. Yani Shakespeare'in 3 kitabında da belirli benzer kelimeleri kullandığını söyleyebiliriz.

ZIPF KANUNU

Zipf kanununu bir şey artarken bir şeyin azalması şeklinde düşünebiliriz. Aynı zamanda dünyadaki birçok olay Zipf Yasasına uygun hareket eder.

book_words ile kelime frekanslarını aldık ve group by ile her bir kitaba göre gruplama yaptık. Daha sonra mutate işlemi ile yeni değişkenler oluşturduk. Rank dediğimiz satır sayılarını veren bir değisken oluşturduk ve kelime frekansı dediğimizde de kelime frekansı/o kitap içerisinde geçen toplam frekans diye bir değisken oluşturduk ve bunu da freq_by_rank olarak isimlendirdik.

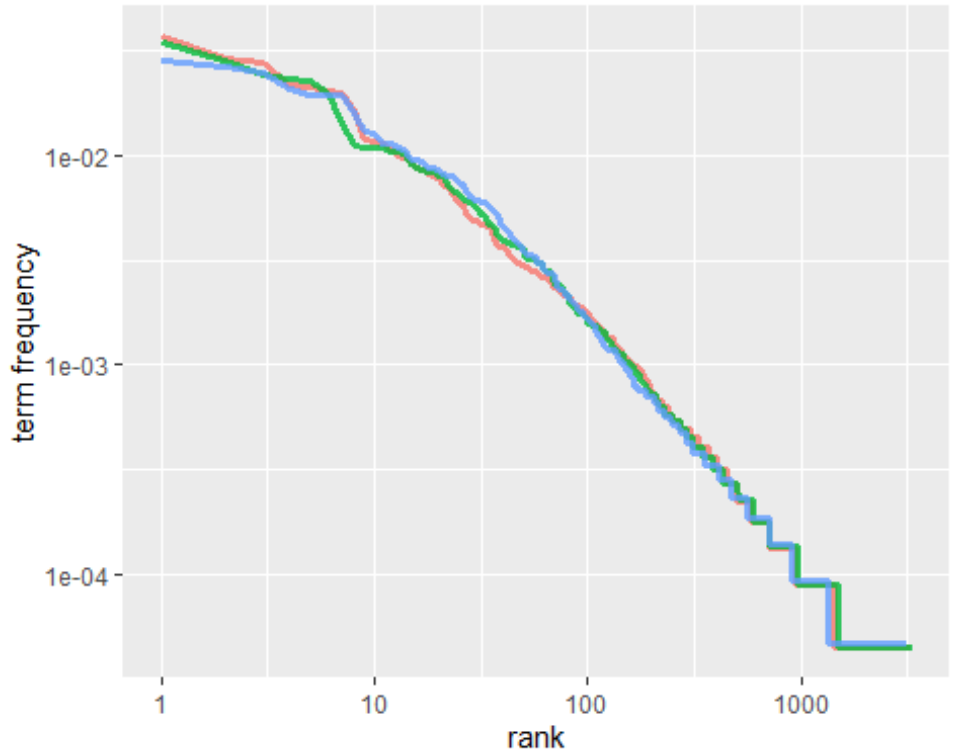
```
freq_by_rank <- book_words %>%
  group_by(title) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total) %>%
  ungroup()

freq_by_rank
```

```
## # A tibble: 9,810 x 6
##    title                          word      n total  rank `term frequency`
##    <chr>                          <chr> <int> <int> <int>            <dbl>
##  1 The Merchant of Venice         the     829 22328     1           0.0371
##  2 The Taming of the Shrew        and     776 22302     1           0.0348
##  3 The Merchant of Venice         i       653 22328     2           0.0292
##  4 The Taming of the Shrew        i       630 22302     2           0.0282
##  5 The Merchant of Venice         and     614 22328     3           0.0275
##  6 Twelfth Night; Or, What You Will i       600 21386     1           0.0281
##  7 Twelfth Night; Or, What You Will the     565 21386     2           0.0264
```

```
##  8 The Taming of the Shrew          to      536 22302     3           0.0240
##  9 Twelfth Night; Or, What You Will and     527 21386     3           0.0246
## 10 The Taming of the Shrew          the     516 22302     4           0.0231
## # ... with 9,800 more rows
```

Görmüs oldugumuz üzere 829/22328=0.0371 seklinde bir deger verdi. Yani The Merchant of Venice kitabinda the kelimesinin gecme orani 0.0371'dir diyor kelime frekansi.

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = title)) +
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```



Zipf yasasi bir seyin artarken bir seyin azalmasini gösteriyordu bu grafik de bunu gösteriyor. Görmüs oldugumuz üzere rank arttikca kelime frekansi azaliyor.

Şimdi rankların alt kümesini alalım.

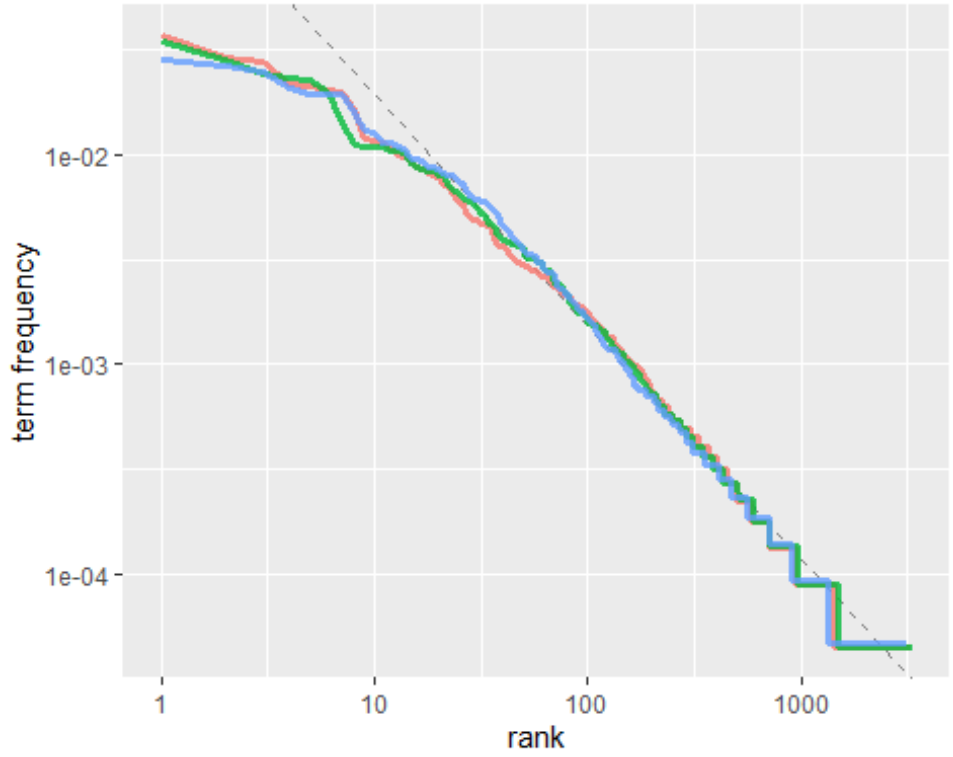
```
rank_subset <- freq_by_rank %>%
  filter(rank < 500,
         rank > 10)

lm(log10(`term frequency`) ~ log10(rank), data = rank_subset)

##
## Call:
## lm(formula = log10(`term frequency`) ~ log10(rank), data = rank_subset)
##
## Coefficients:
## (Intercept)  log10(rank)
##     -0.5978      -1.1080
```

Burada katsayıları bulmak üzere bir regresyon modeli oluşturduk. Regresyon modelinde de kelime frekansının logaritmik hali bağımlı değişken ve rank ise bağımsız değisken olarak aldık.

Bu katsayılar ne işimize yarayacak?

Yukarıda kurmuş olduğumuz regresyon modeli ile aşağıdaki kesikli bir biçimde gösterilen eğimi bulmuş olduk.

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = title)) +
  geom_abline(intercept = -0.5978, slope = -1.1080,
              color = "gray50", linetype = 2) +
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```



bind_tf_idf() FONKSİYONU

Bu fonksiyon her bir kelimenin kitabın ve frekansa göre bir bağlama işlemi yapar.

```
book_tf_idf <- book_words %>%
  bind_tf_idf(word, title, n)

book_tf_idf
```

```
## # A tibble: 9,810 x 7
##    title                           word      n total     tf   idf tf_idf
##    <chr>                           <chr> <int> <int>  <dbl> <dbl>  <dbl>
##  1 The Merchant of Venice          the     829 22328 0.0371     0      0
##  2 The Taming of the Shrew         and     776 22302 0.0348     0      0
##  3 The Merchant of Venice          i       653 22328 0.0292     0      0
##  4 The Taming of the Shrew         i       630 22302 0.0282     0      0
##  5 The Merchant of Venice          and     614 22328 0.0275     0      0
##  6 Twelfth Night; Or, What You Will i       600 21386 0.0281     0      0
##  7 Twelfth Night; Or, What You Will the     565 21386 0.0264     0      0
```

```
##  8 The Taming of the Shrew          to      536 22302 0.0240     0      0
##  9 Twelfth Night; Or, What You Will and     527 21386 0.0246     0      0
## 10 The Taming of the Shrew          the     516 22302 0.0231     0      0
## # ... with 9,800 more rows
```

```
book_tf_idf %>%
  select(-total) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 9,810 x 6
##    title                          word          n      tf   idf  tf_idf
##    <chr>                          <chr>     <int>   <dbl> <dbl>   <dbl>
##  1 Twelfth Night; Or, What You Will sır         274 0.0128   1.10 0.0141
##  2 Twelfth Night; Or, What You Will toby        206 0.00963  1.10 0.0106
##  3 The Taming of the Shrew          petruchıo   176 0.00789  1.10 0.00867
##  4 Twelfth Night; Or, What You Will vıola       134 0.00627  1.10 0.00688
##  5 Twelfth Night; Or, What You Will olıvıa      129 0.00603  1.10 0.00663
##  6 The Merchant of Venice           portıa      129 0.00578  1.10 0.00635
##  7 The Taming of the Shrew          tranıo      107 0.00480  1.10 0.00527
##  8 The Merchant of Venice           shylock     106 0.00475  1.10 0.00522
##  9 The Taming of the Shrew          katherına   103 0.00462  1.10 0.00507
## 10 Twelfth Night; Or, What You Will malvolıo     97 0.00454  1.10 0.00498
## # ... with 9,800 more rows
```

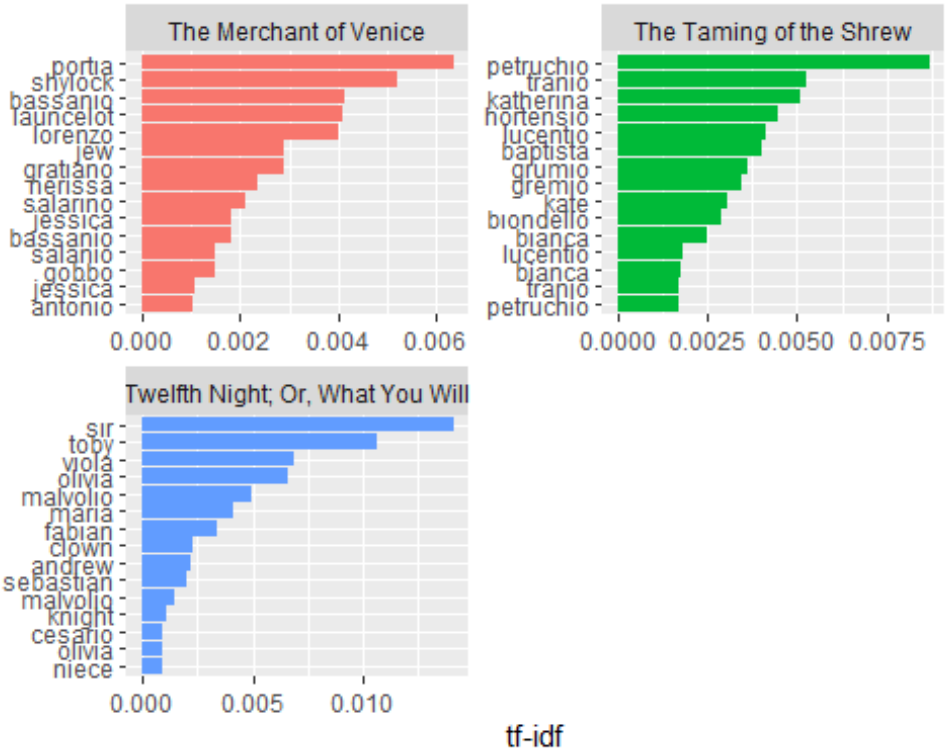
Burada tf_idf değerine göre azalan şekilde göstermesi için ayarladık. Burada tf_idf değerinin en yüksek oldugu kitap Twelfth Night; Or, What You Will kitabıymış ve en yüksek çıkan oran da **sir** ardından da **Toby** kelimesiymiş. 2. en yüksek tf_idf değeri ise The Taming of the Shrew kitabıymış ve en yüksek çıkan oran da **Petruchio** kelimesi olmuş. Bu 2 kelime de bir karakter ismidir.

Söyle düsünmemiz lazim: Bir kitap kisilere olaylara mekanlara dayandigindan kaynakli belirli seyler etrafinda dönüp dolasaktir ve en fazla bunlarin cikmasi beklenir. Burada da zaten her bir kitapta bas kahramanlar ya da önemli olan yan karakterler gözükmüs oluyor.

```
library(forcats)
```

```
## Warning: package 'forcats' was built under R version 4.0.5
```

```
book_tf_idf %>%
  group_by(title) %>%
  slice_max(tf_idf, n = 15) %>%
  ungroup() %>%
  ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = title)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~title, ncol = 2, scales = "free") +
  labs(x = "tf-idf", y = NULL)
```



Grafiğe baktığımızda tf_idf değerlerine göre The Taming of the Shrew eserinde petruchio, tranio,katherina,hortensio gibi kelimeler olduğunu görüyoruz. Çoğunluk olarak isimler mevcuttur.

NOT Kelime frekansı metin içerisinde geçen en fazla tekrar eden kelimeleri gösteriyor bize ama metin içerisinde arka planda gibi gözüken kelimeleri ön plana çıkarmaz. Burada da tf-idf istatistiğine ihtiyaç var. Yani en çok tekrar eden kelimeler değil de metin içerisinde arka planda kalan ana fikri anlatmaya calışan kelimeleri bize gösteriyor. O yüzden tf-idf istatistiği tf'e göre daha fazla tercih edilir.

Ben Jonson

Gutenberg'den Metin Verisi Cekmek

```
library(gutenbergr)

## Warning: package 'gutenbergr' was built under R version 4.0.4

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

benjonson_eserleri<- gutenberg_metadata %>%
  filter(author %in% "Jonson, Ben",
         language == "en")

benjonson<- gutenberg_download(c(4011,49461,4039))

## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest

## Using mirror http://aleph.gutenberg.org
```

Burada filtreleme işlemi ile gutenberg kütüphanesinde Ben Jonson'a ait olan 3 adet komedi türündeki eserlerini çektik.

```
basliklar<-benjonson_eserleri %>%
  select(gutenberg_id, title)

benjonson <- benjonson %>%
  inner_join(basliklar, by = "gutenberg_id")
```

Yukarıda gutenberg id numarasına göre çektiğimiz eserlere başlık atadık. Bu sayede kitapları ayırt edebiliriz.

PARCALAMA ISLEMLERI

```
library(tidytext)

## Warning: package 'tidytext' was built under R version 4.0.5

benjonson1<-benjonson %>%
  unnest_tokens(word, text) %>%
  count(word, sort = TRUE)
benjonson1

## # A tibble: 13,040 x 2
##    word      n
##    <chr> <int>
##  1 the    4909
##  2 and    3868
##  3 of     3572
##  4 a      3103
##  5 to     2919
##  6 you    2522
##  7 ı      2369
##  8 in     2003
##  9 that   1253
## 10 is     1192
## # ... with 13,030 more rows
```

Burada unnest_tokens fonksiyonu ile metindeki her bir kelimeyi parçaladık. Kelime frekanslarına bakmak için count fonksiyonunu kullandık. Burada her bir kelimenin metin içerisinde kaç kez geçtiğini görüyoruz. Burada gördüğümüz gibi en fazla tekrar eden kelimeler anlamsız görünüyor. Burada baktığımızda 82. gözlemde **Cler** karakterinin geldiğini görüyoruz. Yani ilk 81 gözlem bizim için bir anlam ifade etmiyor. Bu durumda metinde çok fazla tekrar eden ama tek başına bir anlam ifade etmeyen kelimelere **duraklama kelimeleri** denir. Duraklama kelimelerini metin içerisinden çıkarmamız gerek.

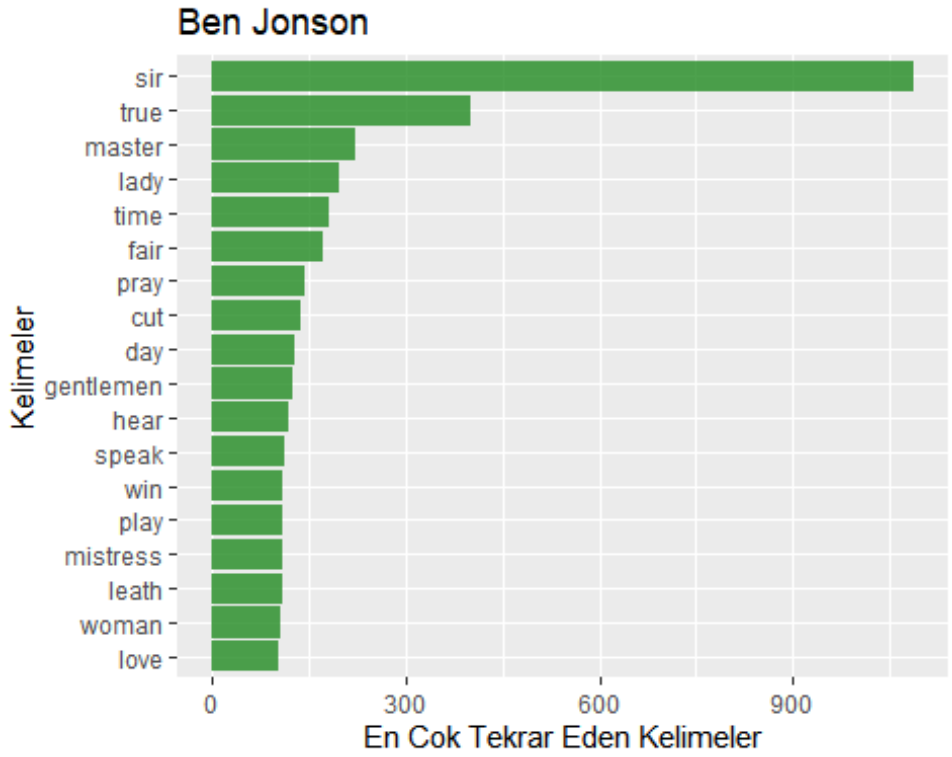
```
tidybenjonson<-benjonson %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE)

## Joining, by = "word"

tidybenjonson

## # A tibble: 12,479 x 2
##    word       n
##    <chr>  <int>
##  1 ı       2369
##  2 sir     1087
##  3 true     400
##  4 mos      327
##  5 thou     285
##  6 jonson   278
##  7 ı'll     240
##  8 cler     225
##  9 volp     224
## 10 master   221
## # ... with 12,469 more rows
```

Suanda tek basina anlamsiz gorunen kelimeler versetimizden cikartildi ve verimiz tidy hale geldi. Verimiz düzenli metin formatına dönüşmüş oldu. Metin eski dilde yazıldığından ötürü stopwords olarak algılanmayan bazı kelimeler de mevcut. **Örnegin** thy:sizin gibi. Bunları da temizlememiz gerekli. Ayrıca amacımız yazar benzerliklerini farketmek olduğundan veri setini karakter isimlerinden de ayrıştırdık.

```
mystopwords<-tibble(word = c("ı", "thou", "ı'll", "jonson", "ın", "la", "thy", "thee", "tis
", "ıt", "ay", "avoc", "sır", "ıf", "ıs", "mine", "jonson's", "f", "1", "_re", "uh", "ha",
"_ıt", "ly", "2", "3", "shakespeare", "ı", "thou", "sır", "thy", "thee", "ı'll", "ıf", "hat
h", "tis", "ın", "ay", "mine", "ıt", "ıs", "la", "toby", "petruchıo", "enter", "vıola", "ol
ıvıa", "portıa", "clown","andrew", "tranıo", "shylock", "katherına", "malvolıo", "antonıo",
"exit", "hortensıo", "bassanıo", "lucentıo", "launcelot", "baptısta", "lorenzo", "marıa", "
grumıo", "exeunt", "gremıo", "fabıan", "kate", "mos","jonson", "cler",  "volp", "cokes", "d
aw", "mor", "daup", "nay","quar", "waspe", "lıt", "avoc", "ott", "corv", "corb", "mosca", "
john", "jonson's", "numps", "volt", "wınw"))
tidybenjonson1<-benjonson %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords) %>%
  count(word, sort = TRUE)

## Joining, by = "word"
## Joining, by = "word"

tidybenjonson1

## # A tibble: 12,425 x 2
##    word          n
```

```
##    <chr>     <int>
##  1 sir        1087
##  2 true        400
##  3 master      221
##  4 lady        198
##  5 time        180
##  6 fair        171
##  7 pray        142
##  8 cut         136
##  9 day         127
## 10 gentlemen   123
## # ... with 12,415 more rows
```

```
library(ggplot2)
tidybenjonson1 %>%
  filter(n > 100) %>%
  ggplot(aes(reorder(word,n),n))+
  geom_col(fill = "forestgreen", alpha = 0.8)+
  coord_flip()+
  labs(x = "Kelimeler",
       y = "En Cok Tekrar Eden Kelimeler",
       title = "Ben Jonson")
```



Baktığımız zaman burada da asalet ünvanlarının sıkça kullanıldığını görüyoruz. Ben Jonson için de eserlerinde aşk temasını işlediğini ve eserlerinde dine yer verdiğini söylemek mümkün olsa da Ben Jonson aşk temasını eserlerinde daha az dile getirmiştir. Komedi türündeki bu eserlerde “win, play, fellow, court” gibi kelimelerin sıklıkla geçtiği görülmektedir. Ben Jonson’ın eserlerindeki gülünç durum dost olan karakterler arasında yaşanan kazanan ve kaybedenin olduğu oyunlar sonrasında olanlardan kaynaklanıyor olabilir. Bu oyunların oynanış sebebi Shakespeare’de olduğu gibi bir aşk hikayesine bağlanıyor olabilir. Ve eserlerinde bir haksızlık söz konusunu olması muhtemel ki mahkeme kelimesine sıklıkla başvurulmuştur. Söz konusu olan haksızlık her neyse bu durum da eserlerdeki gülünçlüğe sebep olmuş olabilir.

Dikkat edilecek olursa Shakespeare “baba” kelimesine sıklıkla başvururken Ben Jonson’da bu durum gözlenmemiştir.

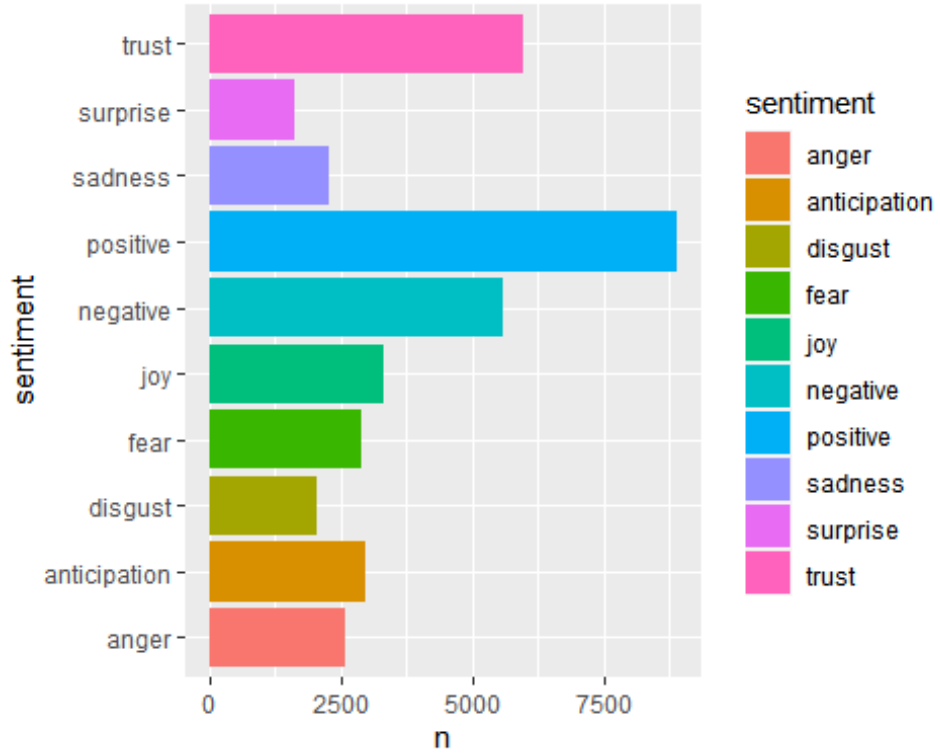
DUYGU ANALIZI

```
tidybenjonson1 %>%
  inner_join(get_sentiments("nrc"))

## Joining, by = "word"

## # A tibble: 5,072 x 3
##    word       n sentiment
##    <chr>  <int> <chr>
##  1 sir     1087 positive
##  2 sir     1087 trust
##  3 true     400 joy
##  4 true     400 positive
##  5 true     400 trust
##  6 master   221 positive
##  7 time     180 anticipation
##  8 fair     171 positive
##  9 pray     142 anticipation
## 10 pray     142 fear
## # ... with 5,062 more rows

tidybenjonson1 %>%
  inner_join(get_sentiments("afinn"))

## Joining, by = "word"

## # A tibble: 860 x 3
##    word       n value
##    <chr>  <int> <dbl>
##  1 true     400     2
##  2 fair     171     2
##  3 pray     142     1
##  4 cut      136    -1
##  5 win      110     4
##  6 love     103     3
##  7 fine      96     2
##  8 leave     96    -1
##  9 humour    93     2
## 10 faith     85     1
## # ... with 850 more rows

tidybenjonson1 %>%
  inner_join(get_sentiments("bing"))

## Joining, by = "word"

## # A tibble: 1,650 x 3
##    word         n sentiment
##    <chr>    <int> <chr>
##  1 master     221 positive
##  2 fair       171 positive
##  3 mistress   110 negative
##  4 win        110 positive
##  5 love       103 positive
##  6 fine        96 positive
##  7 humour      93 positive
##  8 knock       90 negative
```

```
##  9 faith       85 positive
## 10 grace       83 positive
## # ... with 1,640 more rows
```

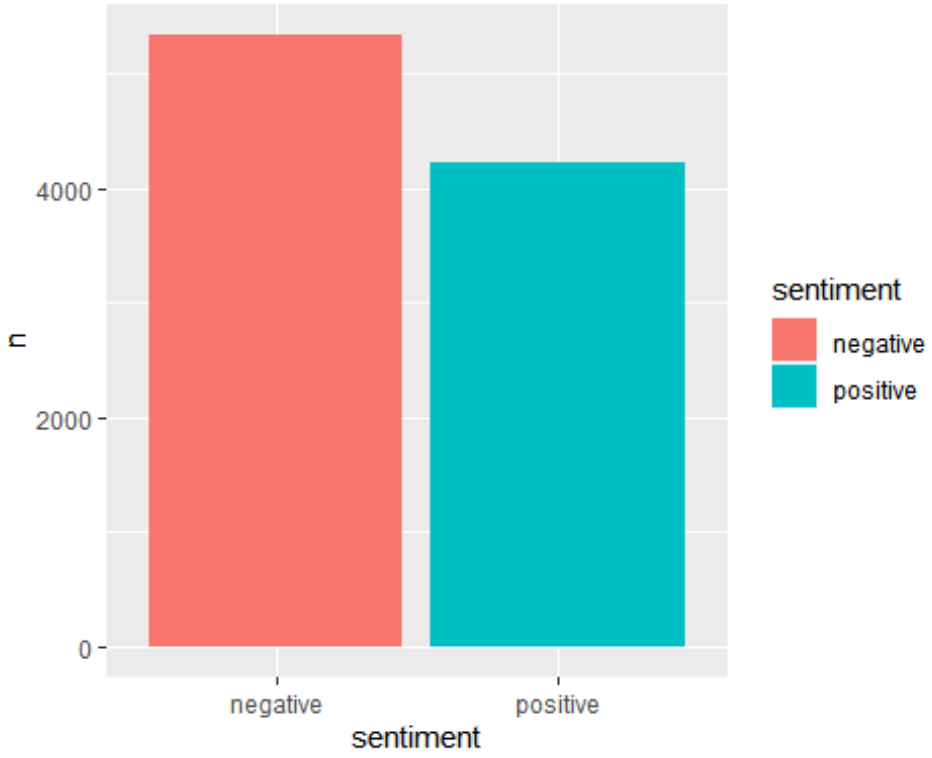
NRC sözlüğüne göre duyguların frekanslarına dayalı bir görselleştirme yapalım.

```
tidybenjonson1 %>%
  inner_join(get_sentiments("nrc"))%>%
  ggplot(aes(sentiment, n, fill = sentiment))+
  coord_flip()+
  geom_col()

## Joining, by = "word"
```



Bu grafiğe göre Ben Jonson'un komedi türündeki eserlerinin pozitif duygular içerdiğini söyleyebiliriz. NRC sözlüğünde pozitiften çok negatif duygular daha ağır basar.

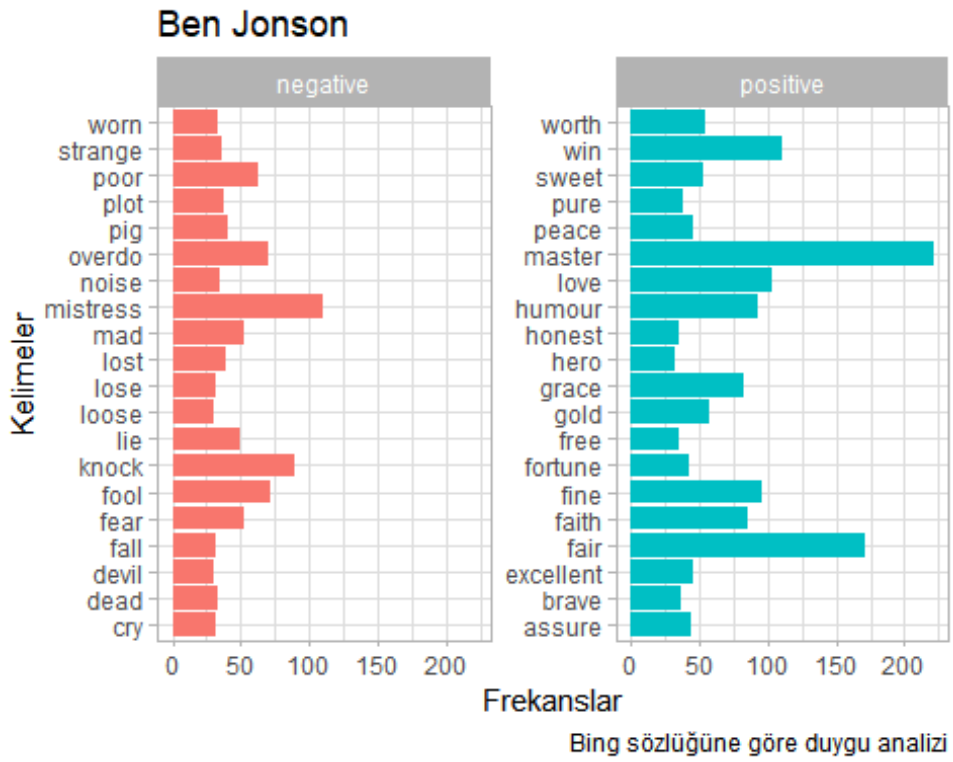
```
tidybenjonson1 %>%
  inner_join(get_sentiments("bing"))%>%
  ggplot(aes(sentiment, n, fill = sentiment))+
  geom_col()

## Joining, by = "word"
```



Bing sözlüğüne göre Ben Jonson'da da negatif duygular daha ağır basmaktadır.Ben Jonson da komedi türündeki eserlerinde negatif kelimelere daha fazla yer vermektedir.

Hangi kelimeler daha fazla ön plana çıkmış?

```
rbind(

tidybenjonson1 %>%
  inner_join(get_sentiments("bing"))%>%
  arrange(-n) %>%
  filter(sentiment == "positive") %>%
  head(20),

tidybenjonson1 %>%
  inner_join(get_sentiments("bing"))%>%
  arrange(-n) %>%
  filter(sentiment == "negative") %>%
  head(20)
) %>%
  ggplot(aes(word, n, fill = sentiment))+
  coord_flip()+
  facet_wrap(~sentiment, scales = "free_y")+
  geom_col(show.legend = "FALSE")+
  theme_light()+
  labs(x = "Kelimeler",
       y= "Frekanslar",
       title= "Ben Jonson",
       caption = "Bing sözlüğüne göre duygu analizi")

## Joining, by = "word"
## Joining, by = "word"

## Warning: `show.legend` must be a logical vector.
```



Negatif kelimeler arasında “yıpranmış, garip, zavallı, domuz, abartmak” gibi kelimelerin ön planda olduğu görülürken bu kelimelerden en ağır basan kelimenin “metres” olduğu görülmektedir.Pozitif kelimeler arasında “kazanmak, tatlı, saf, barış” gibi kelimelerin ön planda olduğu görülürken bu kelimelerden en ağır basan kelimenin “usta” olduğu görülmektedir.

```
library(stringr)

benjonson <- benjonson %>%
  group_by(title) %>%
  mutate(linenumber = row_number(),
         chapter = cumsum(str_detect(text,
                                     regex(c("^scene [\\divxlc]","^act [\\divxlc]"),
                                           ignore_case = TRUE)))) %>%
  ungroup()
benjonson
```

```
## # A tibble: 26,438 x 5
##    gutenberg_id text                    title                 linenumber chapter
##           <int> <chr>                   <chr>                      <int>   <int>
##  1         4011 "EPICOENE; OR, THE SIL~ Epicoene; Or, The Si~          1       0
##  2         4011 ""                      Epicoene; Or, The Si~          2       0
##  3         4011 ""                      Epicoene; Or, The Si~          3       0
##  4         4011 "By Ben Jonson"         Epicoene; Or, The Si~          4       0
##  5         4011 ""                      Epicoene; Or, The Si~          5       0
##  6         4011 ""                      Epicoene; Or, The Si~          6       0
##  7         4011 ""                      Epicoene; Or, The Si~          7       0
##  8         4011 ""                      Epicoene; Or, The Si~          8       0
##  9         4011 ""                      Epicoene; Or, The Si~          9       0
## 10         4011 "INTRODUCTION"          Epicoene; Or, The Si~         10       0
## # ... with 26,428 more rows
```

Stringr ile bir veri manipulasyonu işlemi yapıldı. Daha sonra her bir satıra bir satır numarası ekledik.

```
tidy_benjonson<- benjonson %>%
  unnest_tokens(word, text)
tidy_benjonson
```

```
## # A tibble: 140,597 x 5
##    gutenberg_id title                           linenumber chapter word
##           <int> <chr>                                <int>   <int> <chr>
##  1         4011 Epicoene; Or, The Silent Woman          1       0 epıcoene
##  2         4011 Epicoene; Or, The Silent Woman          1       0 or
##  3         4011 Epicoene; Or, The Silent Woman          1       0 the
##  4         4011 Epicoene; Or, The Silent Woman          1       0 sılent
##  5         4011 Epicoene; Or, The Silent Woman          1       0 woman
##  6         4011 Epicoene; Or, The Silent Woman          4       0 by
##  7         4011 Epicoene; Or, The Silent Woman          4       0 ben
##  8         4011 Epicoene; Or, The Silent Woman          4       0 jonson
##  9         4011 Epicoene; Or, The Silent Woman         10       0 ıntroductıon
## 10         4011 Epicoene; Or, The Silent Woman         12       0 the
## # ... with 140,587 more rows
```

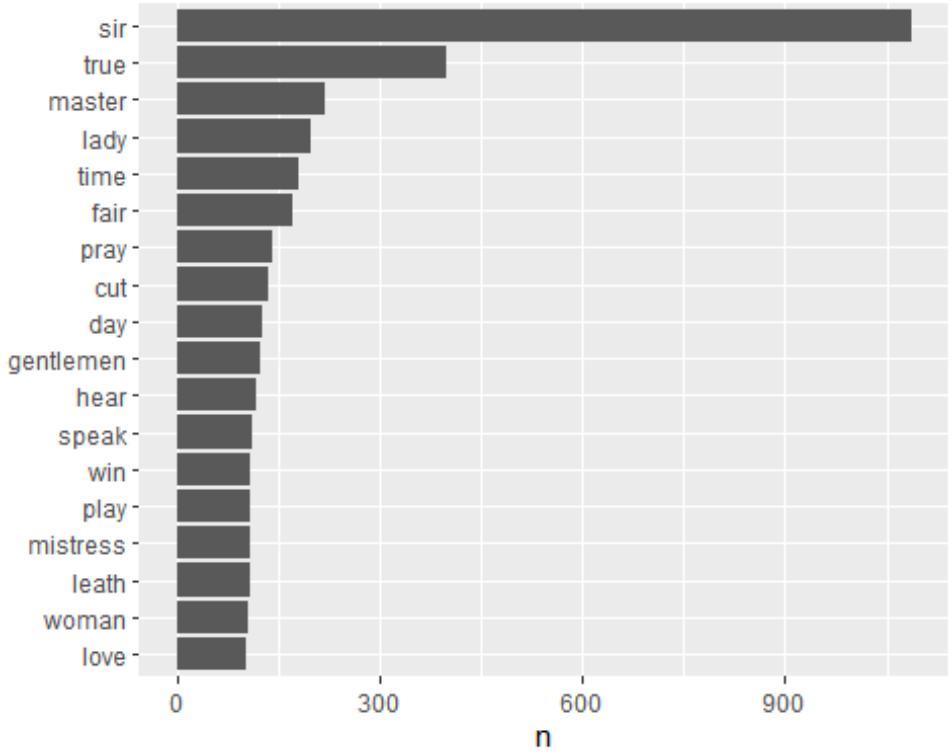
Metni elde ettikten sonra yukarıda incelediğimiz 3 eseri parçaladık. Parçalama işleminden sonra duraklama kelimelerini yani tek başına anlam ifade etmeyen ifadeleri çıkarmamız gerek.

```
data(stop_words)

tidy_benjonson <- tidy_benjonson %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords)

## Joining, by = "word"
## Joining, by = "word"
```

Artık duraklama kelimeleri veri setinden çıkartıldı. Şimdi kelimelerin frekanslarını ortaya çıkartalım.

```
tidy_benjonson %>%
  count(word, sort = TRUE)

## # A tibble: 12,425 x 2
##    word          n
##    <chr>     <int>
##  1 sir        1087
##  2 true        400
##  3 master      221
##  4 lady        198
##  5 time        180
##  6 fair        171
##  7 pray        142
##  8 cut         136
##  9 day         127
## 10 gentlemen   123
## # ... with 12,415 more rows

library(ggplot2)

tidy_benjonson %>%
```

```
  count(word, sort = TRUE) %>%
  filter(n > 100) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)
```



DETAYLI DUYGU ANALIZI

```
nrc_joy <- get_sentiments("nrc") %>%
  filter(sentiment == "joy")

tidy_benjonson %>%
  inner_join(nrc_joy) %>%
  count(word, sort = TRUE)

## Joining, by = "word"

## # A tibble: 318 x 2
##    word       n
##    <chr>  <int>
##  1 true     400
##  2 pray     142
##  3 love     103
##  4 faith     85
##  5 art       81
##  6 friend    76
##  7 hope      73
##  8 money     66
##  9 marry     55
## 10 sweet     53
## # ... with 308 more rows
```

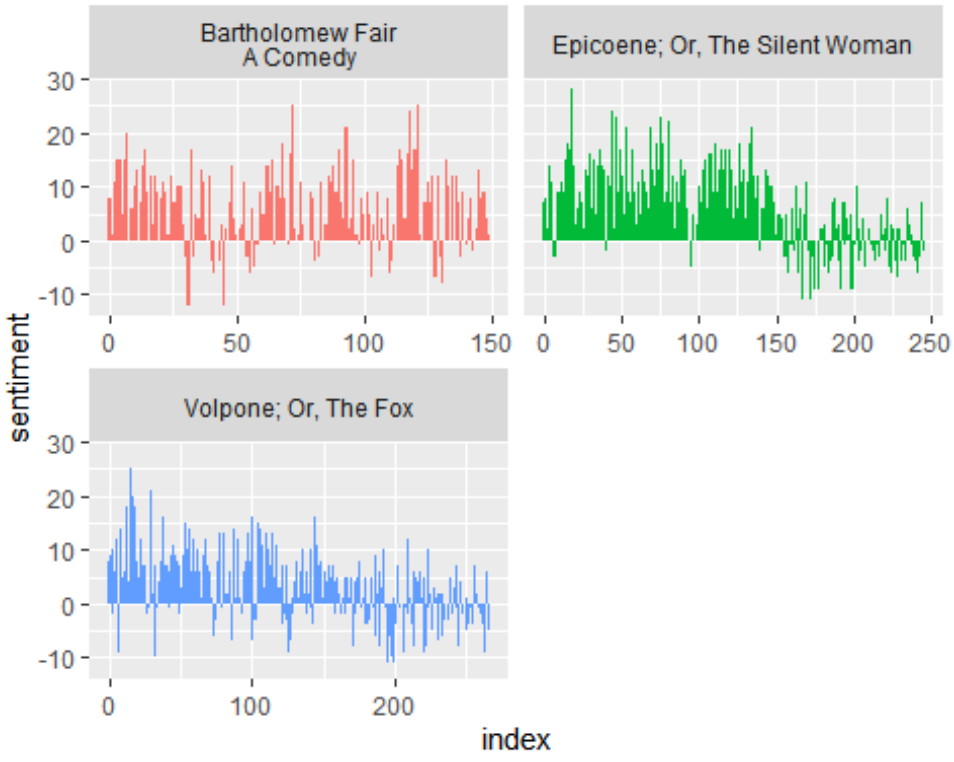
Burada Ben Jonson'un eserlerini seçtik daha sonra NRC sözlüğündeki joy duygusunu seçip count olarak kelimeleri saydırdık.

NOT: %/% bu operator tam sayı bölmeye yarıyor. Biz elimizdeki metni 40 satırlık parçalara ayırmak istiyoruz. Daha sonra bunun duygu analizini yapmak istiyoruz her bir kitap için.

```
library(tidyr)

## Warning: package 'tidyr' was built under R version 4.0.5

benjonson_sentiment <- tidy_benjonson %>%
  inner_join(get_sentiments("nrc")) %>%
  count(title, index = linenumber %/% 40, sentiment) %>%
  spread(sentiment, n , fill = 0) %>%
  mutate(sentiment = positive - negative)

## Joining, by = "word"
```

Şimdi ise daha detaylı bir duygu analizi yapmak için eserlerimiz ile NRC duygu sözlüğünü birleştirdik. Yani bizim elimizdeki kitaplar olumlu mu olumsuz mu bunları karşılaştırmak istiyoruz. Bunu yapmak için öncelikle eserlerimizi 40 satırlık tam sayıya böldük. Daha sonra NRC sözlüğü içerisindeki pozitif ve negatif olarak tanımlı duyguları eserlerimizdeki kelime frekanslarına göre bir değişken olarak tanımladık daha sonra pozitif ve negatif frekanslar arasındaki farkı alıp görselleştirdik.

```
library(ggplot2)

ggplot(benjonson_sentiment, aes(index, sentiment, fill = title)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~title, ncol = 2, scales = "free_x")
```



Yukarıdaki grafiklere bakıldığında Ben Jonson'a ait eserlerin olay örgüsünün, hikayenin gidişatı üzerinde daha olumlu veya olumsuz duygulara doğru nasıl değiştiğini görebiliriz. Genel olarak 3 eserde de olumlu ve olumsuz bölümler olduğu görülmektedir.Ben Jonson'un incelediğimiz 3 eserinin 2'sinin son kısımlarının eserin tamamına göre daha olumsuz olduğu görülüyor.

KELIME BULUTU

Kelime bulutu bir sekil etrafinda kelimeleri frekanslarina göre dagitiyor.

```
library(reshape2)

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths

library(wordcloud)

## Warning: package 'wordcloud' was built under R version 4.0.5

## Loading required package: RColorBrewer

tidy_benjonson %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray60"),
                   max.words = 100)
```

Burada Ben Jonson'un negatiften pozitife en sık kullandığı kelimeleri görmekteyiz.Burada yazının boyutunun büyümesi onların eserlerde daha sık kullanılan kelimeler olduğunu ifade etmektedir. Kelimelerin tonları saydamlaştıkça anlamlarının pozitifleştiğini söyleyebiliriz.

TF-IDF ISTATISTIGI (KELIME VE BELGE SIKLIGI ANALIZI)

```
library(dplyr)
library(tidytext)

book_words <- benjonson %>%
  unnest_tokens(word, text) %>%
  count(title, word, sort = TRUE)

total_words <- book_words %>%
  group_by(title) %>%
  summarize(total = sum(n))

book_words <- left_join(book_words, total_words)

## Joining, by = "title"

book_words

## # A tibble: 23,316 x 4
##    title                          word      n total
##    <chr>                          <chr> <int> <int>
##  1 "Epicoene; Or, The Silent Woman" the    1715 50105
##  2 "Volpone; Or, The Fox"           the    1705 49696
##  3 "Bartholomew Fair\nA Comedy"     the    1489 40796
##  4 "Volpone; Or, The Fox"           of     1424 49696
##  5 "Epicoene; Or, The Silent Woman" and    1393 50105
##  6 "Epicoene; Or, The Silent Woman" of     1356 50105
##  7 "Volpone; Or, The Fox"           and    1249 49696
##  8 "Bartholomew Fair\nA Comedy"     and    1226 40796
##  9 "Epicoene; Or, The Silent Woman" a      1127 50105
## 10 "Epicoene; Or, The Silent Woman" to     1115 50105
## # ... with 23,306 more rows
```

Burada **örneğin** Epicoene; Or, The Silent Woman kitabi icerisinde **the** kelimesi 1715 kere geçmiş ve toplamda Epicoene; Or, The Silent Woman kitabında 50105 kelime geçmiş.

Her bir kitap için bir kullanım sıklığı var ve farklı davranışlar sergiliyor mu sergilemiyor mu bunu araştırmak istiyoruz. Bunun için her bir kitap icin kelimelerin dağılımını gösteren bir grafik çizdirelim.

```
library(ggplot2)

ggplot(book_words, aes(n/total, fill = title)) +
  geom_histogram(show.legend = FALSE) +
  xlim(NA, 0.0009) +
  facet_wrap(~title, ncol = 2, scales = "free_y")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 438 rows containing non-finite values (stat_bin).

## Warning: Removed 3 rows containing missing values (geom_bar).
```

Görüldüğü üzere 3 kitapta da benzer kelime frekansı kullanılmış. Yani Ben Jonson'un 3 kitabinda da belirli ölçülerde kitaplarini yazdigini söyleyebiliriz.

ZIPF KANUNU

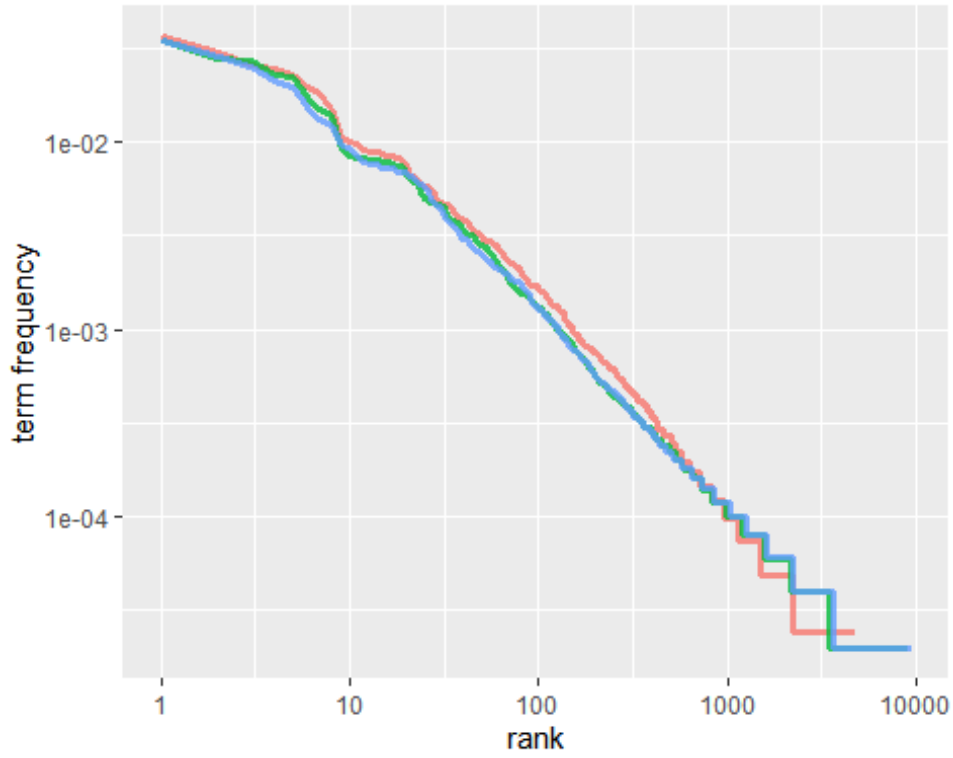
```
freq_by_rank <- book_words %>%
  group_by(title) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total) %>%
  ungroup()

freq_by_rank
```

```
## # A tibble: 23,316 x 6
##    title                          word      n total  rank `term frequency`
##    <chr>                          <chr> <int> <int> <int>            <dbl>
##  1 "Epicoene; Or, The Silent Woman" the    1715 50105     1           0.0342
##  2 "Volpone; Or, The Fox"           the    1705 49696     1           0.0343
##  3 "Bartholomew Fair\nA Comedy"     the    1489 40796     1           0.0365
##  4 "Volpone; Or, The Fox"           of     1424 49696     2           0.0287
##  5 "Epicoene; Or, The Silent Woman" and    1393 50105     2           0.0278
##  6 "Epicoene; Or, The Silent Woman" of     1356 50105     3           0.0271
##  7 "Volpone; Or, The Fox"           and    1249 49696     3           0.0251
##  8 "Bartholomew Fair\nA Comedy"     and    1226 40796     2           0.0301
##  9 "Epicoene; Or, The Silent Woman" a      1127 50105     4           0.0225
## 10 "Epicoene; Or, The Silent Woman" to     1115 50105     5           0.0223
## # ... with 23,306 more rows
```

Görmüş olduğumuz üzere **1715/50105=0.0342** şeklinde bir değer verdi. Yani Epicoene; Or, The Silent Woman kitabında the kelimesinin geçme oranı **0.0342'**dir diyor kelime frekansı.

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = title)) +
```

```
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```



Zipf yasası bir şeyin artarken bir şeyin azalmasını gösteriyordu bu grafik de bunu gösteriyor. Görmüş olduğumuz üzere rank arttıkca kelime frekansı azalıyor.

Şimdi rankların alt kümesini alalım.

```
rank_subset <- freq_by_rank %>%
  filter(rank < 500,
         rank > 10)

lm(log10(`term frequency`) ~ log10(rank), data = rank_subset)

##
## Call:
## lm(formula = log10(`term frequency`) ~ log10(rank), data = rank_subset)
##
## Coefficients:
## (Intercept)  log10(rank)
##      -0.760       -1.061
```

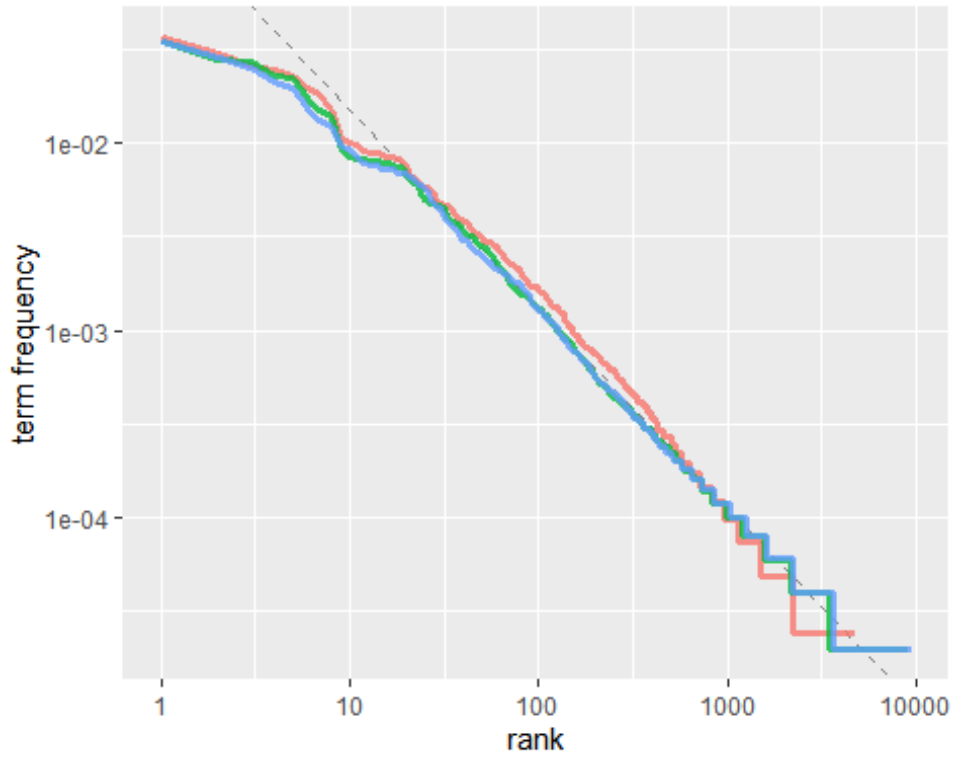
Burada katsayıları bulmak üzere bir regresyon modeli oluşturuldu. Regresyon modelinde de kelime frekansının logaritmik hali bağımlı değişken ve rank da bağımsız değişken olarak alındı.

Bu katsayılar ne işimize yarayacak?

Yukarıda kurmuş olduğumuz regresyon modeli ile aşağıdaki kesikli bir biçimde gösterilen eğimi bulmuş olduk.

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = title)) +
  geom_abline(intercept = -0.760, slope = -1.061,
```

```
            color = "gray50", linetype = 2) +
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```



bind_tf_idf() FONKSİYONU

Bu fonksiyon bir bağlama işlemi yapar. Her bir kelimenin kitabın ve frekansa göre.

```
book_tf_idf <- book_words %>%
  bind_tf_idf(word, title, n)

book_tf_idf
```

```
## # A tibble: 23,316 x 7
##    title                             word      n total     tf   idf tf_idf
##    <chr>                             <chr> <int> <int>  <dbl> <dbl>  <dbl>
##  1 "Epicoene; Or, The Silent Woman" the    1715 50105 0.0342     0      0
##  2 "Volpone; Or, The Fox"           the    1705 49696 0.0343     0      0
##  3 "Bartholomew Fair\nA Comedy"     the    1489 40796 0.0365     0      0
##  4 "Volpone; Or, The Fox"           of     1424 49696 0.0287     0      0
##  5 "Epicoene; Or, The Silent Woman" and    1393 50105 0.0278     0      0
##  6 "Epicoene; Or, The Silent Woman" of     1356 50105 0.0271     0      0
##  7 "Volpone; Or, The Fox"           and    1249 49696 0.0251     0      0
##  8 "Bartholomew Fair\nA Comedy"     and    1226 40796 0.0301     0      0
##  9 "Epicoene; Or, The Silent Woman" a      1127 50105 0.0225     0      0
## 10 "Epicoene; Or, The Silent Woman" to     1115 50105 0.0223     0      0
## # ... with 23,306 more rows
```

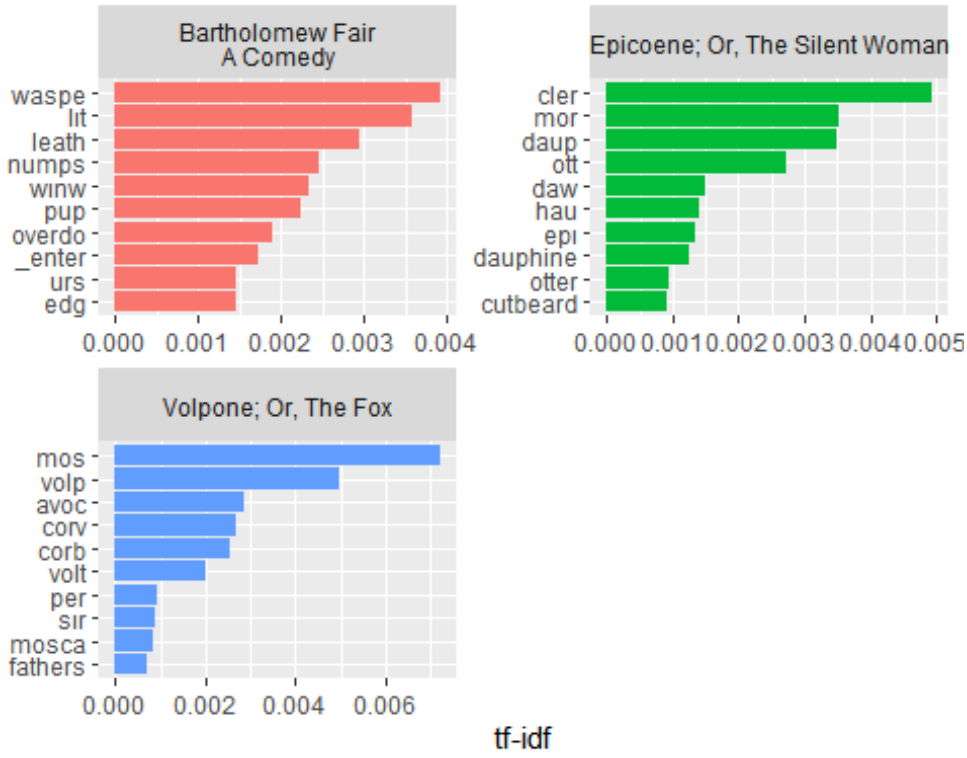
```
book_tf_idf %>%
  select(-total) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 23,316 x 6
##    title                             word      n      tf   idf  tf_idf
##    <chr>                             <chr> <int>   <dbl> <dbl>   <dbl>
```

```
##  1 "Volpone; Or, The Fox"           mos     327 0.00658  1.10 0.00723
##  2 "Volpone; Or, The Fox"           volp    224 0.00451  1.10 0.00495
##  3 "Epicoene; Or, The Silent Woman" cler    225 0.00449  1.10 0.00493
##  4 "Bartholomew Fair\nA Comedy"     waspe   146 0.00358  1.10 0.00393
##  5 "Bartholomew Fair\nA Comedy"     lıt     133 0.00326  1.10 0.00358
##  6 "Epicoene; Or, The Silent Woman" mor     160 0.00319  1.10 0.00351
##  7 "Epicoene; Or, The Silent Woman" daup    159 0.00317  1.10 0.00349
##  8 "Bartholomew Fair\nA Comedy"     leath   109 0.00267  1.10 0.00294
##  9 "Volpone; Or, The Fox"           avoc    130 0.00262  1.10 0.00287
## 10 "Epicoene; Or, The Silent Woman" ott     124 0.00247  1.10 0.00272
## # ... with 23,306 more rows
```

Burada tf_idf değerine göre azalan şekilde göstermesi için ayarladık. Burada tf_idf değerinin en yüksek olduğu kitap **Volpone; Or, The Fox** kitabıymış ve en yüksek çıkan oran da **Mos** kelimesi imiş. 2. en yüksek tf_idf degeri ise **Epicoene; Or, The Silent Woman** kitabıymış ve en yüksek çıkan oran da **Cler** kelimesi olmuş. Bu 2 kelime de bir karakter ismidir.

```
library(forcats)

## Warning: package 'forcats' was built under R version 4.0.5

book_tf_idf %>%
  group_by(title) %>%
  slice_max(tf_idf, n = 10) %>%
  ungroup() %>%
  ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = title)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~title, ncol = 2, scales = "free") +
  labs(x = "tf-idf", y = NULL)
```



Grafiğe baktığımızda tf_idf değerlerine göre **Epicoene; Or, The Silent Woman** eserinde **Cler, Mor, Daup, Otter** gibi kelimeler olduğunu görüyoruz. Çoğunluk olarak isimler mevcut.

TOPIC MODELLING (KONU MODELLEMESI)

Konu modellemesi **örnegin** bir gazetenin her bir sayfasında farklı konulardan olan haberler var. Elimizde pek çok döküman olduğunu ve dökümanların farklı konular içerdiğini düşünelim. Konu modellemesi bu dökümanların ayrılmasında kullanılıyor ve düzgünce bir şekilde ayırmaya çalışıyoruz. Bu yöntem nümerik dataları kümeleme yöntemine benziyor.

LDA YÖNTEMİ

Konu modellemesi icin LDA yaygın bir algoritmadır. LDA'in 2 tane prensibi vardır.

1- Elimizdeki bir metin bir konuyu içermelidir. **Örneğin** elimizde sporla alakalı bir metin olsun. Bunun ekonomi konusuna girme olasılığı cok düşüktür. Ekonomiyi de ilgilendiriyorsa o zaman ekonomi içine girebilir ama spor ekonomiden bagimsiz olduğu için o döküman spor konusuna ait olmalıdır.

2- Dökümanlar bir konuya ait oluyorsa, konular da kendi içlerinde onlari temsil eden kelimelere ait olmalıdır.

UYGULAMA 6 tane kitabımız var. Yani 6 tane dökümanımız var. Biz bu 6 dökümanı sınıflandırmak istiyoruz. Yani doğru bir şekilde tahmin etmek istiyoruz. Hangi sayfa hangi kitaba ait bilmiyoruz. Bunu algoritma yöntemiyle bulabilir miyiz?

```
library(gutenbergr)

## Warning: package 'gutenbergr' was built under R version 4.0.4

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

yazarlar<- gutenberg_download(c(4011,49461,4039,1515,1508,1526))

## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest

## Using mirror http://aleph.gutenberg.org

eserler<- gutenberg_metadata %>%
  filter(author %in% c("Jonson, Ben", "Shakespeare, William"),
         language == "en")



title<- eserler %>%
  select(gutenberg_id, title)

books <- yazarlar %>%
  inner_join(title, by = "gutenberg_id")
```

```
library(tidyr)

## Warning: package 'tidyr' was built under R version 4.0.5

library(dplyr)
library(tidytext)

## Warning: package 'tidytext' was built under R version 4.0.5

mystopwords<-tibble(word = c("ı", "thou", "sır", "thy", "thee", "ı'll", "ıf", "hath", "tis"
, "ın", "ay", "mine", "ıt", "ıs", "la", "toby", "petruchıo", "enter", "vıola", "olıvıa", "p
ortıa", "clown","andrew", "tranıo", "shylock", "katherına", "malvolıo", "antonıo", "exit",
"hortensıo", "bassanıo", "lucentıo", "launcelot", "baptısta", "lorenzo", "marıa", "grumıo",
"exeunt", "gremıo", "fabıan", "kate", "mos","jonson", "cler",  "volp", "cokes", "daw", "mor
", "daup", "nay","quar", "waspe", "lıt", "avoc", "ott", "corv", "corb", "mosca", "john", "j
onson's", "numps", "volt", "wınw", "p", "f", "exıt", "i'faith", "epi","urs", "gobbo", "grac
e", "wıt", "cut", "antonıo", "1", "2", "3", "volpone"))
library(stringr)

# divide into documents, each representing one chapter
by_chapter <- books %>%
  group_by(title) %>%
  mutate(chapter = cumsum(str_detect(
    text, regex(c("^act ", "^scene") ,ignore_case = TRUE)
  ))) %>%
   filter(chapter > 0) %>%
  ungroup() %>%
  unite(document, title, chapter)

## Warning in stri_detect_regex(string, pattern, negate = negate, opts_regex =
## opts(pattern)): longer object length is not a multiple of shorter object length

## Warning in stri_detect_regex(string, pattern, negate = negate, opts_regex =
## opts(pattern)): longer object length is not a multiple of shorter object length

## Warning in stri_detect_regex(string, pattern, negate = negate, opts_regex =
## opts(pattern)): longer object length is not a multiple of shorter object length

# split into words
by_chapter_word <- by_chapter %>%
  unnest_tokens(word, text)

# find document-word counts
word_counts <- by_chapter_word %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords) %>%
  count(document, word, sort = TRUE) %>%
  ungroup()

## Joining, by = "word"
## Joining, by = "word"

word_counts

## # A tibble: 39,366 x 3
##    document                           word      n
##    <chr>                              <chr> <int>
##  1 "Volpone; Or, The Fox_4"           sir     236
##  2 "Epicoene; Or, The Silent Woman_3" sir     228
##  3 "Epicoene; Or, The Silent Woman_3" true    176
##  4 "Epicoene; Or, The Silent Woman_1" sir     102
##  5 "Volpone; Or, The Fox_2"           sir      98
##  6 "Volpone; Or, The Fox_4"           lady     95
##  7 "Epicoene; Or, The Silent Woman_1" true     94
##  8 "Epicoene; Or, The Silent Woman_2" sir      86
##  9 "Bartholomew Fair\nA Comedy_8"     pup      79
```

```
## 10 "Bartholomew Fair\nA Comedy_8"     leath    78
## # ... with 39,356 more rows
```

Burada baktığımızda **örnegin** Volpone; Or, The Fox'un 4. sahnesinde **Lady** kelimesi 95 kere gecmis.

Burada düzenli olan bir metin formatı var. Biz bunu döküman terim matrisine çevirmek istiyoruz.

```
chapters_dtm <- word_counts %>%
  cast_dtm(document, word, n)
chapters_dtm

## <<DocumentTermMatrix (documents: 54, terms: 13901)>>
## Non-/sparse entries: 39366/711288
## Sparsity           : 95%
## Maximal term length: 22
## Weighting          : term frequency (tf)
```

Baktığımızda bu veri seti içerisinde 54 tane döküman oldugunu ve 13901 tane kelime olduğunu görüyoruz. Matrisimizin seyrekliği %95 imiş.

Simdi bu matrisi LDA fonksiyonuna sokuyoruz fakat elimizde 6 kitap vardi. Dolayisiyla 6 farkli konum olmasını istiyorum. LDA algoritmasında bir k argümanı mevcut. k=6 olduğu zaman LDA 6 konuya göre modelliyor. k'yı arttırdıkça konu sayısı artıyor.

```
library(topicmodels)

## Warning: package 'topicmodels' was built under R version 4.0.5

chapters_lda <- LDA(chapters_dtm, k = 6, control = list(seed = 1234))
chapters_lda

## A LDA_VEM topic model with 6 topics.
```

Kelime konu olasılıklarına bakmak için beta olasılığına bakmalıyız. Burada tidy fonksiyonu ile beta olasılıklarını görebiliyoruz.

beta her bir konuda her bir kelimenin olasılığını temsil eder. Kelime konu olasılığına bakmak istediğimizde beta'ya bakmalıyız.

Konu değişkeninde 6 kategorimiz var. term'de ise bu konuda geçen kelimeleri bize veriyor. beta bize 6 konu için **örneğin** sir kelimesinin geçme olasılığını bize veriyor.

```
chapter_topics <- tidy(chapters_lda, matrix = "beta")
chapter_topics

## # A tibble: 83,406 x 3
##    topic term     beta
##    <int> <chr>   <dbl>
##  1     1 sir   0.0192
##  2     2 sir   0.0277
##  3     3 sir   0.0170
##  4     4 sir   0.0216
##  5     5 sir   0.00632
##  6     6 sir   0.0317
##  7     1 true  0.00821
##  8     2 true  0.00215
##  9     3 true  0.00198
## 10     4 true  0.00361
## # ... with 83,396 more rows
```

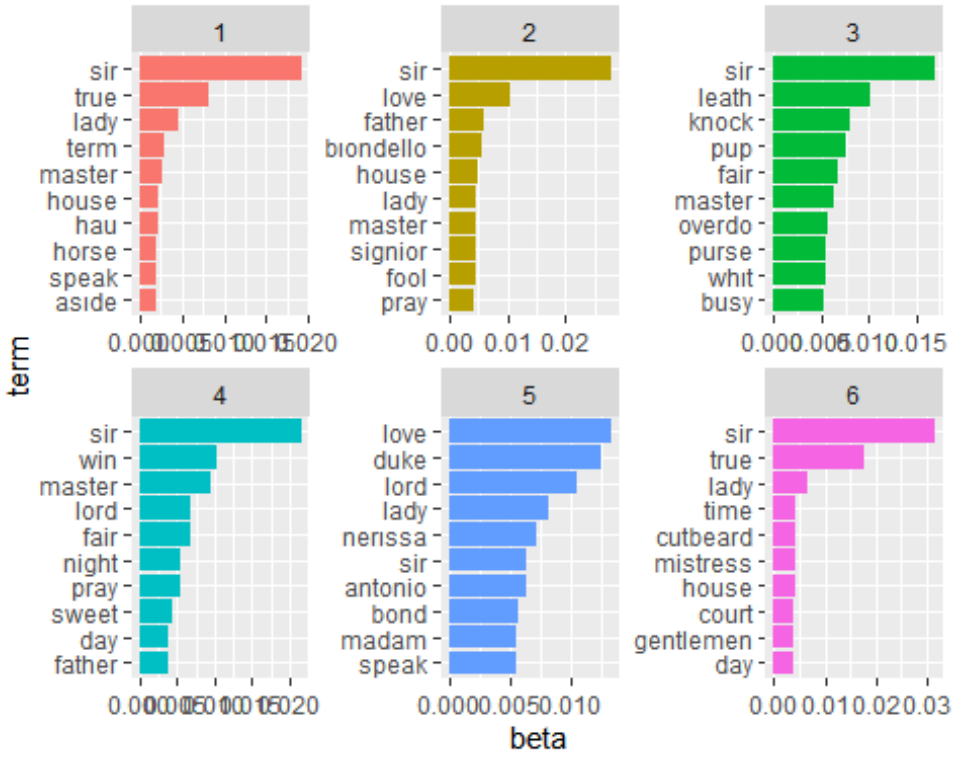
Simdi dplyr ile top_n() fonksiyonunu kullanarak ilk 10 gözlemi görebiliriz. ggplot2 ile de bunu görsellestirelim.

```
top_terms <- chapter_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

top_terms
```

```
## # A tibble: 60 x 3
##    topic term      beta
##    <int> <chr>    <dbl>
##  1     1 sir    0.0192
##  2     1 true   0.00821
##  3     1 lady   0.00448
##  4     1 term   0.00299
##  5     1 master 0.00253
##  6     1 hau    0.00207
##  7     1 house  0.00207
##  8     1 horse  0.00203
##  9     1 asıde  0.00199
## 10     1 speak  0.00199
## # ... with 50 more rows
```

Burada **master** kelimesi 1. konu için 0.002 olasılığına sahipmiş.

```
library(ggplot2)

top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```



Burada her bir konuda en fazla olasılıkları temsil eden kelimeler var. Burada çıkan kelimeler aslında 6 kitapta en çok geçen kelimeler. Her bir kitapta kendine özel kelimeler bunlar.

DÖKUMAN BASINA SINIFLAMA OLASILIGI

Bizim döküman konu olasılığımız **gamma**dır. Aşağıda her bir dökümanın 1'den 6. konuya kadar olan olasılıklarını bize gösteriyor.

```
chapters_gamma <- tidy(chapters_lda, matrix = "gamma")
chapters_gamma
```

```
## # A tibble: 324 x 3
##    document                          topic      gamma
##    <chr>                             <int>      <dbl>
##  1 "Volpone; Or, The Fox_4"              1 1.00
##  2 "Epicoene; Or, The Silent Woman_3"    1 1.00
##  3 "Epicoene; Or, The Silent Woman_1"    1 0.00000551
##  4 "Volpone; Or, The Fox_2"              1 0.0000102
##  5 "Epicoene; Or, The Silent Woman_2"    1 0.0000105
##  6 "Bartholomew Fair\nA Comedy_8"        1 0.00000620
##  7 "Volpone; Or, The Fox_3"              1 0.00000987
##  8 "Bartholomew Fair\nA Comedy_1"        1 0.00000982
##  9 "Bartholomew Fair\nA Comedy_3"        1 0.00000678
## 10 "Bartholomew Fair\nA Comedy_5"        1 0.00000897
## # ... with 314 more rows
```

Biz eserlerimizi bölümlere parçalamıştık. Yani bölümleri parçalamıştık.

```
chapters_gamma <- chapters_gamma %>%
  separate(document, c("title", "chapter"), sep = "_", convert = TRUE)

chapters_gamma
```

```
## # A tibble: 324 x 4
##    title                            chapter topic      gamma
##    <chr>                              <int> <int>      <dbl>
##  1 "Volpone; Or, The Fox"                 4     1 1.00
##  2 "Epicoene; Or, The Silent Woman"       3     1 1.00
##  3 "Epicoene; Or, The Silent Woman"       1     1 0.00000551
##  4 "Volpone; Or, The Fox"                 2     1 0.0000102
##  5 "Epicoene; Or, The Silent Woman"       2     1 0.0000105
##  6 "Bartholomew Fair\nA Comedy"           8     1 0.00000620
##  7 "Volpone; Or, The Fox"                 3     1 0.00000987
##  8 "Bartholomew Fair\nA Comedy"           1     1 0.00000982
##  9 "Bartholomew Fair\nA Comedy"           3     1 0.00000678
## 10 "Bartholomew Fair\nA Comedy"           5     1 0.00000897
## # ... with 314 more rows
```

```
chapter_classifications <- chapters_gamma %>%
  group_by(title, chapter) %>%
  slice_max(gamma) %>%
  ungroup()
chapter_classifications
```

```
## # A tibble: 54 x 4
##    title                            chapter topic gamma
##    <chr>                              <int> <int> <dbl>
##  1 "Bartholomew Fair\nA Comedy"           1     4 1.00
##  2 "Bartholomew Fair\nA Comedy"           2     3 1.00
##  3 "Bartholomew Fair\nA Comedy"           3     3 0.508
##  4 "Bartholomew Fair\nA Comedy"           4     3 1.00
##  5 "Bartholomew Fair\nA Comedy"           5     3 1.00
##  6 "Bartholomew Fair\nA Comedy"           6     3 0.999
##  7 "Bartholomew Fair\nA Comedy"           7     2 0.609
```

```
##  8 "Bartholomew Fair\nA Comedy"           8     3 1.00
##  9 "Epicoene; Or, The Silent Woman"       1     6 1.00
## 10 "Epicoene; Or, The Silent Woman"       2     6 1.00
## # ... with 44 more rows
```

Yukarıda gama olasılıklarına göre bölümlere bir baktık. Burada da her bir olasılıklara göre eserlerin en iyi gözlemlerine bakıyor.

```
book_topics <- chapter_classifications %>%
  count(title, topic) %>%
  group_by(title) %>%
  slice_max(n, n = 1) %>%
  ungroup() %>%
  transmute(consensus = title, topic)

chapter_classifications %>%
  inner_join(book_topics, by = "topic") %>%
  filter(title != consensus)
```

```
## # A tibble: 58 x 5
##    title                        chapter topic gamma consensus
##    <chr>                          <int> <int> <dbl> <chr>
##  1 "Bartholomew Fair\nA Comedy"       1     4 1.00  Volpone; Or, The Fox
##  2 "Bartholomew Fair\nA Comedy"       7     2 0.609 The Taming of the Shrew
##  3 "Epicoene; Or, The Silent W~       1     6 1.00  Volpone; Or, The Fox
##  4 "Epicoene; Or, The Silent W~       2     6 1.00  Volpone; Or, The Fox
##  5 "Epicoene; Or, The Silent W~       3     1 1.00  Volpone; Or, The Fox
##  6 "The Merchant of Venice"           1     5 0.989 Twelfth Night; Or, What You~
##  7 "The Merchant of Venice"           1     5 0.989 Volpone; Or, The Fox
##  8 "The Merchant of Venice"           2     5 0.565 Twelfth Night; Or, What You~
##  9 "The Merchant of Venice"           2     5 0.565 Volpone; Or, The Fox
## 10 "The Merchant of Venice"           3     5 0.999 Twelfth Night; Or, What You~
## # ... with 48 more rows
```

Yukarıda yaptığımız işlemleri doğru bir şekilde sınıflandırdık mı diye kontrol etmeliyiz. Yukarıda gördüğümüz üzere **The Merchant of Venice'in 1. bölümüne baktıgımızda bu konu 5'e aitmiş ve %98lük bir olasilikla varmiş ve bunun karsılıgı da Twelfth Night; Or, What You Will'e karsılık geliyormuş.**

augment fonksiyonu ile LDA işlemi yapmıştık ve döküman terim matrisine çevirmiştik her bir bölümü. Asagida bunlari karsilastiracagiz.

```
assignments <- augment(chapters_lda, data = chapters_dtm)
assignments
```

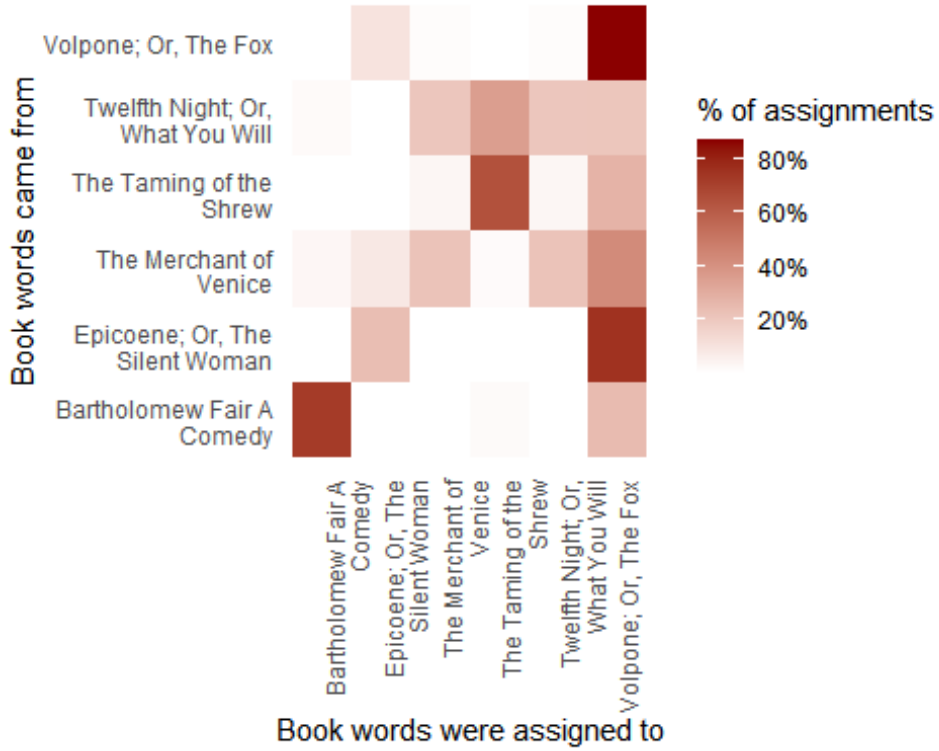
```
## # A tibble: 39,366 x 4
##    document                          term  count .topic
##    <chr>                             <chr> <dbl>  <dbl>
##  1 "Volpone; Or, The Fox_4"          sir     236      1
##  2 "Epicoene; Or, The Silent Woman_3" sir    228      1
##  3 "Epicoene; Or, The Silent Woman_1" sir    102      6
##  4 "Volpone; Or, The Fox_2"          sir      98      4
##  5 "Epicoene; Or, The Silent Woman_2" sir     86      6
##  6 "Bartholomew Fair\nA Comedy_8"    sir      60      3
##  7 "Volpone; Or, The Fox_3"          sir      78      6
##  8 "Bartholomew Fair\nA Comedy_1"    sir      39      4
##  9 "Bartholomew Fair\nA Comedy_3"    sir      55      4
## 10 "Bartholomew Fair\nA Comedy_5"    sir      45      3
## # ... with 39,356 more rows
```

```
assignments <- assignments %>%
  separate(document, c("title", "chapter"),
           sep = "_", convert = TRUE) %>%
```

```
  inner_join(book_topics, by = c(".topic" = "topic"))

assignments
```

```
## # A tibble: 52,785 x 6
##    title                    chapter term  count .topic consensus
##    <chr>                      <int> <chr> <dbl>  <dbl> <chr>
##  1 "Volpone; Or, The Fox"         4 sir     236      1 "Volpone; Or, The Fox"
##  2 "Epicoene; Or, The Sile~       3 sir     228      1 "Volpone; Or, The Fox"
##  3 "Epicoene; Or, The Sile~       1 sir     102      6 "Epicoene; Or, The Silen~
##  4 "Epicoene; Or, The Sile~       1 sir     102      6 "Volpone; Or, The Fox"
##  5 "Volpone; Or, The Fox"         2 sir      98      4 "Volpone; Or, The Fox"
##  6 "Epicoene; Or, The Sile~       2 sir      86      6 "Epicoene; Or, The Silen~
##  7 "Epicoene; Or, The Sile~       2 sir      86      6 "Volpone; Or, The Fox"
##  8 "Bartholomew Fair\nA Co~       8 sir      60      3 "Bartholomew Fair\nA Com~
##  9 "Volpone; Or, The Fox"         3 sir      78      6 "Epicoene; Or, The Silen~
## 10 "Volpone; Or, The Fox"         3 sir      78      6 "Volpone; Or, The Fox"
## # ... with 52,775 more rows
```

```
library(scales)

assignments %>%
  count(title, consensus, wt = count) %>%
  mutate(across(c(title, consensus), ~str_wrap(., 20))) %>%
  group_by(title) %>%
  mutate(percent = n / sum(n)) %>%
  ggplot(aes(consensus, title, fill = percent)) +
  geom_tile() +
  scale_fill_gradient2(high = "darkred", label = percent_format()) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        panel.grid = element_blank()) +
  labs(x = "Book words were assigned to",
       y = "Book words came from",
       fill = "% of assignments")
```



Yukarıdaki görsele baktığımızda LDA algoritmasının eserleri ayırt etme konusunda pek iyi bir ayrışım yaptığını söylemek mümkün değil. Bunun sebebi LDA algoritmasının yanlış sınıfandırdığı eserlerin benzetmiş olduğu eserler ile ortak kelimelere sahip olmasıdır. **Örneğin** The Taming of the Shrew ile Volpone, Or, The Fox eserlerinin benzer olduğu görülmektedir. Bu grafik bize çok net olarak 2 yazarın eserlerinde ortaklıklar olduğunu göstermektedir.

Yanlış eşleştirilen kitaplar ve kelimeler

```
wrong_words <- assignments %>%
  filter(title != consensus)

wrong_words
```

```
## # A tibble: 28,750 x 6
##    title                       chapter term  count .topic consensus
##    <chr>                         <int> <chr> <dbl>  <dbl> <chr>
##  1 "Epicoene; Or, The Silent~       3 sir     228      1 Volpone; Or, The Fox
##  2 "Epicoene; Or, The Silent~       1 sir     102      6 Volpone; Or, The Fox
##  3 "Epicoene; Or, The Silent~       2 sir      86      6 Volpone; Or, The Fox
##  4 "Volpone; Or, The Fox"           3 sir      78      6 Epicoene; Or, The Sile~
##  5 "Bartholomew Fair\nA Come~       1 sir      39      4 Volpone; Or, The Fox
##  6 "Bartholomew Fair\nA Come~       3 sir      55      4 Volpone; Or, The Fox
##  7 "Twelfth Night; Or, What ~      11 sir      37      2 The Taming of the Shrew
##  8 "Twelfth Night; Or, What ~      14 sir      28      5 The Merchant of Venice
##  9 "Twelfth Night; Or, What ~      14 sir      28      5 Volpone; Or, The Fox
## 10 "Twelfth Night; Or, What ~      10 sir      29      2 The Taming of the Shrew
## # ... with 28,740 more rows
```

```
wrong_words %>%
  count(title, consensus, term, wt = count) %>%
  ungroup() %>%
  arrange(desc(n))
```

```
## # A tibble: 22,352 x 4
##    title                             consensus                     term      n
##    <chr>                             <chr>                         <chr> <dbl>
##  1 "Epicoene; Or, The Silent Woman"  Volpone; Or, The Fox           sir     416
##  2 "Epicoene; Or, The Silent Woman"  Volpone; Or, The Fox           true    319
##  3 "Twelfth Night; Or, What You Will" The Taming of the Shrew       sir     135
##  4 "Bartholomew Fair\nA Comedy"      Volpone; Or, The Fox           win      97
##  5 "Bartholomew Fair\nA Comedy"      Volpone; Or, The Fox           sir      94
##  6 "Volpone; Or, The Fox"            Epicoene; Or, The Silent Woman sir      78
##  7 "Epicoene; Or, The Silent Woman"  Volpone; Or, The Fox           mast~    76
##  8 "Epicoene; Or, The Silent Woman"  Volpone; Or, The Fox           hau      64
##  9 "Epicoene; Or, The Silent Woman"  Volpone; Or, The Fox           lady     63
## 10 "Epicoene; Or, The Silent Woman"  Volpone; Or, The Fox           epı      61
## # ... with 22,342 more rows
```

The Merchant of Venice Eserinin Analizi

Kütüphaneler

```
library(tidytext)
library(dplyr)
library(tidyverse)
library(gutenbergr)
library(tidyr)  # Düzenli veri oluşturma
library(stopwords) # İstenmeyen kelimeleri çıkartma
library(ggplot2)
library(wordcloud) # Kelime Bulutu
library(wordcloud2) #Kelime Bulutu
```

Öncelikle gerekli kütüphanelerimizi indirelim.

Gutenberg Meta Data'da birçok yazarın neredeyse tüm eserleri bulunmaktadır.

```
gutenberg_metadata
```

```
## # A tibble: 51,997 x 8
##    gutenberg_id title  author  gutenberg_autho~ language gutenberg_books~ rights
##           <int> <chr>  <chr>              <int> <chr>    <chr>            <chr>
##  1            0  <NA>  <NA>                  NA en       <NA>             Publi~
##  2            1 "The ~ Jeffer~              1638 en       United States L~ Publi~
##  3            2 "The ~ United~                 1 en       American Revolu~ Publi~
##  4            3 "John~ Kenned~              1666 en       <NA>             Publi~
##  5            4 "Linc~ Lincol~                 3 en       US Civil War     Publi~
##  6            5 "The ~ United~                 1 en       American Revolu~ Publi~
##  7            6 "Give~ Henry,~                 4 en       American Revolu~ Publi~
##  8            7 "The ~ <NA>                   NA en       <NA>             Publi~
##  9            8 "Abra~ Lincol~                 3 en       US Civil War     Publi~
## 10            9 "Abra~ Lincol~                 3 en       US Civil War     Publi~
## # ... with 51,987 more rows, and 1 more variable: has_text <lgl>
```

Analiz yapacağımız William Shakespeare ait Venedik Taciri eserini indirelim.

```
venice <- gutenberg_download((1515), meta_fields = "title")

## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harvest

## Using mirror http://aleph.gutenberg.org
```

Duraklama Kelimeleri

```
tidy_venice <- venice %>%
  unnest_tokens(word, text) %>%
   anti_join(stop_words); tidy_venice

## # A tibble: 9,400 x 3
##    gutenberg_id title                 word
##           <int> <chr>                 <chr>
##  1         1515 The Merchant of Venice merchant
##  2         1515 The Merchant of Venice venıce
##  3         1515 The Merchant of Venice william
##  4         1515 The Merchant of Venice shakespeare
##  5         1515 The Merchant of Venice dramatıs
##  6         1515 The Merchant of Venice personae
##  7         1515 The Merchant of Venice duke
##  8         1515 The Merchant of Venice venıce
##  9         1515 The Merchant of Venice prınce
## 10         1515 The Merchant of Venice morocco
## # ... with 9,390 more rows
```

Anti join komuduyla kullandığımız stop words eserimizdeki duraklama kelimeleri çıkartıyor.

En Çok Kullanılan Kelimeler

```
head(tidy_venice %>%
  count(word, sort = TRUE),20)

## # A tibble: 20 x 2
##    word          n
##    <chr>     <int>
##  1 ı           653
##  2 portıa      129
##  3 shylock     106
##  4 thou        103
##  5 bassanıo     84
##  6 launcelot    83
##  7 lorenzo      81
##  8 thee         65
##  9 love         60
## 10 gratıano     59
## 11 jew          59
## 12 thy          59
## 13 antonıo      57
## 14 hath         52
## 15 ıf           52
## 16 enter        50
## 17 nerıssa      48
## 18 antonio      47
## 19 ı'll         47
## 20 lord         45
```

En çok kullanılan kelimelerde Portia, Shylock, Bassanio, Launcelot, Lorenzo, Gratiano, Antonio ve Nerissa gibi isimler sıkça var. Bu isimler incelediğimiz eserin kahramanlarıdır. Ayrıca I,I'll, if,thou, thee,

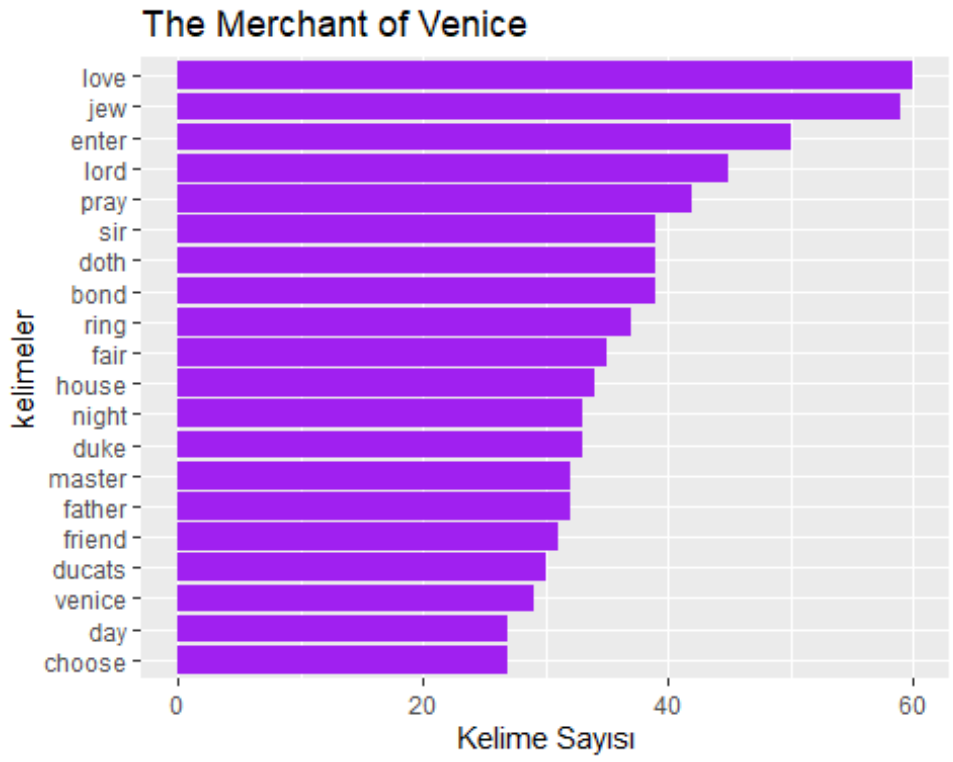
thy gibi zamirler çokça kullanıldığı için incelenilen eserde diyalogların fazla olduğunu yani bir tiyatro oyunu olduğu sonucuna varabiliriz.

Daha iyi analiz edebilmek için eserdeki kahramanları ve bu zamirleri eserimizden çıkartalım.

```
mystopwords <- tibble(word =
c("ı","thou","thee","ıf","thy","ın","ı'll","ıs","ıt","thou","thee","launcelot","nerıssa",
"shylock","portıa","bassanio","bassanıo","antonio","antonıo","salarıno","jessıca","gratıano
","lorenzo","salanıo","gobbo","hath"))

tidy_venice <- venice %>%
  unnest_tokens(word, text) %>%
   anti_join(stop_words) %>%
  anti_join(mystopwords)

## Joining, by = "word"
## Joining, by = "word"

head(tidy_venice %>%
  count(word, sort = TRUE),20)

## # A tibble: 20 x 2
##    word       n
##    <chr>  <int>
##  1 love      60
##  2 jew       59
##  3 enter     50
##  4 lord      45
##  5 pray      42
##  6 bond      39
##  7 doth      39
##  8 sir       39
##  9 ring      37
## 10 fair      35
## 11 house     34
## 12 duke      33
## 13 night     33
## 14 father    32
## 15 master    32
## 16 friend    31
## 17 ducats    30
## 18 venice    29
## 19 choose    27
## 20 day       27
```

Kelime Kullanılma Sıklığı Histogram

```
tidy <- venice %>%
  unnest_tokens(word,text) %>%
  count(word,sort=T) %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords) %>%
  filter(!word %in% stop_words$word,

         str_detect(word, "[a-z]"))

tidy %>%
  head(20)%>%
  ggplot(aes(reorder(word,n),n))+
  geom_col(fill= "purple")+
  coord_flip()+
  labs(x="kelimeler",
       y="Kelime Sayısı",
       title= "The Merchant of Venice")
```



Grafiği incelediğimizde yahudi, dua etmek gibi kelimeler fazlaca kullanılmıştır. Bu eserde dine yer verildiğini söyleyebiliriz. Ayrıca tahvil, duke(para birimi),borç almak gibi kelimelerde sıkça kullanılmıştır. Eserde borç alışverişi olduğunu ve bu şekilde de eserin bunun üzerinden konuya alındığını düşünebiliriz. Efendim, lord, master gibi kelimeler kullanıldığı için kahramanlar arası bir hiyerarşi olduğu sonucuna varabiliriz.

Duygu Analizi

Bing

```
tidy %>%
  inner_join(get_sentiments("bing"))
```

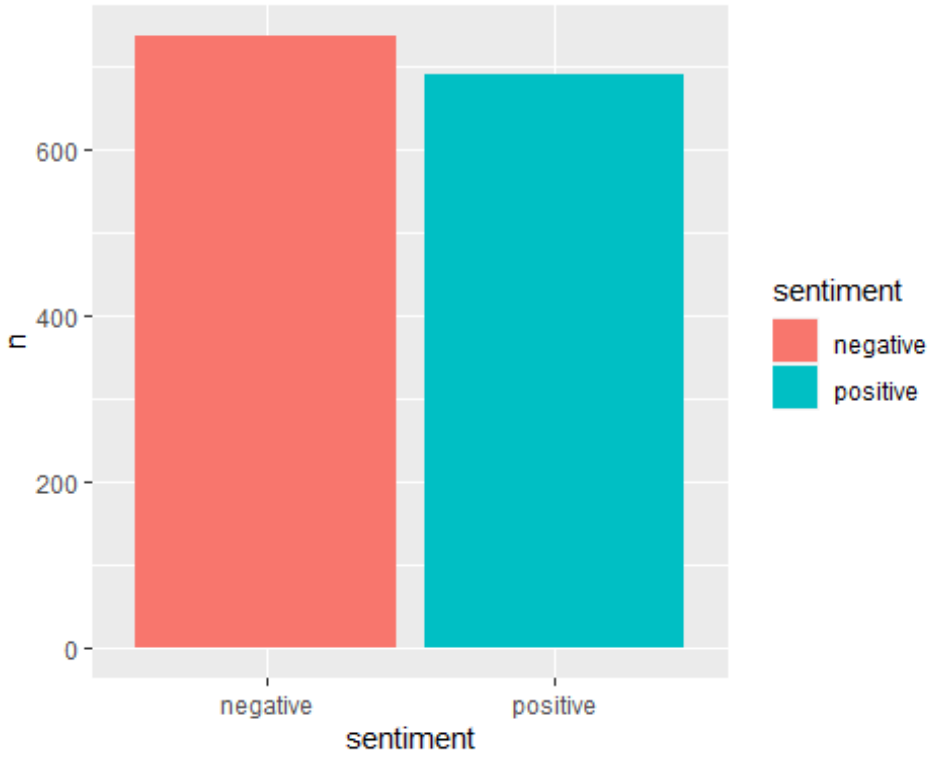
```
## Joining, by = "word"
```

```
## # A tibble: 550 x 3
##    word        n sentiment
##    <chr>   <int> <chr>
##  1 love       60 positive
##  2 fair       35 positive
##  3 master     32 positive
##  4 sweet      23 positive
##  5 fortune    21 positive
##  6 heaven     16 positive
##  7 fear       15 negative
##  8 gold       15 positive
##  9 devil      14 negative
## 10 faith      14 positive
## # ... with 540 more rows
```

Bing komudu ile yapılan duygu analizinde kelimelerin pozitif ya da negatif anlamda olduğu çıkarabiliriz.

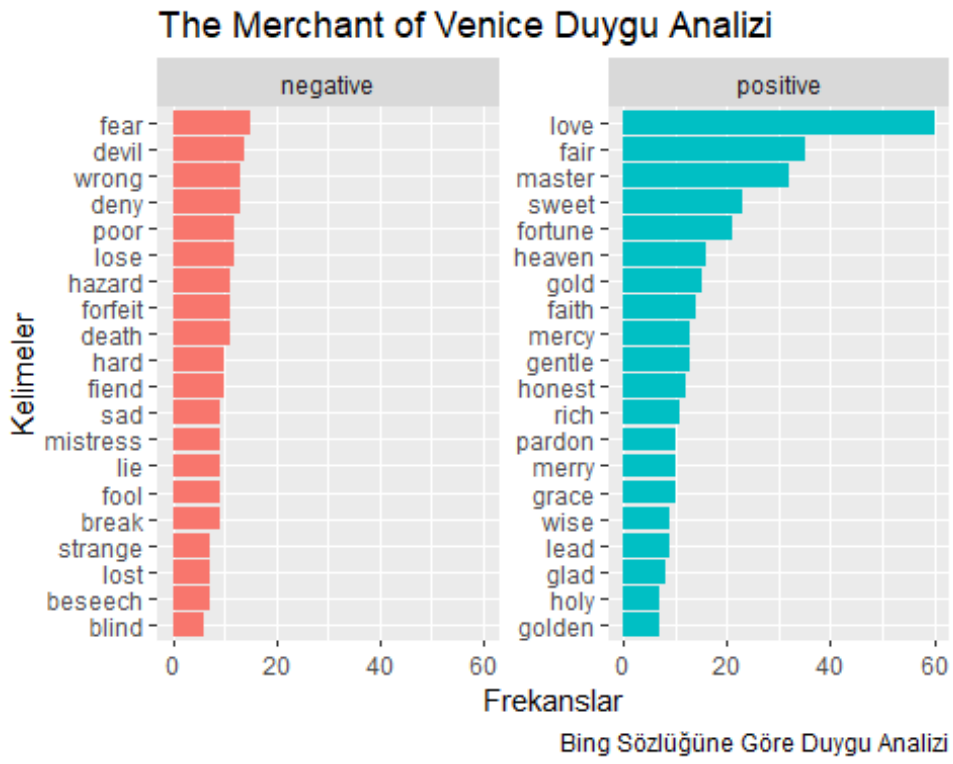
```
tidy %>%
  inner_join(get_sentiments("bing")) %>%
  ggplot(aes(sentiment, n, fill= sentiment))+
  geom_col()
```

```
## Joining, by = "word"
```



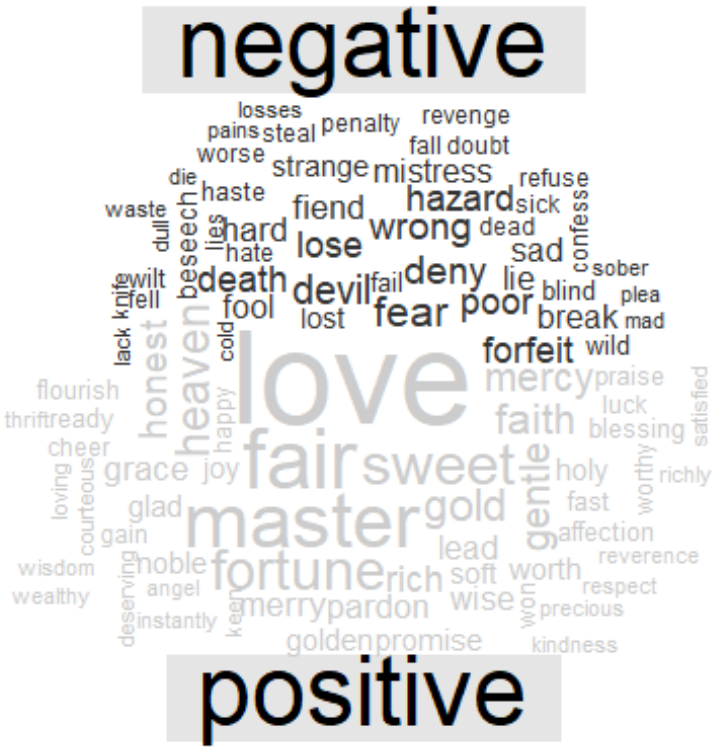
Grafiğe göre negatif duygularla pozitif duygular arası çok büyük bir fark olmadığından eserin trajikomik türünde yazıldığını düşünebiliriz.

```
rbind(


tidy %>%
  inner_join(get_sentiments("bing")) %>%
  arrange(-n) %>%
  filter(sentiment == "positive") %>%
  head(20),


tidy %>%
  inner_join(get_sentiments("bing")) %>%
  arrange(-n) %>%
  filter(sentiment == "negative") %>%
  head(20)) %>%
  ggplot(aes(reorder(word,n),n, fill=sentiment)) +
  geom_col(show.legend = FALSE)+
  facet_wrap(~sentiment,scales = "free_y") +
  coord_flip() +
  labs( x = "Kelimeler" , y = "Frekanslar" , title = "The Merchant of Venice Duygu Analizi"
, caption = "Bing Sözlüğüne Göre Duygu Analizi" )

## Joining, by = "word"
## Joining, by = "word"
```



Negatif ve pozitiflik durumuna göre en çok kullanılan 20 duygu içeren kelimenin frekans tablosundan ölüm,cennet,şeytan, korku gibi kelimeler kullanıldığını görüyoruz. Eserde sıkça ölüm konusunun yer aldığını düşünebiliriz. Ayrıca önceki yorumlamalarımızda borç durumundan olay döndüğünü belirtmiştik. Buna göre borç yüzünden tehditler döndüğünü ya da bu borç olayının yaşam mücadelesine dönüştüğünü düşünebiliriz.

```
library(reshape2)

tidy_venice %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray20", "gray80"),
                   max.words = 100)

## Joining, by = "word"
```



Pozitif ve negatif kelimelerin oranlarını inceleyelim.

```
bingnegative <- get_sentiments("bing") %>%
  filter(sentiment == "negative")
bingnegative

## # A tibble: 4,781 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # ... with 4,771 more rows
```

İlk olarak Bing sözlüğünde yer alan negatif kelimeleri filtreledik.

```
wordcounts <- tidy_venice %>%
  group_by(title, gutenberg_id) %>%
  summarize(words = n());wordcounts
```

```
## # A tibble: 1 x 3
## # Groups:   title [1]
##   title                 gutenberg_id words
##   <chr>                        <int> <int>
## 1 The Merchant of Venice         1515  7409
```

İncelediğimiz eserde toplam kaç kelime olduğunu gösterdik.

```
tidy_venice %>%
  semi_join(bingnegative) %>%
  group_by(title, gutenberg_id) %>%
  summarize(negativewords = n()) %>%
  left_join(wordcounts, by = c("title", "gutenberg_id")) %>%
  mutate(ratio = negativewords/words) %>%
  filter(gutenberg_id != 0) %>%
  slice_max(ratio, n = 1) %>%
  ungroup()
```

```
## Joining, by = "word"
```

```
## `summarise()` has grouped output by 'title'. You can override using the `.groups`
argument.
```

```
## # A tibble: 1 x 5
##   title                 gutenberg_id negativewords words  ratio
##   <chr>                        <int>         <int> <int>  <dbl>
## 1 The Merchant of Venice         1515           737  7409 0.0995
```

Eserdeki olumsuz kelimelerle tüm kelimelerin arasında oran kurduk. Eserimizde toplam 737 olumsuz kelime var ve toplam 7409 kelime olduğundan bunları oranladığımızda eserin %9,9'luk kısmının negatif anlamlı kelimelerden oluştuğunu görüyoruz.

Aynı hesabı pozitif kelimeler için de yaptık.

```
bingpositive <- get_sentiments("bing") %>%
  filter(sentiment == "positive")
bingpositive
```

```
## # A tibble: 2,005 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 abound      positive
##  2 abounds     positive
##  3 abundance   positive
##  4 abundant    positive
##  5 accessable  positive
##  6 accessible  positive
##  7 acclaim     positive
##  8 acclaimed   positive
##  9 acclamation positive
## 10 accolade    positive
## # ... with 1,995 more rows
```

Tekrardan eserimizdeki pozitif kelimeleri filtreledik.

```
wordcounts <- tidy_venice %>%
  group_by(title, gutenberg_id) %>%
  summarize(words = n());wordcounts
```

```
## `summarise()` has grouped output by 'title'. You can override using the `.groups`
argument.
```

```
## # A tibble: 1 x 3
## # Groups:   title [1]
##   title                 gutenberg_id words
##   <chr>                        <int> <int>
## 1 The Merchant of Venice         1515  7409
```

```
tidy_venice %>%
  semi_join(bingpositive) %>%
  group_by(title, gutenberg_id) %>%
  summarize(positivewords = n()) %>%
  left_join(wordcounts, by = c("title", "gutenberg_id")) %>%
  mutate(ratio = positivewords/words) %>%
  filter(gutenberg_id != 0) %>%
  slice_max(ratio, n = 1) %>%
  ungroup()
```

```
## Joining, by = "word"
```

```
## `summarise()` has grouped output by 'title'. You can override using the `.groups`
argument.
```

```
## # A tibble: 1 x 5
##   title                 gutenberg_id positivewords words  ratio
##   <chr>                        <int>         <int> <int>  <dbl>
## 1 The Merchant of Venice         1515           690  7409 0.0931
```

Eserdeki pozitif kelimelerle tüm kelimelerin arasında oran kurduk. Eserimizde toplam 690 pozitif kelime var ve toplam 7409 kelime olduğundan bunları oranladığımızda eserin %9,3'lük kısmının pozitif anlamlı kelimelerden oluştuğunu görüyoruz.

Buna göre eserimizde negatif kelimelerin yoğunluğu pozitif kelimelerin yoğunluğundan daha fazla olduğu sonucuna varabiliriz.

Kelime Bulutu

```
tidy_venice %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100,colors=brewer.pal(8, "Dark2")))
```

Kelime bulutunu incelediğimizde merhamet, hukuk, yasa, yargılamak, kölelik gibi kelimelerinde yer aldığını görüyoruz. Olay örgüsünün devamında bu borçlanma olayının mahkemeleye dayandığını düşünebiliriz.

NGRAM (n=2)

```
venice_bigrams <- venice %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)
venice_bigrams
```

```
## # A tibble: 20,448 x 3
##    gutenberg_id title                 bigram
##           <int> <chr>                 <chr>
##  1         1515 The Merchant of Venice the merchant
##  2         1515 The Merchant of Venice merchant of
##  3         1515 The Merchant of Venice of venıce
##  4         1515 The Merchant of Venice <NA>
##  5         1515 The Merchant of Venice by william
##  6         1515 The Merchant of Venice william shakespeare
##  7         1515 The Merchant of Venice <NA>
##  8         1515 The Merchant of Venice <NA>
##  9         1515 The Merchant of Venice <NA>
## 10         1515 The Merchant of Venice <NA>
## # ... with 20,438 more rows
```

n-gram ile ardaşık sözlükleri yan yana görebiliriz. Bu sayede analizde daha anlamlı sonuçlar çıkartabiliriz.

```
venice_bigrams %>%
  count(bigram, sort = TRUE)
```

```
## # A tibble: 12,854 x 2
##    bigram      n
##    <chr>   <int>
##  1 <NA>     1510
##  2 ı am       70
##  3 ı will     53
##  4 of the     52
##  5 ı have     51
##  6 in the     51
##  7 of my      35
##  8 ı pray     34
##  9 my lord    28
## 10 you are    28
## # ... with 12,844 more rows
```

İkili kelime gruplarından, ben, yapacağım, sahibim, dua ederim, lordum, sen gibi kelimeler sıkça kullanıldığından bolca diyaloglu yani tiyatro oyunu olduğunu bu 2'li ngram yardımıyla çıkarabiliriz.

```
bigrams_separated <- venice_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")


bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word1 %in% mystopwords$word) %>%
  filter(!word2 %in% mystopwords$word)


# new bigram counts:
bigram_counts <- bigrams_filtered %>%
  count(word1, word2, sort = TRUE)


head(bigram_counts,30)
```

```
## # A tibble: 30 x 3
##    word1     word2      n
##    <chr>     <chr>  <int>
##  1 <NA>      <NA>    1510
##  2 thousand  ducats    16
##  3 portıa's  house      7
##  4 dear      friend     4
##  5 learned   judge      4
##  6 cornets   enter      3
##  7 doth      teach      3
##  8 fair      lady       3
##  9 fair      portia     3
## 10 fourscore ducats     3
## # ... with 20 more rows
```

En çok kullanulan ikili kelimelere baktığımızda borcun bin duka altın olduğu sonucuna varabiliriz.Portia'nın evi de sıkça kullanılmış. Olay örgüsünün Portia'nın evinde de sıkça geçtiğini düşünebiliriz. Dürüst Portia, tatlı, Portia gibi ikili kelimeler kullanılmış. Portia'nın iyi bir karakter olduğunu düşünebiliriz.Ayrıca yargıç öğrendi, yargıç katibi, dürüst yargıç gibi ikili kelimeler kullanılmış. Yani bu borç olayının mahkemeye taşındığını düşünebiliriz.

```
bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")


bigrams_united
```

```
## # A tibble: 3,011 x 3
##    gutenberg_id title                 bigram
##           <int> <chr>                 <chr>
##  1         1515 The Merchant of Venice NA NA
##  2         1515 The Merchant of Venice william shakespeare
##  3         1515 The Merchant of Venice NA NA
##  4         1515 The Merchant of Venice NA NA
##  5         1515 The Merchant of Venice NA NA
##  6         1515 The Merchant of Venice NA NA
##  7         1515 The Merchant of Venice dramatıs personae
##  8         1515 The Merchant of Venice NA NA
##  9         1515 The Merchant of Venice morocco suitor
## 10         1515 The Merchant of Venice arragon suitor
## # ... with 3,001 more rows
```

Bu sefer 3 ard arda gelen kelime gruplarını inceleyelim.

```
venice %>%
  unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
  separate(trigram, c("word1", "word2", "word3","word4"), sep = " ") %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word,
         !word3 %in% stop_words$word,
         !word1 %in% mystopwords$word,
         !word2 %in% mystopwords$word,
         !word3 %in% mystopwords$word) %>%
  count(word1, word2, word3,sort = TRUE)
```

```
## Warning: Expected 4 pieces. Missing pieces filled with `NA` in 16223 rows [1, 2,
## 4, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, ...].
```

```
## # A tibble: 254 x 4
##    word1   word2   word3       n
##    <chr>   <chr>   <chr>   <int>
##  1 <NA>    <NA>    <NA>     1572
##  2 ding    dong    bell        2
##  3 judge   mark    jew         2
##  4 scene   2       belmont     2
##  5 scene   3       venice      2
##  6 adieu   tears   exhibit     1
##  7 alas    fifteen wives       1
##  8 ancient roman   honour      1
##  9 axe     bear    half        1
## 10 bait    fish    withal      1
## # ... with 244 more rows
```

Üçlü kelime gruplarında frekans sayıları az geldiğinden eser hakkında doğru yorum çıkarmamız mümkün değil.

İkili kelime gruplarından belli kelimeleri seçerek daha çok fikir sahibi olabiliriz.

```
bigrams_filtered %>%
  filter(word2 == "house") %>%
  count(title, word1, sort = TRUE)
```

```
## # A tibble: 9 x 3
##   title                 word1         n
##   <chr>                 <chr>     <int>
## 1 The Merchant of Venice portia's      7
## 2 The Merchant of Venice shylock's     3
## 3 The Merchant of Venice antonio's     1
## 4 The Merchant of Venice father's      1
## 5 The Merchant of Venice jew's         1
## 6 The Merchant of Venice pent          1
## 7 The Merchant of Venice sober         1
## 8 The Merchant of Venice treasure      1
## 9 The Merchant of Venice unquiet       1
```

2. kelimeyi house olarak seçtiğimizde olayların Portia'nın, Shylock'un Antonio'nun, babanın, yahudinin evlerinde geçtiğini görebiliriz. Tekrar sayısına baktığımızda ise en çok Portia'nın evinde olayların geçtiğini görebiliyoruz.

```
bigrams_filtered %>%
  filter(word1 == "bond") %>%
  count(title, word2, sort = TRUE)
```

```
## # A tibble: 4 x 3
##   title                  word2       n
##   <chr>                  <chr>   <int>
## 1 The Merchant of Venice doth        2
## 2 The Merchant of Venice expires     1
## 3 The Merchant of Venice speak       1
## 4 The Merchant of Venice thrice      1
```

1. Kelimeyi bond olarak seçtiğimizde senetin bittiğini görüyoruz. Diğer çıkan 2. kelimeler bond kelimesi ile birleştirildiğinde anlamsız çıkmaktadır.

```
bigrams_filtered %>%
  filter(word2 == "jew") %>%
  count(title, word1, sort = TRUE)
```

```
## # A tibble: 16 x 3
##    title                  word1         n
##    <chr>                  <chr>     <int>
##  1 The Merchant of Venice mark          2
##  2 The Merchant of Venice rich          2
##  3 The Merchant of Venice answer        1
##  4 The Merchant of Venice contented     1
##  5 The Merchant of Venice currish       1
##  6 The Merchant of Venice daniel        1
##  7 The Merchant of Venice dog           1
##  8 The Merchant of Venice faithless     1
##  9 The Merchant of Venice father        1
## 10 The Merchant of Venice gentle        1
## 11 The Merchant of Venice harsh         1
## 12 The Merchant of Venice peril         1
## 13 The Merchant of Venice sweet         1
## 14 The Merchant of Venice tarry         1
## 15 The Merchant of Venice villain       1
## 16 The Merchant of Venice wealthy       1
```

Burada yahudiden önce gelen kelimelere baktığımızda rich jew 2 kere kullanılmış. Yani yahudi zengin biri borcun bu yahudiden alındığını düşünebiliriz.

```
bigram_tf_idf <- bigrams_united %>%
  count(title, bigram) %>%
  bind_tf_idf(bigram, title, n) %>%
  arrange(desc(tf_idf))


bigram_tf_idf
```

```
## # A tibble: 1,391 x 6
##    title                 bigram               n       tf   idf tf_idf
##    <chr>                 <chr>            <int>    <dbl> <dbl>  <dbl>
##  1 The Merchant of Venice 2 belmont            2 0.000664     0      0
##  2 The Merchant of Venice 2 venice             1 0.000332     0      0
##  3 The Merchant of Venice 3 venice             2 0.000664     0      0
##  4 The Merchant of Venice 4 belmont            1 0.000332     0      0
##  5 The Merchant of Venice 7 belmont            1 0.000332     0      0
##  6 The Merchant of Venice 8 venice             1 0.000332     0      0
##  7 The Merchant of Venice 9 belmont            1 0.000332     0      0
##  8 The Merchant of Venice a'leven widows       1 0.000332     0      0
##  9 The Merchant of Venice acquaintance hie     1 0.000332     0      0
## 10 The Merchant of Venice act 1                1 0.000332     0      0
## # ... with 1,381 more rows
```

```
AFINN <- get_sentiments("afinn")
AFINN
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # ... with 2,467 more rows
```

```
not_words <- bigrams_separated %>%
  filter(word1 == "no") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word1,word2, value, sort = TRUE)
not_words
```

```
## # A tibble: 11 x 4
##    word1 word2   value     n
##    <chr> <chr>   <dbl> <int>
##  1 no    no         -1     3
##  2 no    better      2     2
##  3 no    doubt      -1     1
##  4 no    good        3     1
##  5 no    ill        -2     1
##  6 no    matter      1     1
##  7 no    mercy       2     1
##  8 no    pray        1     1
##  9 no    revenge    -2     1
## 10 no    tears      -2     1
## 11 no    wrong      -2     1
```

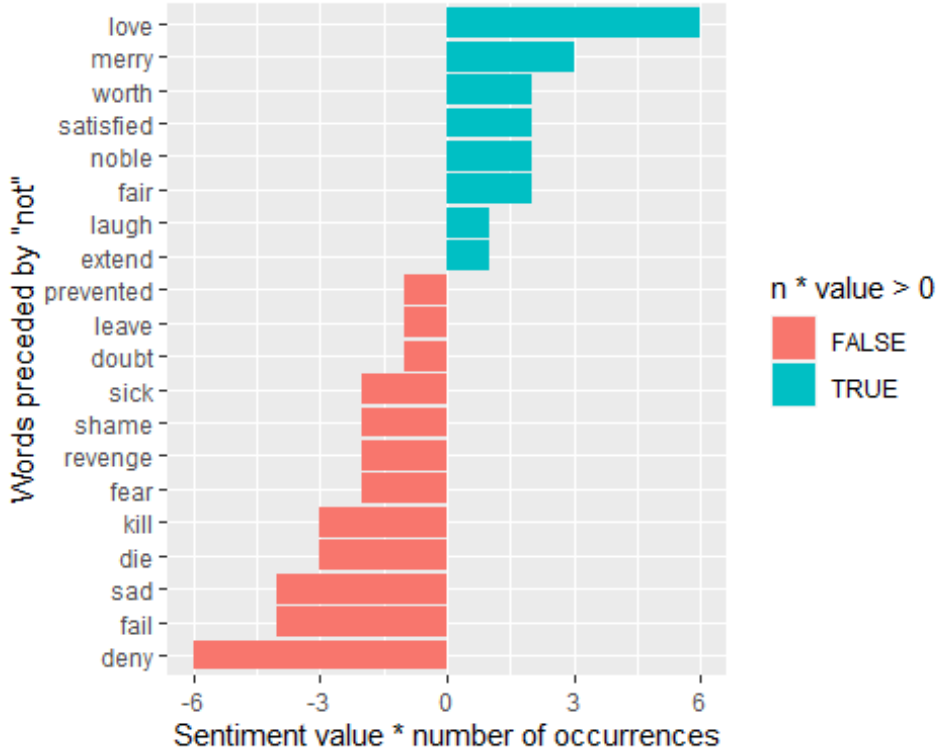
AFINN sözlüğündeki kelimelerin başında no eklediğimizde daha daha iyi değil, iyi değil gibi ikili gruplar yer almaktadır. Buna göre kahramanlarımız için bazı olayların seyrinin onlar için iyi gitmediğini söyleyebiliriz.

```
not_words <- bigrams_separated %>%
  filter(word1 == "not") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word1,word2, value, sort = TRUE)
not_words
```

```
## # A tibble: 20 x 4
##    word1 word2     value     n
##    <chr> <chr>     <dbl> <int>
##  1 not   deny         -2     3
##  2 not   fail         -2     2
##  3 not   love          3     2
##  4 not   sad          -2     2
##  5 not   die          -3     1
##  6 not   doubt        -1     1
##  7 not   extend        1     1
##  8 not   fair          2     1
##  9 not   fear         -2     1
## 10 not   kill         -3     1
## 11 not   laugh         1     1
## 12 not   leave        -1     1
## 13 not   merry         3     1
## 14 not   noble         2     1
## 15 not   prevented    -1     1
## 16 not   revenge      -2     1
## 17 not   satisfied     2     1
## 18 not   shame        -2     1
## 19 not   sick         -2     1
## 20 not   worth         2     1
```

Ayrıca AFINN sözlüğündeki duygu kelimelerinin başına not geldiğinde olumsuz olarak sınıfladırılan kelimelerin olumlu, olumlu olarak sınıflandırılan kelimelerinde olumsuz yapıya dönüştüğünü görmekteyiz.

```
not_words %>%
  mutate(contribution = n * value) %>%
  arrange(desc(abs(contribution))) %>%
  head(20) %>%
  mutate(word2 = reorder(word2, contribution)) %>%
  ggplot(aes(n * value, word2, fill = n * value > 0)) +
  geom_col(show.legend = T) +
  labs(x = "Sentiment value * number of occurrences",
       y = "Words preceded by \"not\"")
```



Yukarıdaki grafikte not sözcüğünden sonra gelen kelimeler gösterilmiş. Daha önce de bahsettiğimiz gibi olumlu kelimeleri olumsuz, olumsuz kelimeleri olumluya çevirmişti. Yani anlamda değişiklik ortaya çıkmıştı. Bu grafikte aslında ondan önce gelen anlam değiştiren sözcükleri algılamadığını görüyoruz. Yani aşk olumlu bir şey ama aşk yok dediğimizde olumsuza dönmesi gerekiyor ancak duygu analizi bunu ayırmıyor. Bu durum aslında analizlerde yanlış yorumlamaya yol açıyor.

Örümcek grafiği oluşturalım.

```
bigram_counts
```

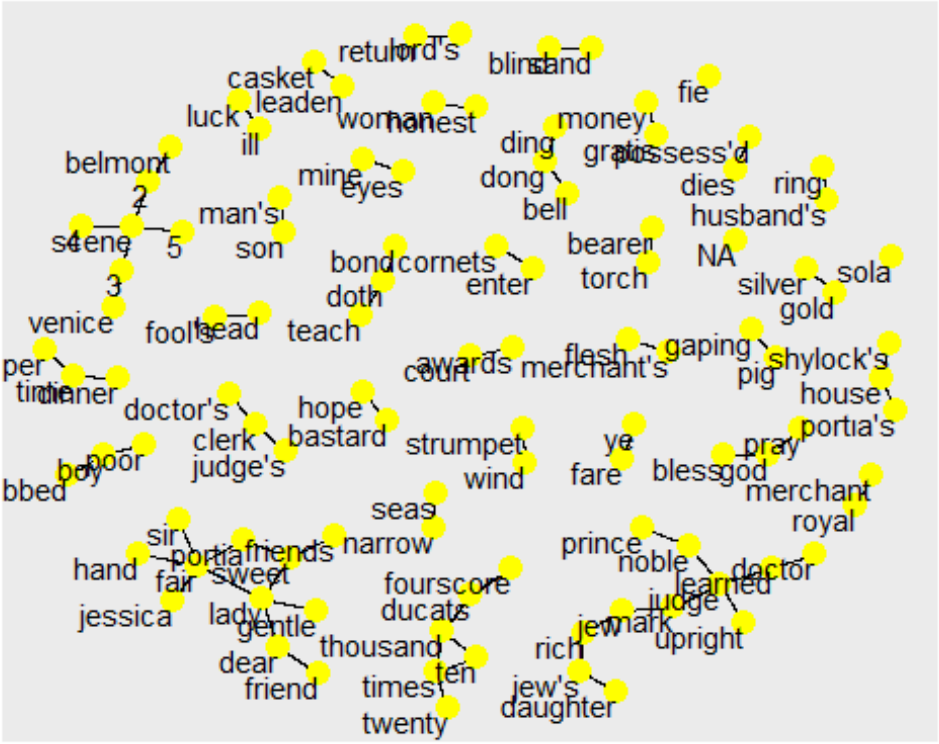
```
## # A tibble: 1,391 x 3
##    word1     word2      n
##    <chr>     <chr>  <int>
##  1 <NA>      <NA>    1510
##  2 thousand  ducats    16
##  3 portıa's  house      7
##  4 dear      friend     4
##  5 learned   judge      4
##  6 cornets   enter      3
##  7 doth      teach      3
##  8 fair      lady       3
##  9 fair      portia     3
## 10 fourscore ducats     3
## # ... with 1,381 more rows
```

```
bigram_graph <- bigram_counts %>%
  filter(n > 1) %>%
  graph_from_data_frame()
```

```
## Warning in graph_from_data_frame(.): In `d' `NA' elements were replaced with
## string "NA"
```

Örümcek grafiği oluşturabilmek için kelimelerden bir data frame oluşturmamız gerekiyor.

```
library(ggraph)

## Warning: package 'ggraph' was built under R version 4.0.5

set.seed(2017)


ggraph(bigram_graph, layout = "fr") +
  geom_edge_link() +
  geom_node_point(color = "yellow", size = 4) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```



Şimdi kelimeler arası kolerasyon inceleyelim. Kahramanlarla da ilgileneceğimiz için kendi oluşturduğumuz stopwords kelimelerinden kahramları çıkartıyoruz.

```
mystopwords1 <- tibble(word =
c("ı","thou","thee","ıf","thy","ın","ı'll","ıs","ıt","thou","thee"))
```

Metni parçalara bölelim. Bunun sebebi her bir bölümün frekansına ve kolerasyonunu incelemek.

```
venice_section_words <- venice %>%
  mutate(section = row_number() %/% 10) %>%
  filter(section > 0) %>%
  unnest_tokens(word, text) %>%
  filter(!word %in% stop_words$word,
         !word %in% mystopwords1$word)


venice_section_words
```

```
## # A tibble: 8,326 x 4
##    gutenberg_id title                   section word
##           <int> <chr>                     <dbl> <chr>
##  1         1515 The Merchant of Venice        1 duke
##  2         1515 The Merchant of Venice        1 venıce
##  3         1515 The Merchant of Venice        1 prınce
##  4         1515 The Merchant of Venice        1 morocco
##  5         1515 The Merchant of Venice        1 suitor
##  6         1515 The Merchant of Venice        1 portia
##  7         1515 The Merchant of Venice        1 prınce
##  8         1515 The Merchant of Venice        1 arragon
##  9         1515 The Merchant of Venice        1 suitor
## 10         1515 The Merchant of Venice        1 portia
## # ... with 8,316 more rows
```

Kelime eşlerinin frekanslarını bulduk. Bu sayede hangi kelimelerin birbirleyile bağlantılı olduğunu görebiliriz.

```
library(widyr)
```

```
## Warning: package 'widyr' was built under R version 4.0.5
```

```
word_pairs <- venice_section_words %>%
  pairwise_count(word, section, sort = TRUE)
```

```
## Warning: `distinct_()` was deprecated in dplyr 0.7.0.
## Please use `distinct()` instead.
## See vignette('programming') for more help
```

```
## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.
```

```
word_pairs
```

```
## # A tibble: 132,904 x 3
##    item1     item2         n
##    <chr>     <chr>     <dbl>
##  1 nerıssa   portıa       30
##  2 portıa    nerıssa      30
##  3 portıa    shylock      20
##  4 gobbo     launcelot    20
##  5 launcelot gobbo        20
##  6 shylock   portıa       20
##  7 salarıno  salanıo      19
##  8 salanıo   salarıno     19
##  9 bond      shylock      19
## 10 shylock   bond         19
## # ... with 132,894 more rows
```

Bond kelimesine göre kelime eşlerini görelim.

```
word_pairs %>%
  filter(item1 == "bond")
```

```
## # A tibble: 306 x 3
##    item1 item2        n
##    <chr> <chr>    <dbl>
##  1 bond  shylock     19
##  2 bond  portıa      11
##  3 bond  bassanıo     7
##  4 bond  antonıo      6
##  5 bond  jew          6
##  6 bond  forfeit      6
##  7 bond  hear         5
##  8 bond  judge        5
##  9 bond  law          5
## 10 bond  antonio      4
## # ... with 296 more rows
```

Senetler yani borç olayları Shylock, Portia, Bassanio, Antonio, Yahudi ve yargıç arasında geçmektedir.

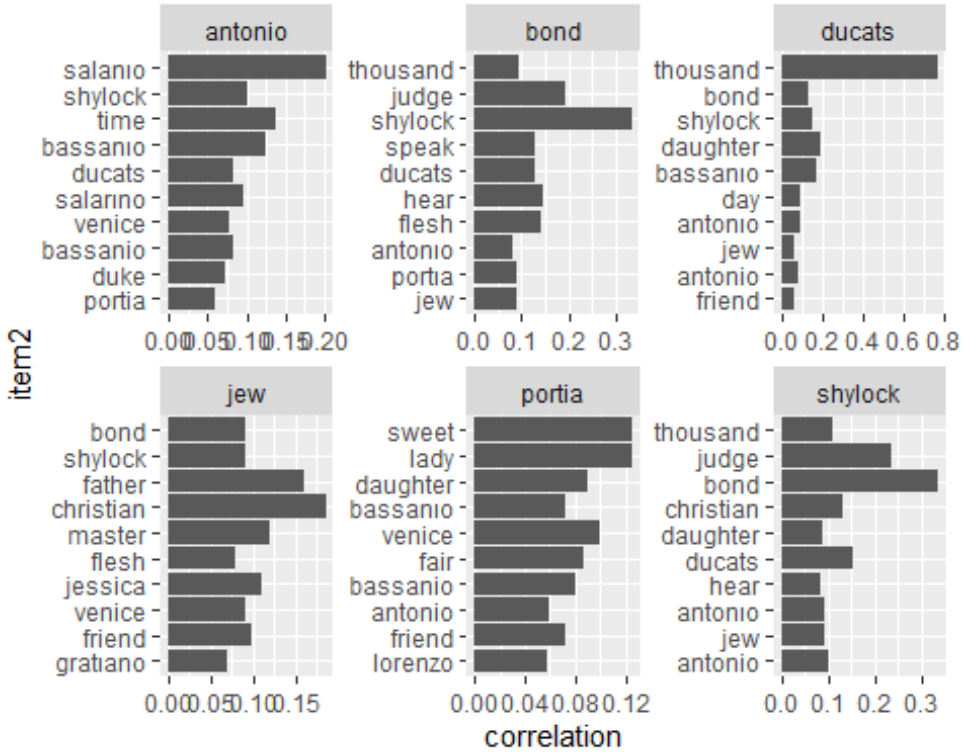
Kelimeler arası kolerasyon değerlerine bakalım.

```
word_cors <- venice_section_words %>%
  group_by(word) %>%
  filter(n() >= 20) %>%
  pairwise_cor(word, section, sort = TRUE)
```

```
word_cors
```

```
## # A tibble: 3,540 x 3
##    item1     item2     correlation
##    <chr>     <chr>           <dbl>
##  1 ducats    thousand        0.769
##  2 thousand  ducats          0.769
##  3 gobbo     launcelot       0.604
##  4 launcelot gobbo           0.604
##  5 salarino  salanio         0.557
##  6 salanio   salarino        0.557
##  7 enter     scene           0.467
##  8 scene     enter           0.467
##  9 father    launcelot       0.466
## 10 launcelot father          0.466
## # ... with 3,530 more rows
```

```
word_cors %>%
  filter(item1 %in% c("bond", "shylock", "jew", "ducats","law","antonio","portia")) %>%
  group_by(item1) %>%
  slice_max(correlation, n = 10) %>%
  ungroup() %>%
  mutate(item2 = reorder(item2, correlation)) %>%
  ggplot(aes(item2, correlation)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ item1, scales = "free") +
  coord_flip()
```



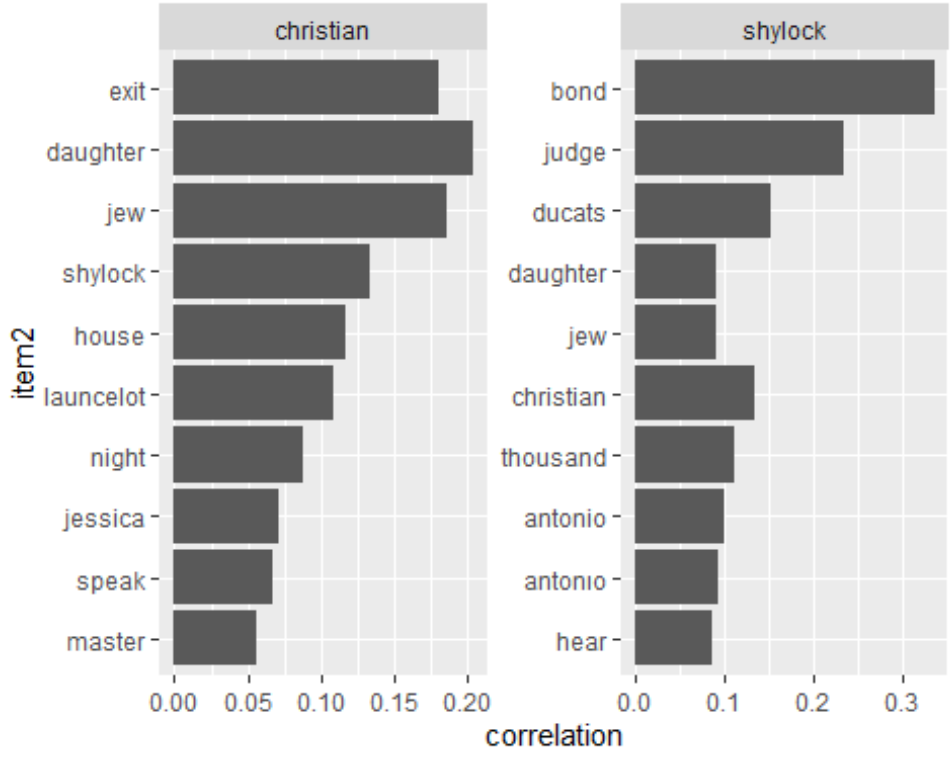
Kolerasyon grafiklerine göre;

- **İlk grafikte** Antonio, Shylock ve Salanio arasındaki diyalogların sıkça geçtiğini görüyoruz. Ayrıca Antonio dukalarla ve zamanla da ilişkili çıkmıştır. Buradan Antonio'nun aslında borcu alan kişi olduğunu düşünebiliriz çünkü senetler belli bir zaman içerisinde ödenmelidir ve Antonio zamanla ilişkisi çıkmıştır.
- **İkinci grafikte** senetle birlikte bin değeri ve dukalar ilişkili çıkmıştır. Daha önce de bahsettiğimiz gibi senetin 1000 duka olduğu sonucuna varabiliriz. Ayrıca yargıç kelimesiyle de senet kelimesi ilişki içerisinde olduğundan bu borç olayının mahkemeye taşındığını tekrardan söyleyebiliriz. Senet aynı zamanda Shylock, Portia, Antonio ve yahudiyle de ilişkisi vardır. Yani bu borç olaylarında Yahudi, Portia, Antonio ve Shylock vardır.
- **Üçüncü grafikte** dukalarla Antonio ve Shylock arasında ilişki çıkmıştır. Bu borç alışverişinin bu ikili arasında yaşandığını söyleyebiliriz. Çünkü eserde Antonio ve Antonıo aynı kişilerdir. Dosyamızı R programına çekerken oluşan yazım hatasından ötürü iki farklı kişi olarak görünüyor. Bu yüzden bu ikisinin kolerasyonunu birleştirdiğimizde oldukça ilişkili çıkmaktadır. Dukalar senetle de ilişkili çıkmıştır. Bu zaten beklenilen bir durum çünkü duka bir para birimi ve senetler de para birimleri üzerinden yapılır.
- **Dördüncü grafikte** Yahudi, Shylock ve Christian karakterleri ilişkili çıkmaktadır. Bu iki karakterin Yahudi olduğunu düşünebiliriz.
- **Beşinci grafikte** Portia lady ile ilişkili çıktığından soylu biri olduğunu düşünebiliriz ayrıca tatlı ve dürüst kelimeleriyle de ilişkili olduğundan Portia'nın iyi bir karakter olduğunu düşünebiliriz. Ayrıca Antonio, Bassanio ve Lorenzo ile ilişkisi oldukça fazladır. Bu kahramanların arkadaş olduğu sonucuna varabiliriz.
- **Altıncı grafikte** Shylock senetle ve dukalarla ilişkili çıkmıştır. Borç alışverişinde etkin bir kahraman olduğu sonucuna varabiliriz. Yargıçla, Yahudilikle ve Antonio'yla yüksek bir ilişkisi olduğundan borç veren yahudinin Shylock olduğunu düşünebiliriz.

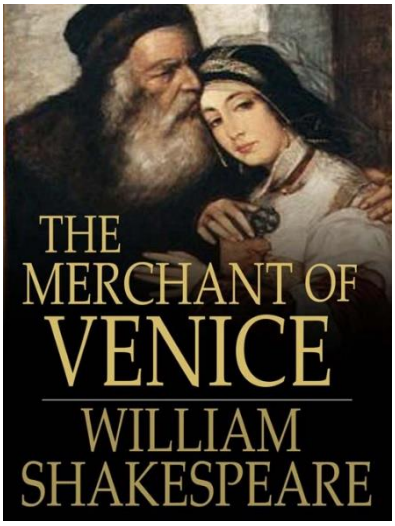
Altı grafiği tamamen incelediğimizde Yahudi, senetlerle, Shylockla ve dukalarla ilişkilidir. Yani borç alışverişinde yahudinin etkin bir rol oynadığını düşünebiliriz ve 2 yahudi adayımız vardı. Bunlar Christian ve Shylock. Ayrıca Antonio'nun borç alan kişi olduğunu düşünüyoruz çünkü zamanla oldukça ilişkiliydi. Yani bu borç olayı Yahudi ve Antonio arasında gerçekleşmiş olabilir ve bu alışverişte borç alan kişi Antonio'dur.

Şimdi ise hangi Yahudi adayımızın borç verdiğiyle ilgili uygun kolerasyon grafiğini oluşturarak kıyaslamasını yapalım.

```
word_cors %>%
  filter(item1 %in% c("shylock", "christian")) %>%
  group_by(item1) %>%
  slice_max(correlation, n = 10) %>%
  ungroup() %>%
  mutate(item2 = reorder(item2, correlation)) %>%
  ggplot(aes(item2, correlation)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ item1, scales = "free") +
  coord_flip()
```

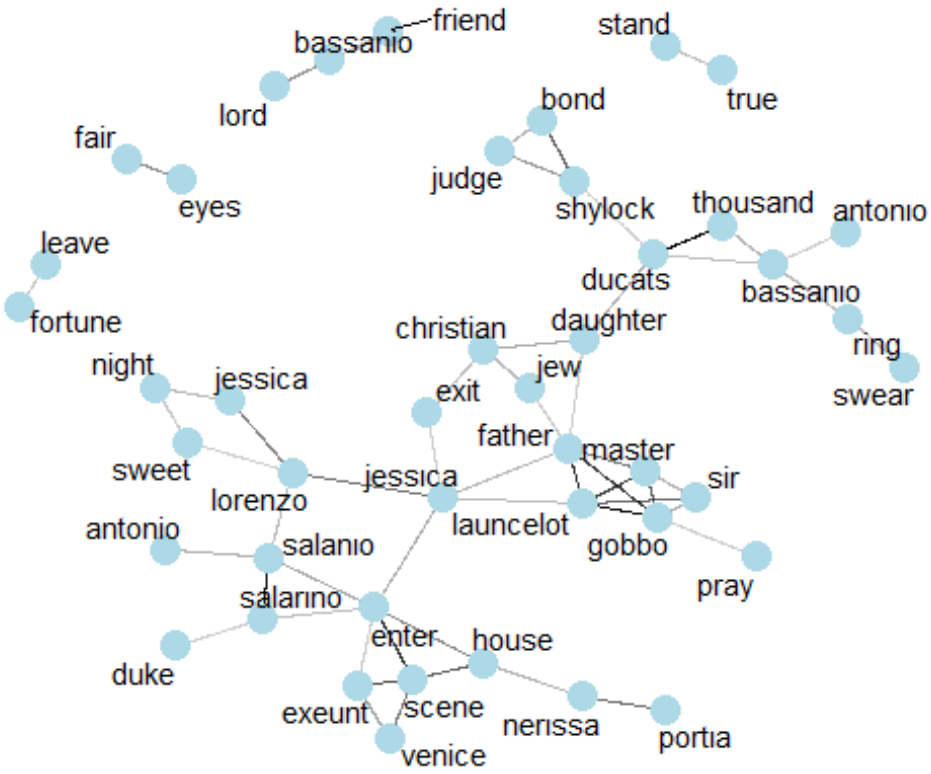


Grafikleri incelediğimizde Shylock senetlerle, yargıçla, dukalarla ve Antonio'yla ilişkiliyken Christian sadece bu olay örgüsü için sadece Yahudilikle ve Shylockla ilişkilidir. Yani borç veren kişinin Shylock olduğu ve Christian ile Shylock'un arkadaş olduğu sonucuna varabiliriz.



ESERIN GERCEK OZETI: Armatör Antonio nakit sıkıntısına düşmüş, arkadaşının sevgilisi Portia' yı gönderebilmek için bir vakitler hakaretler yağdırdığı Yahudi tefeci Shylock' tan üç bin duka borç altın almıştır. Yahudi Tefeci Shylock ise Antonio'nun borcu ödeyememesi halinde, vücudunun neresinden isterse orasından, yarım kilo eti keseceğine dair sözleşmeyi senedin sonuna şart olarak ekletmişti. Antonio' nun gemileri teker teker batmış, Soylu tüccar Antiono ,Tefeci Shylock' a. borcunu ödeyememiştir. Shylock, " ***Hakkımı isterim***, ***senette ne yazıyorsa onu isterim***!" diye Antonio'yu mahkemeye vermiştir. Shylock, Antonio 'dan ya parasını, ya da vücudundan yarım kilo et parçasını vermesini istemektedir. Antonio kendisini savunan genç bir avukatın zekâsı sayesinde davayı kazanmıştır.

```r
set.seed(2016)

word_cors %>%
  filter(correlation > .15) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```



jonson

Kütüphaneler

```
library(tidytext)
library(dplyr)
library(tidyverse)
library(gutenbergr) # Eserleri içerir
library(tidyr)  # Düzenli veri oluşturma
library(stopwords) # İstenmeyen kelimeleri çıkartma
library(ggplot2)  # Görselleştirme
library(wordcloud) # Kelime Bulutu
library(igraph)  # Ağları analiz etme
library(ggraph) # Katmanlı ağ görselleştirmesi
library(reshape2) # Matris dönüşümü
```

Gerekli kütüphaneleri indirdik.

```
epicoene <- gutenberg_download((4011), meta_fields = "title")
```

Gutenberg kütüphanesinden analizini yapmak üzere 4011 kitap numarasına sahip Ben Jonson'un Epicoene Sessiz Kadınlar adlı eserini indiriyoruz.

Duraklama Kelimeleri

```
mystopwords <- tibble(word = c("ı","la","ıt","ın","thou","ı'll","ay","thy"))

tidy_epicoene <- epicoene %>%
  unnest_tokens(word, text) %>%
   anti_join(stop_words) %>%
   anti_join(mystopwords)
```

İlk olarak unnest_tokens ile eseri yani verimizi, her satırda tek bir kelime olacak şekilde böldük.Sonrasında ise stop_words ile duraklama kelimeleri olarak adlandırılan the,of,to gibi kelimeleri eledik. Ancak bu işlem, tüm duraklama kelimelerini elemek için yeterli olmadı.Bu sebeple kendi duraklama kelimelerimizi mystopwords şeklinde oluşturarak eserden eledik.

En Çok Kullanılan Kelimeler

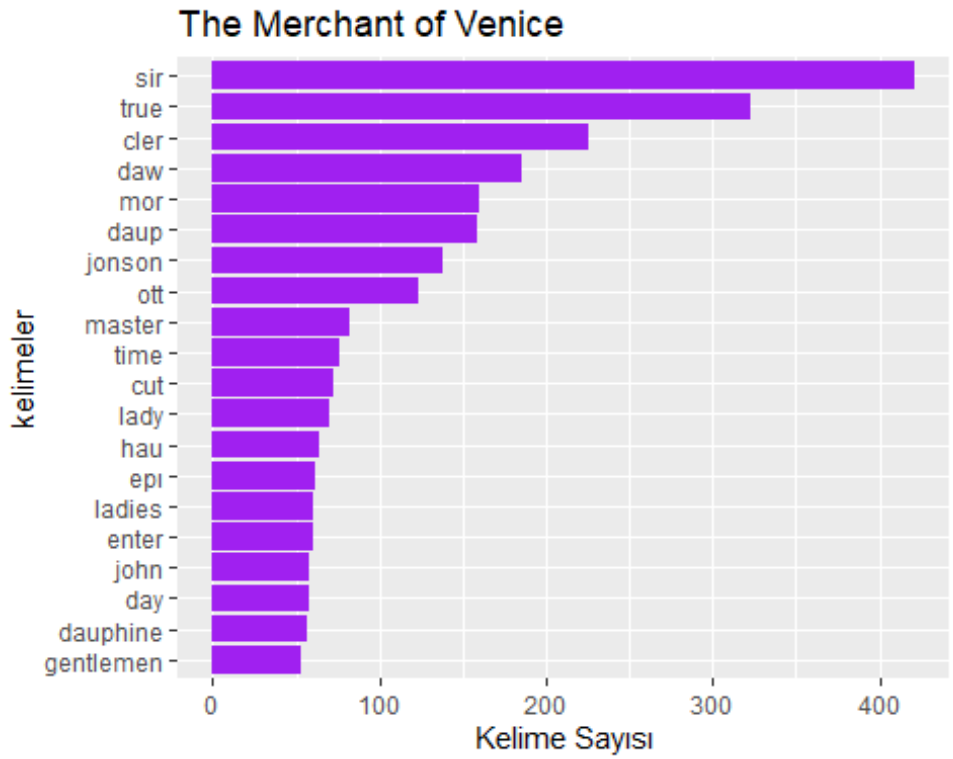
```
tidy_epicoene %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 8,559 x 2
##    word       n
##    <chr>  <int>
##  1 sir      421
##  2 true     323
##  3 cler     225
##  4 daw      185
##  5 mor      160
##  6 daup     159
##  7 jonson   138
##  8 ott      124
##  9 master    82
## 10 time      76
## # ... with 8,549 more rows
```

En çok kullanılan kelimeleri, kelime sıklığına göre sırasıyla görmekteyiz.İlk 20 kelimeyi inceleyelim.

Kelime Kullanılma Sıklığı Histogram

```
tidy <- epicoene %>%
  unnest_tokens(word,text) %>%
  count(word,sort=T) %>%
  anti_join(stop_words) %>%
  anti_join(mystopwords) %>%
  filter(!word %in% stop_words$word,

         str_detect(word, "[a-z]"))

tidy %>%
  head(20)%>%
  ggplot(aes(reorder(word,n),n))+
  geom_col(fill= "purple")+
  coord_flip()+
  labs(x="kelimeler",
       y="Kelime Sayısı",
       title= "The Merchant of Venice")
```



En çok kullanılan ilk 20 kelimeye baktığımızda mor,daup,john,daw,hau,epi,ott gibi kelimelerin eserdeki karakterler olduğunu söyleyebiliriz.Kelimeleri daha geniş çaplı incelediğimizde mor'un Morose kişisini, daup'un Dauphine kişisini ve diğer kişilerin isimlerinin de en sık kullanılan kelimelerde kısaltılmış hallerini temsil ettiğini görebilmekteyiz.Daw, şafak anlamına gelirken bunun bir karakter olduğunu anlamak için geniş çaplı bir analiz şarttır.Bunun dışında sir, master gibi kelimelerin de sık kullanıldığını görüyoruz.Bu da bize kahramanlar arası bir hiyerarşi olduğunu gösterir.Ayrıca lady,ladies,gentlemen gibi kelimelerden de kelime sıklık sıralamasına göre eserde kadınların ön planda olduğunu söyleyebiliriz.

Duygu Analizi

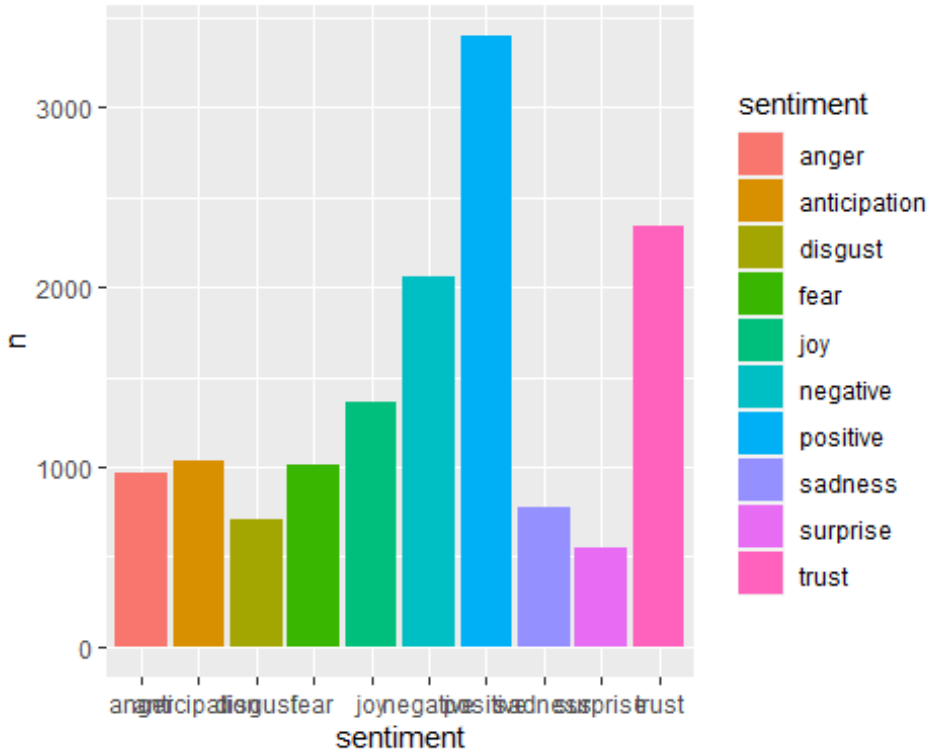
Nrc

```
tidy %>%
  inner_join(get_sentiments("nrc"))
```

```
## # A tibble: 3,909 x 3
##    word          n sentiment
##    <chr>     <int> <chr>
##  1 sir         421 positive
##  2 sir         421 trust
##  3 true        323 joy
##  4 true        323 positive
##  5 true        323 trust
##  6 master       82 positive
##  7 time         76 anticipation
##  8 john         58 disgust
##  9 john         58 negative
## 10 mistress     52 anger
## # ... with 3,899 more rows
```

Metindeki düşünce veya duyguyu değerlendirmek için var olan çeşitli yöntemler ve sözlükler vardır. Bu sözlüklerden biri olan nrc, duyguyu olumlu,olumsuz,sevinç,üzüntü,öfke vb şekilde ayırarak numaralandırır.

```
tidy %>%
  inner_join(get_sentiments("nrc")) %>%
  ggplot(aes(sentiment, n, fill= sentiment))+
  geom_col()
```



Grafiğe baktığımızda, esere genel olarak pozitif bir duygunun hakim olduğunu görmekteyiz.Karşılıklı güvenilir ilişkiler de söz konusudur. Bunun yanı sıra öfke,beklenti,iğrenme,korku,üzüntü,sevinç gibi duygular da

eserde yerini almıştır.Duyguların çoğunluğu negatif olmasına karşın bu sözlükte pozitif bir sonuç elde ettik.Şimdi bir de bing sözlüğünü inceleyelim.

AFİNN

```
AFINN <- get_sentiments("afinn")
AFINN
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # ... with 2,467 more rows
```

AFİNN sözlüğü, kelimeleri duygu yoğunluğuna göre -4 ila 4 arasında numaralandırır.

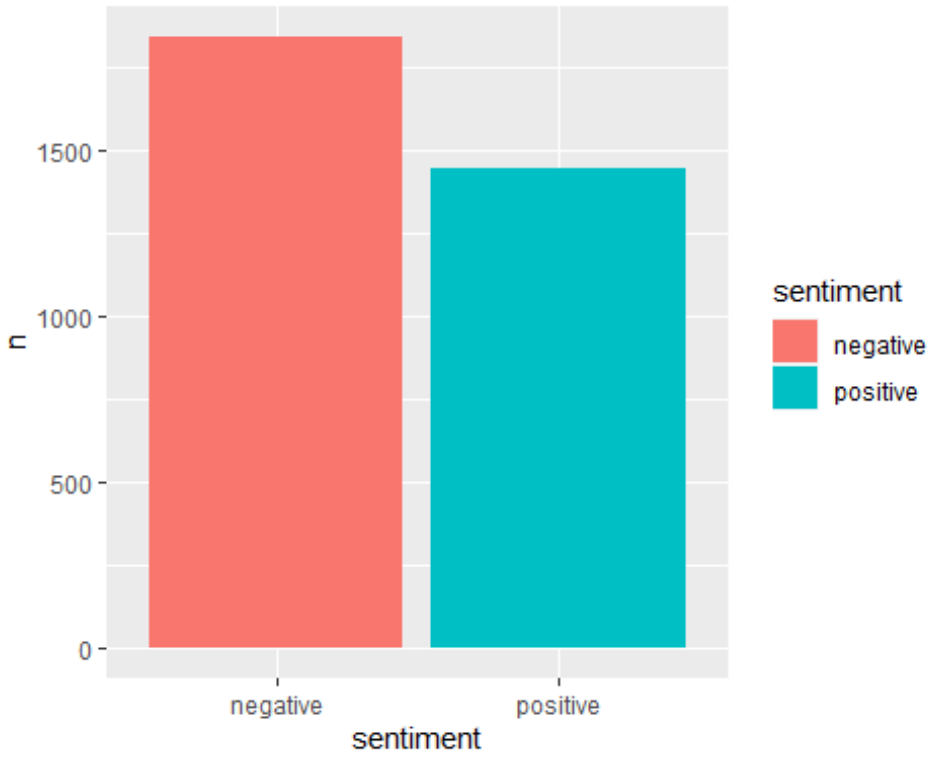
Bing

```
tidy %>%
  inner_join(get_sentiments("bing"))
```

```
## # A tibble: 1,244 x 3
##    word          n sentiment
##    <chr>     <int> <chr>
##  1 master       82 positive
##  2 mistress     52 negative
##  3 humour       44 positive
##  4 love         43 positive
##  5 faith        34 positive
##  6 fair         27 positive
##  7 gold         21 positive
##  8 noise        21 negative
##  9 excellent    20 positive
## 10 worth        20 positive
## # ... with 1,234 more rows
```

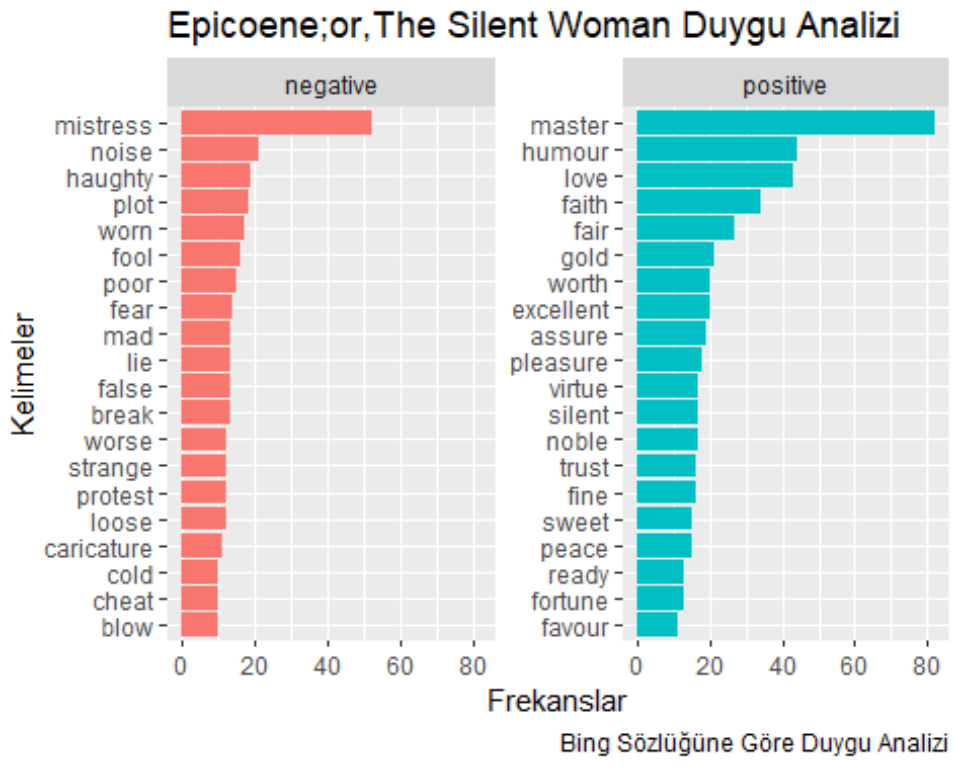
Bing sözlüğünde ise analizin, duyguları pozitif ve negatif şekilde ayırmaya yönelik yapıldığını görüyoruz.

```
tidy %>%
  inner_join(get_sentiments("bing")) %>%
  ggplot(aes(sentiment, n, fill= sentiment))+
  geom_col()
```



Grafiğe baktığımızda bu analizde nrc sözlüğünün aksine eserde negatif bir duygu yoğunluğunun olduğunu net bir şekilde görmekteyiz.

```
rbind(


tidy %>%
  inner_join(get_sentiments("bing")) %>%
  arrange(-n) %>%
  filter(sentiment == "positive") %>%
  head(20),


tidy %>%
  inner_join(get_sentiments("bing")) %>%
  arrange(-n) %>%
  filter(sentiment == "negative") %>%
  head(20)) %>%
  ggplot(aes(reorder(word,n),n, fill=sentiment)) +
  geom_col(show.legend = FALSE)+
  facet_wrap(~sentiment,scales = "free_y") +
  coord_flip() +
  labs( x = "Kelimeler" , y = "Frekanslar" , title = "Epicoene;or,The Silent Woman Duygu
Analizi" , caption = "Bing Sözlüğüne Göre Duygu Analizi" )
```



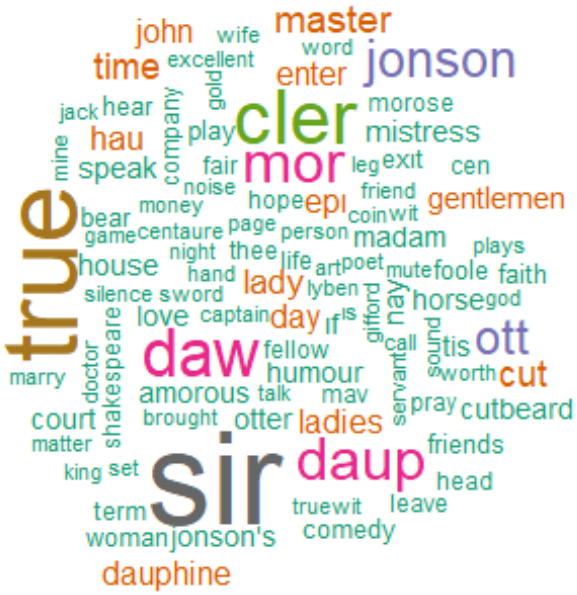
Bing sözlüğüne göre negatif ve pozitif duyguları detaylı incelediğimizde negatif duygularda ilk sıralarda arsa,yıpranmış kelimelerini görürken sonlara doğru protesto,hile,darbe gibi kelimeleri görmekteyiz.Bu kelimelerden; eserde arsa, miras sorunları olduğunu ve bu sorunların hile,darbe gibi kelimelerden yola çıkarak yakın çevre arası sorun ve çeşitli kötü oyunlara sebebiyet verdiğini söyleyebiliriz.Pozitif duygulara baktığımızda da altın,güven,servet,değer gibi kelimelerden de bu olayın izlerini görmekteyiz.Şimdi nrc sözlüğünü öfke kelimeleri için kullanalım.

```
nrc_anger <- get_sentiments("nrc") %>%
  filter(sentiment == "anger")


tidy_epicoene %>%
  inner_join(nrc_anger) %>%
  count(word, sort = TRUE)

## # A tibble: 344 x 2
##    word          n
##    <chr>     <int>
##  1 mistress     52
##  2 court        46
##  3 bear         36
##  4 money        20
##  5 haughty      19
##  6 fear         14
##  7 disease      13
##  8 lie          13
##  9 mad          13
## 10 words        12
## # ... with 334 more rows
```

Sonuçlara baktığımızda mahkeme,para gibi kelimeler ilk sırada yerini aldı.Buradan da arsa ile ilgili sorunun mahkeme ile bağlantılı olabileceğini söyleyebiliriz.

Kelime Bulutu

```
tidy_epicoene %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100,colors=brewer.pal(8, "Dark2")))
```



Kelime bulutuna baktığımızda ise kelimelerin kullanım sıklığına göre boyutlarının değiştiğini görmekteyiz.İlk 100 kelimeye baktığımızda karakterlerin isimlerine dair kısaltmaları ve orjinallerini daha net

görüyoruz.Bunun dışında bir evlilik ve ayrılığın olduğunu, dua etme,inanç kelimelerinden bir inanış söz konusu olduğunu söyleyebiliriz.

```
tidy_epicoene %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("gray10", "gray60"),
                   max.words = 100)
```

comporison.cloud fonksiyonu ile belgelerdeki sözcüklerin sıklığını karşılaştıran bir bulut oluşturduk.acast fonksiyonu ise bir veri çerçevesi oluşturmamıza yardımcı oldu.Gafikte ise kelime boyutu arttıkça kullanım sıklığının arttığını, renk tonu arttıkça ise negatif duygu durumunun arttığını söyleyebiliriz.

```
bingnegative <- get_sentiments("bing") %>%
  filter(sentiment == "negative")


wordcounts <- tidy_epicoene %>%
  group_by(title, gutenberg_id) %>%
  summarize(words = n())


tidy_epicoene %>%
  semi_join(bingnegative) %>%
  group_by(title, gutenberg_id) %>%
  summarize(negativewords = n()) %>%
  left_join(wordcounts, by = c("title", "gutenberg_id")) %>%
  mutate(ratio = negativewords/words) %>%
  filter(gutenberg_id != 0) %>%
  slice_max(ratio, n = 1) %>%
  ungroup()

## # A tibble: 1 x 5
##   title                          gutenberg_id negativewords words  ratio
##   <chr>                                 <int>         <int> <int>  <dbl>
## 1 Epicoene; Or, The Silent Woman         4011          1842 22196 0.0830
```

İlk olarak bing sözlüğüne göre negatif kelimeleri filtreledik.Sonrasında bölümlerin uzunluklarını normalleştirebilmemiz için her bölümde kaç kelime olduğuna dair wordcounts adında bir veri çerçevesi elde ettik.Son olarak ise her bölümdeki olumsuz kelime sayısının toplam kelimelere oranını elde ettik.Bu oran 0.08 oldu.Aynı işlemi pozitif kelimeler için de yapalım.

```
bingpositive <- get_sentiments("bing") %>%
  filter(sentiment == "positive")


wordcounts <- tidy_epicoene %>%
  group_by(title, gutenberg_id) %>%
  summarize(words = n())


tidy_epicoene %>%
  semi_join(bingpositive) %>%
  group_by(title, gutenberg_id) %>%
  summarize(positivewords = n()) %>%
  left_join(wordcounts, by = c("title", "gutenberg_id")) %>%
  mutate(ratio = positivewords/words) %>%
  filter(gutenberg_id != 0) %>%
  slice_max(ratio, n = 1) %>%
  ungroup()

## # A tibble: 1 x 5
##   title                          gutenberg_id positivewords words  ratio
##   <chr>                                 <int>         <int> <int>  <dbl>
## 1 Epicoene; Or, The Silent Woman         4011          1444 22196 0.0651
```

Aynı işlemi pozitif kelimelerde uyguladığımızda oranın 0.06 olduğunu görüyoruz.Buradan da negatif duygunun baskın olduğu sonucuna ulaşıyoruz.

NGRAMLAR (n=2)

```
epicoene_bigrams <- epicoene %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)
epicoene_bigrams
```

```
## # A tibble: 47,383 x 3
##    gutenberg_id title                          bigram
##           <int> <chr>                          <chr>
##  1         4011 Epicoene; Or, The Silent Woman epıcoene or
##  2         4011 Epicoene; Or, The Silent Woman or the
##  3         4011 Epicoene; Or, The Silent Woman the sılent
##  4         4011 Epicoene; Or, The Silent Woman sılent woman
##  5         4011 Epicoene; Or, The Silent Woman <NA>
##  6         4011 Epicoene; Or, The Silent Woman <NA>
##  7         4011 Epicoene; Or, The Silent Woman by ben
##  8         4011 Epicoene; Or, The Silent Woman ben jonson
##  9         4011 Epicoene; Or, The Silent Woman <NA>
## 10         4011 Epicoene; Or, The Silent Woman <NA>
## # ... with 47,373 more rows
```

n-gram ile n sayıda ardışık sözcük dizinini bir araya getiririz. X kelimesinin ardından Y kelimesinin ne sıklıkla geldiğini görerek, aralarındaki ilişkilerin bir modelini oluşturabiliriz. n=2 aldığımız bu durum bigram olarak adlandırılır.

```
epicoene_bigrams %>%
  count(bigram, sort = TRUE)
```

```
## # A tibble: 29,474 x 2
##    bigram      n
##    <chr>   <int>
##  1 <NA>     3636
##  2 of the    214
##  3 in the    169
##  4 to the    122
##  5 la f      100
##  6 of a       81
##  7 to be      78
##  8 ı will     71
##  9 ı have     67
## 10 for the    58
## # ... with 29,464 more rows
```

Bir araya gelme sıklıklarına göre kelimeleri ikişerli şekilde sıraladığımızda duraklama kelimelerini ön planda görmekteyiz.Bu kelimelerden kurtulmak için öncelikle kelime gruplarını 2 sütuna ayırmalıyız.

```
bigrams_separated <- epicoene_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")


bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word1 %in% mystopwords$word) %>%
  filter(!word2 %in% mystopwords$word)


# new bigram counts:
bigram_counts <- bigrams_filtered %>%
  count(word1, word2, sort = TRUE)


bigram_counts
```

```
## # A tibble: 6,374 x 3
##    word1  word2        n
##    <chr>  <chr>    <int>
##  1 <NA>   <NA>      3636
##  2 sir    john        46
##  3 sir    amorous     37
##  4 sir    dauphine    28
##  5 john   daw         19
##  6 ben    jonson      18
##  7 master truewit     16
##  8 jack   daw         14
##  9 true   nay         14
## 10 master doctor      13
## # ... with 6,364 more rows
```

seperate, sütunları bir sınırlayıcıya göre birden çok bölüme ayıran bir tidyr fonksiyonudur.Bu fonksiyon ile sütunları ayırdıktan sonra durdurma kelimelerini filter ile eledik.Bu eserde karakterlerin en yaygın çitfler olduğunu söyleyebiliriz.

```
bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")


bigrams_united
```

```
## # A tibble: 10,714 x 3
##    gutenberg_id title                          bigram
##           <int> <chr>                          <chr>
##  1         4011 Epicoene; Or, The Silent Woman sılent woman
##  2         4011 Epicoene; Or, The Silent Woman NA NA
##  3         4011 Epicoene; Or, The Silent Woman NA NA
##  4         4011 Epicoene; Or, The Silent Woman ben jonson
##  5         4011 Epicoene; Or, The Silent Woman NA NA
##  6         4011 Epicoene; Or, The Silent Woman NA NA
##  7         4011 Epicoene; Or, The Silent Woman NA NA
##  8         4011 Epicoene; Or, The Silent Woman NA NA
##  9         4011 Epicoene; Or, The Silent Woman NA NA
## 10         4011 Epicoene; Or, The Silent Woman NA NA
## # ... with 10,704 more rows
```

unite fonksiyonu, seperatenin tam tersi işlevde olup ayrı olan kelime sütunlarını tekrar birleştirebilmemizi sağlar.

(n=3)

```
epicoene %>%
  unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
  separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word,
         !word3 %in% stop_words$word,
         !word1 %in% mystopwords$word,
         !word2 %in% mystopwords$word,
         !word3 %in% mystopwords$word) %>%
  count(word1, word2, word3, sort = TRUE)
```

```
## # A tibble: 2,176 x 4
##    word1    word2    word3        n
##    <chr>    <chr>    <chr>    <int>
##  1 <NA>     <NA>     <NA>      4253
##  2 sir      john     daw         15
##  3 gold     coin     worth        5
##  4 haughty  centaure mavıs        4
##  5 enter    captaın  otter        3
##  6 enter    morose   wıth         3
##  7 protest  sir      john         3
##  8 true     alas     sir          3
##  9 centaure mavıs    mıstress     2
## 10 centaure mistress dol          2
## # ... with 2,166 more rows
```

Trigram sonuçlarında da karakter isimlerinin çoğunlukta olduğunu görmekteyiz.Bu da bize karşılıklı diyalogların çok yani bir tiyatro oyunu olduğu ipucunu verir.

```
bigrams_filtered %>%
  filter(word2 == "house") %>%
  count(title, word1, sort = TRUE)
```

```
## # A tibble: 9 x 3
##   title                          word1          n
##   <chr>                          <chr>      <int>
## 1 Epicoene; Or, The Silent Woman morose's        6
## 2 Epicoene; Or, The Silent Woman bake            1
## 3 Epicoene; Or, The Silent Woman banquetting     1
## 4 Epicoene; Or, The Silent Woman clerımont's     1
## 5 Epicoene; Or, The Silent Woman daw's           1
## 6 Epicoene; Or, The Silent Woman lazar           1
## 7 Epicoene; Or, The Silent Woman otter's         1
## 8 Epicoene; Or, The Silent Woman public          1
## 9 Epicoene; Or, The Silent Woman tyrıng          1
```

İkili kelime gruplarında house kelimesini filtreleyerek olayın genel olarak Morose karakterinin evinde geçtiğini görüyoruz.

```
bigram_tf_idf <- bigrams_united %>%
  count(title, bigram) %>%
  bind_tf_idf(bigram, title, n) %>%
  arrange(desc(tf_idf))


bigram_tf_idf
```

```
## # A tibble: 6,374 x 6
##    title                          bigram              n        tf   idf tf_idf
##    <chr>                          <chr>           <int>     <dbl> <dbl>  <dbl>
##  1 Epicoene; Or, The Silent Woman __ incidentally     1 0.0000933     0      0
##  2 Epicoene; Or, The Silent Woman 10 shillings        1 0.0000933     0      0
##  3 Epicoene; Or, The Silent Woman 100 variously       1 0.0000933     0      0
##  4 Epicoene; Or, The Silent Woman 1592 jonson         1 0.0000933     0      0
##  5 Epicoene; Or, The Silent Woman 1593 marlowe        1 0.0000933     0      0
##  6 Epicoene; Or, The Silent Woman 1597 paying         1 0.0000933     0      0
##  7 Epicoene; Or, The Silent Woman 1598 jonson         1 0.0000933     0      0
##  8 Epicoene; Or, The Silent Woman 1601 02             1 0.0000933     0      0
##  9 Epicoene; Or, The Silent Woman 1605 volpone        1 0.0000933     0      0
## 10 Epicoene; Or, The Silent Woman 1609 fol            1 0.0000933     0      0
## # ... with 6,364 more rows
```

```
bigrams_separated %>%
  filter(word1 == "no") %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 108 x 3
##    word1 word2     n
##    <chr> <chr> <int>
##  1 no    ı        11
##  2 no    more     11
##  3 no    sir       8
##  4 no    man       7
##  5 no    means     7
##  6 no    noise     5
##  7 no    such      5
##  8 no    faith     4
##  9 no    no        4
## 10 no    other     4
## # ... with 98 more rows
```

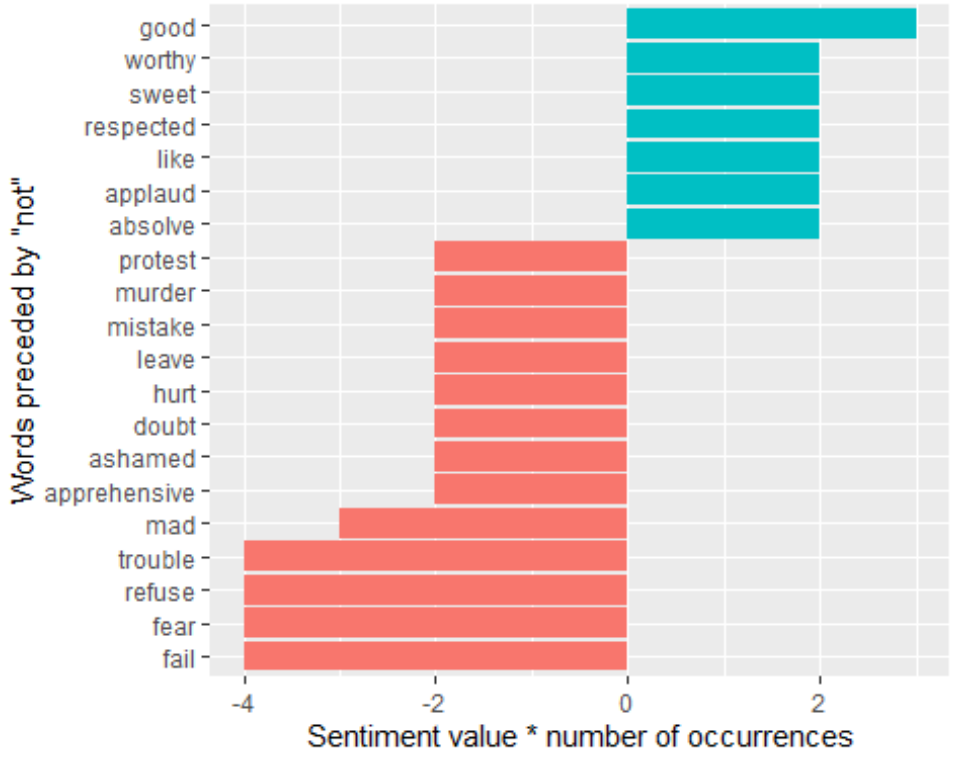
İkili kelime grubu olan bigramlardan “no” olumsuzluk kelimesini filtrelediğimizde gürültü ve inanç kelimesi dikkatimizi çekiyor. Eserde gürültülü ortamlar bulunup bu ortamların hoş karşılanmadığını söyleyebiliriz.Kelime bulutunda karşılaştığımız inanç ve dua etme sözcüklerine zıt olarak burada inanç kelimesi olumsuz bir durum olarak sonuç verdi.Bu da bize karakterlerin belli bir inanca sahip olduğu genellemesini yapamayacağımızı gösteriyor.

```
not_words <- bigrams_separated %>%
  filter(word1 == "not") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, value, sort = TRUE)
not_words
```

```
## # A tibble: 29 x 3
##    word2        value     n
##    <chr>        <dbl> <int>
##  1 doubt           -1     2
##  2 fail            -2     2
##  3 fear            -2     2
##  4 leave           -1     2
##  5 refuse          -2     2
##  6 trouble         -2     2
##  7 absolve          2     1
##  8 allow            1     1
##  9 applaud          2     1
## 10 apprehensive    -2     1
## # ... with 19 more rows
```

not kelimesiyle bir araya gelen kelimelerin AFINN sözlüğüne göre puanlamasını görüyoruz.Burada iyi anlamına gelen good kelimesinin değil anlamına gelen not kelimesinden sonra 3 (pozitif) AFINN puanıyla yer alması,onun yanlış yöne ne kadar katkıda bulunduğunun bir göstergesidir.

```
not_words %>%
  mutate(contribution = n * value) %>%
  arrange(desc(abs(contribution))) %>%
  head(20) %>%
  mutate(word2 = reorder(word2, contribution)) %>%
  ggplot(aes(n * value, word2, fill = n * value > 0)) +
  geom_col(show.legend = FALSE) +
  labs(x = "Sentiment value * number of occurrences",
       y = "Words preceded by \"not\"")
```



“iyi olmamak” bigramı yanlış tanımlamalara sebep oldu.Çünkü “iyi” kelimesi pozitif bir kelime olduğundan “not”

yanında yer alarak eserin olumlu sonuçlar vermesine katkı sağladı. Sonuç olarak bu dört olumsuzlamanın yanında olan her kelime analiz sonuçlarımızı yanlış bir şekilde etkiledi.

```
negation_words <- c("not", "no", "never", "without")


negated_words <- bigrams_separated %>%
  filter(word1 %in% negation_words) %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word1, word2, value, sort = TRUE)
negated_words
```

```
## # A tibble: 51 x 4
##    word1 word2   value     n
##    <chr> <chr>   <dbl> <int>
##  1 no    faith       1     4
##  2 no    no         -1     4
##  3 no    comfort     2     2
##  4 no    matter      1     2
##  5 no    noble       2     2
##  6 not   doubt      -1     2
##  7 not   fail       -2     2
##  8 not   fear       -2     2
##  9 not   leave      -1     2
## 10 not   refuse     -2     2
## # ... with 41 more rows
```

“not” bir sonraki terimi olumsuzlaştıran tek bağlam değildir. No,never,without ile de yandaki terimleri olumsuzlaştırmak mümkündür.Bu tabloda, özünde olumlu olup aslında olumsuz olan ve yanlış analiz sonuçlar alınmasına sebep olan tüm kelimeleri görmekteyiz.

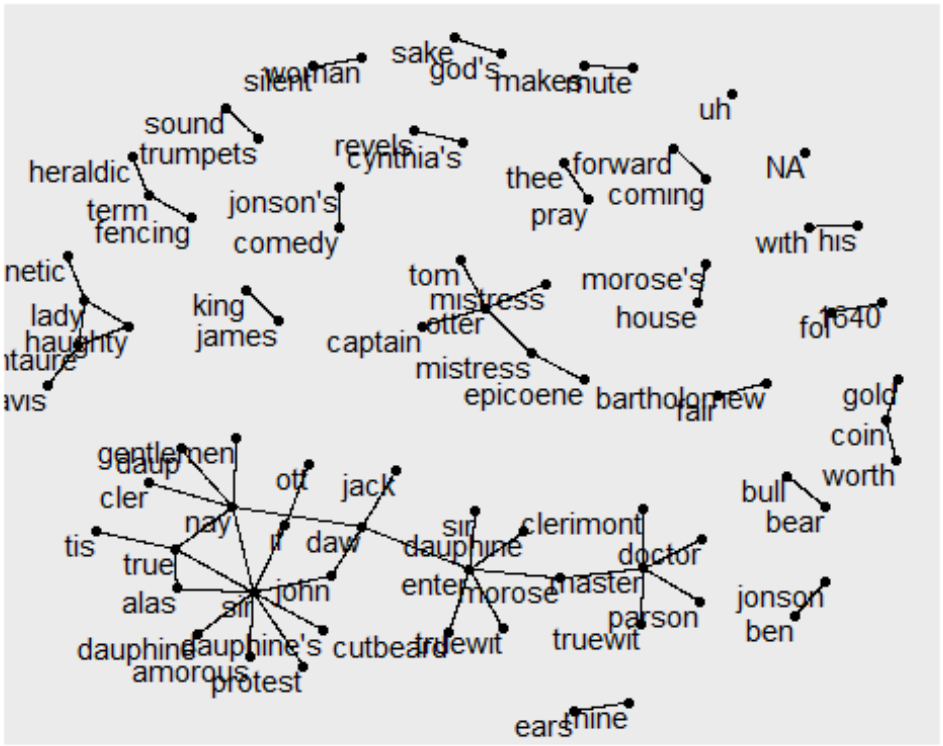
```
bigram_graph <- bigram_counts %>%
  filter(n >3) %>%
  graph_from_data_frame()

## Warning in graph_from_data_frame(.): In `d' `NA' elements were replaced with
## string "NA"

bigram_graph

## IGRAPH bbdd7ab DN-- 79 62 --
## + attr: name (v/c), n (e/n)
## + edges from bbdd7ab (vertex names):
##  [1] NA      ->NA        sir     ->john      sir     ->amorous
##  [4] sir     ->dauphine  john    ->daw       ben     ->jonson
##  [7] master  ->truewit   jack    ->daw       true    ->nay
## [10] master  ->doctor    master  ->parson    coin    ->worth
## [13] cler    ->nay       enter   ->truewıt   lady    ->haughty
## [16] mute    ->makes     king    ->james     master  ->clerimont
## [19] mistress->otter     mıstress->otter     daw     ->nay
## [22] gold    ->coin      enter   ->morose    fencing ->term
## + ... omitted several edges
```

Tabloda en yaygın ikilemeleri yani bigramları görmekteyiz.Buradan; Otter karakterinin bir kaptan olduğunu,olayın geçtiği evin Morose karakterine ait olduğunu,olayın ev dışında Bartholomew adında bir fuarda da geçtiğini söyleyebiliriz. Ayrıca Cynthia’s revels ikilemesinden bir eğlence olduğunu ve silent woman ikileminden ise sessiz bir kadın olduğunu görüyoruz. Önceki yorumlarımızda “no” olumsuzlaştırmasının yanında gürültü kelimesini görmüştük.Bu eğlencenin bir gürültüye sebep olduğunu ve yine önceki analiz sonuçlarımıza göre bir boşanmaya sebebiyet verebileceği varsayımında bulunabiliriz.Hatırlarsak house kelimesini filtrelediğimizde ilk sırada Morose karakterini

görmüştük. Tüm sonuçları birleştirdiğimizde bu evliliğin Morose karakterine ait olduğunu da düşünebiliriz.

```
set.seed(2019)
```

```
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```



Olaylar arasındaki ilişkileri bu ağ görselleştirmesiyle de görebiliriz.

```
epicoene_section_words <- epicoene %>%
  mutate(section = row_number() %/% 10) %>%
  filter(section > 0) %>%
  unnest_tokens(word, text) %>%
  filter(!word %in% stop_words$word)


epicoene_section_words
```

```
## # A tibble: 23,338 x 4
##    gutenberg_id title                          section word
##           <int> <chr>                            <dbl> <chr>
##  1         4011 Epicoene; Or, The Silent Woman       1 ıntroductıon
##  2         4011 Epicoene; Or, The Silent Woman       1 english
##  3         4011 Epicoene; Or, The Silent Woman       1 dramatists
##  4         4011 Epicoene; Or, The Silent Woman       1 shakespeare
##  5         4011 Epicoene; Or, The Silent Woman       1 literary
##  6         4011 Epicoene; Or, The Silent Woman       1 dictator
##  7         4011 Epicoene; Or, The Silent Woman       1 poet
##  8         4011 Epicoene; Or, The Silent Woman       1 laureate
##  9         4011 Epicoene; Or, The Silent Woman       1 writer
## 10         4011 Epicoene; Or, The Silent Woman       1 verse
## # ... with 23,328 more rows
```

Burada eseri 10ar satırlık bölümlere ayırdık ve hangi kelimelerin görünme eğiliminde olduğunu analiz ettik.

```
library(widyr)
word_pairs <- epicoene_section_words %>%
  anti_join(mystopwords) %>%
  pairwise_count(word, section, sort = TRUE)
word_pairs
```

```
## # A tibble: 438,602 x 3
##    item1 item2     n
##    <chr> <chr> <dbl>
##  1 sir   true    115
##  2 true  sir     115
##  3 daw   sir      82
##  4 sir   daw      82
##  5 mor   sir      79
##  6 sir   mor      79
##  7 cler  true     78
##  8 cler  sir      78
##  9 true  cler     78
## 10 sir   cler     78
## # ... with 438,592 more rows
```

Widyr paketinin yararlı bir işlevi olan pairwise_count() fonksiyonu bize aynı bölümlerde ilişkili 2li kelimeleri, kullanılma sıklığına göre görüntüleme olanağı sağlar.Tablodan da karakterler arası diyalogların sıklık sayılarını görebiliriz.

```
word_pairs %>%
  filter(item1 == "morose")
```

```
## # A tibble: 363 x 3
##    item1  item2      n
##    <chr>  <chr>  <dbl>
##  1 morose sir       11
##  2 morose hau       11
##  3 morose mor       10
##  4 morose true       9
##  5 morose epı        9
##  6 morose enter      8
##  7 morose madam      8
##  8 morose master     7
##  9 morose wıth       6
## 10 morose cen        6
## # ... with 353 more rows
```

2li kelimelerden Morose karakterini filtrelediğimizde bu karakterin en çok ilişkili olduğu karakterleri görmemiz mümkündür.

```
word_cors <- epicoene_section_words %>%
  group_by(word) %>%
  filter(n() >= 20) %>%
  pairwise_cor(word, section, sort = TRUE)
```

```
word_cors
```

```
## # A tibble: 11,772 x 3
##    item1 item2 correlation
##    <chr> <chr>       <dbl>
##  1 cen   hau         0.604
##  2 hau   cen         0.604
##  3 hau   madam       0.581
##  4 madam hau         0.581
##  5 foole la          0.554
##  6 la    foole       0.554
##  7 coin  worth       0.512
##  8 worth coin        0.512
##  9 thee  thou        0.490
## 10 thou  thee        0.490
## # ... with 11,762 more rows
```

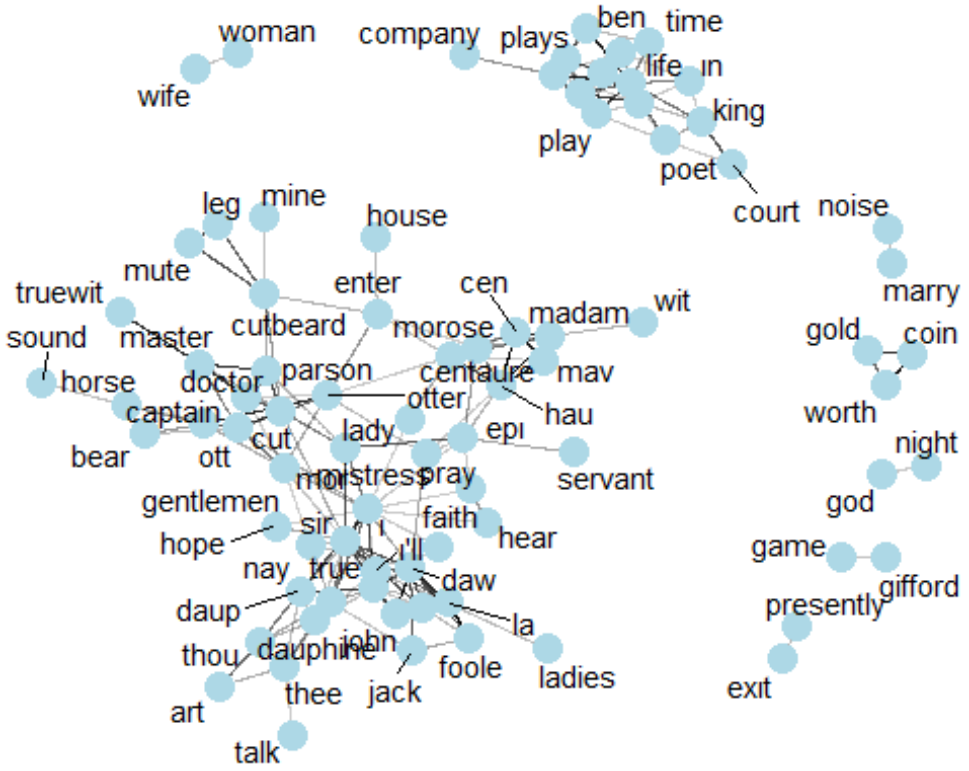
pairwise_cor fonksiyonu, bize ikili korelasyon için ortak bir ölçü olan phi katsayısını verir.Phi katsayısı, ikili verilere uygulandığında, Pearson korelasyonuna eşdeğerdir. Korelasyon katsayıları arttıkça ikili karakter ilişkilerini sık bir şekilde görüyoruz.Buradan da eserde diyalogların sık olduğunu anlamamız mümkündür.

```
word_cors %>%
  filter(item1 == "epɪ")
```

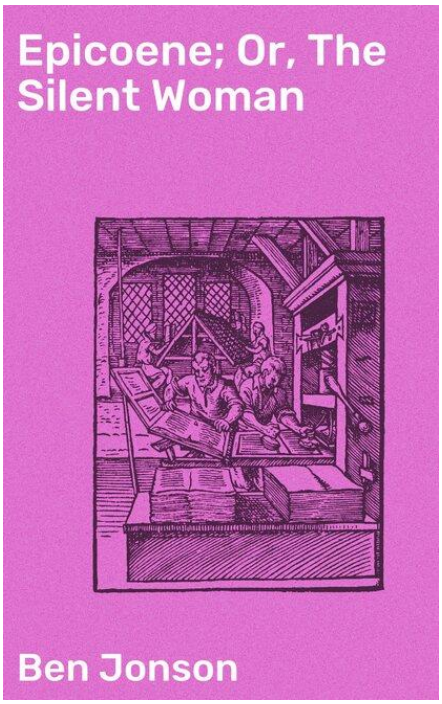
```
## # A tibble: 108 x 3
##    item1 item2    correlation
##    <chr> <chr>          <dbl>
##  1 epɪ   mor            0.305
##  2 epɪ   morose         0.217
##  3 epɪ   servant        0.202
##  4 epɪ   hau            0.185
##  5 epɪ   mistress       0.167
##  6 epɪ   ɪ              0.161
##  7 epɪ   pray           0.159
##  8 epɪ   lady           0.146
##  9 epɪ   cutbeard       0.134
## 10 epɪ   ladies         0.126
## # ... with 98 more rows
```

Epicoene karakterinin bigramlarında çıkan korelasyon değerine bakacak olursak korelasyon değerlerinin küçük olduğunu fakat ilk iki sırada Morose karakterinin yer aldığını görüyoruz.Önceki analizlerde Morose karakterinin bir evlilik geçirdiği varsayımından bahsetmiştik. Aynı analizde sessiz bir kadın profiliyle de karsılaşmıştık. Bu tabloya baktığımızda Morose karakterinin evlendiği ve sessiz bir yapıda olan bu kadının Epicoene karakteri olabileceği kanısına varabiliriz.

```
set.seed(2016)


word_cors %>%
  filter(correlation > .15) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = correlation), show.legend = FALSE) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()

## Warning: ggrepel: 7 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



Analizlerde bir evlilik ve boşanmanın söz konusu olduğunu söylemiştik.Hatta bu evliliğin Morose ile Epicoene karakterleri arasında olabileceği ve Epicoene karakterinin sessiz bir yapıda olduğu varsayımlarında bulunmuştuk. Grafikteki kelimelerin kümeleşmelerine baktığımızda noise ve marry eşleşmesi dikkatimizi çekebilir.Çünkü önceki analizlerde bu evliliğin gürültülü bir olay sonucu bitebileceğine değinmiştik.Epicoene karakteri hem sessiz hem de gürültü anlamına gelen kelimelerle eşleştiğine göre bu karakterde analizlerde sonuç alamadığımız bazı durumlar olduğu sonucunu çıkarabiliriz. Bu sonuç,gürültüden hiç hoşlanmayan Morose karakteriyle ayrılmalarına da sebep olmuş olabilir.Bunun dışında gold,coin,worth kümeleşmesini de görüyoruz.Bazı arsa problemleri olduğuna da değinmiştil. Sonuca bağlayacak olursak eserimiz başlıca bu 2 olayı ele almaktadır dıyebiliriz.



ESERIN GERCEK ÖZETİ:

Morose, arabaların geçemeyeceği ve gürültü çıkaramayacağı dar bir sokakta yaşayacak kadar ileri giden, gürültüye karşı takıntılı bir nefrete sahip zengin, yaşlı bir adamdır. Yeğeni Dauphine'i evlenerek mirastan mahrum etmek için planlar yapmıştır. Bunun nedeni, Dauphine'in geçmişte ona oynadığı entrikalar ve numaralardır. Dauphine, bu durumla mücadele etmek için, Morose'un berberi Cutbeard ile bir plan yapar. Cutbeard, Morose'a evlenmesi için genç (ve sözde) sessiz bir kadın sunar. Morose, Epicœne ile tanıştığında, onun gerçekten sessiz bir kadın olup olmadığını öğrenmeye çalışır ve onun itaatini sınar. Ona mahkemenin cazibelerine yenik düşmemesini söyler ve ona sessizliğin erdemlerinden bahseder. Nişanlısı Epicœne'nin son derece sessiz bir kadın olduğu varsayımı altında, Morose heyecanla evliliklerini planlar.

Truewit, arkadaşının mirasını güvence altına almayı umarak Morose'u evliliğin kendisi için iyi olmayacağına ikna etmeye çalışır. Çift, Dauphine'in arkadaşı Truewit'in iyi niyetli müdahalesine rağmen evlenir. Evlilik sonrası Morose'un evinde (Morose'un hakkında hiçbir şey bilinmeden ve gürültüden dolayı sıkıntıya girmesine neden olan) bir parti düzenlenir ve bu da birçok misafirin davet edilmesine yol açar. Morose artık Epicoene'nin gerçek yüzünü görmüştür.Onun aslında sessiz bir kadın olmadığını anlamıştır.Morose, boşanmak için yeğeni Dauphine'den yardım ister. Dauphine de ona bir teklifte bulunur. Ona, miras karşılığında bu boşanmayı sağlayabileceğini söyler. Morose çaresiz kabul eder ve miras devri için bir sözleşme imzalar. Sonrasında Dauphine, Epicoene'nin kostümünü çıkarır. Epicoene aslında bir erkektir ve bu evlilik zaten geçersizdir. Dauphine, amcasına Clerimont ve kaptan Otter ile büyük bir oyun oynamıştır.

KAYNAKÇA:

Gutenberg Projesi - Vikipedi (wikipedia.org)

https://www.nkfu.com/gutenberg-neyi-bulmustur-kisaca-hayati/

https://cran.r-project.org/web/packages/gutenbergr/vignettes/intro.html

http://metinmadenciligi.com/

Metin madenciliği - Vikipedi (wikipedia.org)

https://www.veribilimiokulu.com/r-ile-metin-madenciligi

https://web.archive.org/web/20160305115302/http://mis.sadievrenseker.com/2014/06/metin-madenciligi-text-mining/

kergun.baun.edu.tr/veri_madenciligi_hafta11.pdf,

Metin Madenciliği (Text Mining) (Veri Bilimcisi Olma Yolunda 38. Video) - YouTube

Draw insights from fiction books with Text Mining | Analytics Vidhya (medium.com)

1 The tidy text format | Text Mining with R (tidytextmining.com)

RPubs - Basic Text Mining in R

Advancing Text Mining with R and quanteda | R-bloggers (r-bloggers.com)

http://www.olaganustukanitlar.com/zipf-kanunu-nedir/

https://tr.wikipedia.org/wiki/Zipf_yasas%C4%B1

https://medium.com/@anilguven1055/latent-dirichlet-allocation-lda-algoritmas%C4%B1-13154d246e05